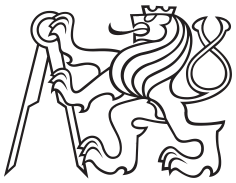
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Pokročilé metody detekce automobilů s využitím měření magnetického pole

Bc. Michaela Brejchová

Školitel: Ing. David Novotný
Obor: Kybernetika a robotika
Zaměření: Kybernetika a robotika
Květen 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Brejchová** Jméno: **Michaela** Osobní číslo: **456985**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Pokročilé metody detekce automobilů s využitím měření magnetického pole

Název diplomové práce anglicky:

Advanced methods for car detection using magnetic field measurement

Pokyny pro vypracování:

Ověřte možnosti využití korelačních metod, neuronových sítí a případně jiné metody pro zpracování magnetických měření za účelem detekce přítomnosti automobilu na parkovacím místě. Vhodné metody aplikujte na reálná data, poskytnutá školitelem z předešlých měření. Porovnejte a diskutujte výsledné parametry detekčních metod.

Seznam doporučené literatury:

[1] RIPKA, P.: Magnetic sensors and magnetometers. Boston: Artech House, 2001. ISBN 15-805-3057-5.
[2] Janošek M, Platil A, Vyhnánek J (2016): Simple estimation of dipole source z-distance with compact magnetic gradiometer, in IOP Conf. Series: Materials Science and Engineering 108 (2016) 012025 doi:10.1088/1757-899X/108/1/012025 (IC-MAST2015)
[3] Markevicius V et al. (2014): Vehicle Influence on the Earth's Magnetic Field Changes, Elektronika ir Elektrotechnika, ISSN 1392–1215, Vol. 20, No. 4, doi: 10.5755/j01.eee.20.4.4552
[4] Hongmei Zhu and Fengqi Yu (2015): Vehicle Parking Detection Method Based on Correlation of Magnetic Signals, International Journal of Distributed Sensor Networks, Volume 2015, Article ID 361242, doi: 10.1155/2015/361242
[5] Nara T, Suzuki S and Ando S (2006): A closed-form formula for magnetic dipole localization by measurement of its magnetic field and spatial gradients. IEEE Transactions on Magnetics, Volume 42, 3291-93, doi: 10.1109/TMAG.2006.879151

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Novotný, katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **10.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce:
do konce letního semestru 2020/2021

Ing. David Novotný podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studentky

Poděkování

Tímto bych chtěla poděkovat panu Ing. Davidovi Novotnému za vedení, vstřícnost a cenné rady, trpělivost při konzultacích a pomoc v době, kdy jsem kvůli krizovému stavu neměla přístup na fakultu.

Také bych chtěla poděkovat svému snoubenci a rodině za podporu, motivaci a hlavně trpělivost, kterou se mnou měli.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla všechny zdroje informací v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2020

Abstrakt

Cílem této práce je upravit stávající prototyp magnetického senzoru pro detekci obsazenosti parkovacího místa automobilem.

Změny se týkají hardwaru, ale především detekčního algoritmu. Za tímto účelem je vyzkoušeno několik možností detekce s využitím vzájemné korelace, neuronových sítí a metody nejbližšího souseda.

Vybrané metody jsou implementovány s využitím reálných dat naměřených senzorem a dále testovány na několika typech automobilů.

Klíčová slova: magnetický senzor, AMR, vzájemná korelace, neuronové sítě, metoda nejbližších sousedů

Školitel: Ing. David Novotný

Abstract

The aim of this work is to modify the existing prototype of a magnetic sensor that detects the occupancy of a parking space by a car.

The changes concern the hardware, but notably the detection algorithm. For this purpose, several detection methods are tried using cross-correlation, neural networks and the nearest neighbour method.

Selected methods are implemented using real data measured by a sensor and further tested on several types of cars.

Keywords: magnetic sensor, AMR, cross-correlation, neural networks, nearest neighbor method

Title translation: Advanced methods for car detection using magnetic field measurement

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tato práce se zabývá pokračováním vývoje parkovacího senzoru využívajícího k detekci změnu magnetického pole při příjezdu automobilu. Vývoj tohoto typu detektoru probíhá již několik let a jeho cílem je navrhnout takový detektor, který by měl menší rozměry než ty současně používané a nevyžadoval by zásah do podlahy (senzor by nemusel být umístěn pod automobilem, ale třeba před ním).

V průběhu vývoje bylo otestováno několik různých magnetických senzorů a nakonec se podařilo vyvinout detektor, který byl následně v několika kusech instalován na parkovišti FEL ČVUT pro další testování. Cílem této práce je upravit tento prototyp detektoru a vyzkoušet nové způsoby detekce.

Obrázek 1.1: Fotografie z měření

Původním záměrem bylo ověřit možnosti detekce s využitím neuronových sítí. Po pokusech s korelačním měřením rychlosti automobilu se přišlo s nápadem využít tuto metodu i pro detekci příjezdu/odjezdu. Další metody byly navržené v průběhu práce s ohledem na to, jak vypadala doposud naměřená data.

Změnami prošel i hardware detektorů. Byl vybrán jiný typ senzoru magnetického pole a po rozhodnutí o přidání ještě třetího senzoru ke stávající dvojici i nový mikrokontrolér. To pak vedlo k dalším úpravám obvodu.

Prototyp senzoru s návrhem obvodu i detekčním algoritmem je ve zkratce popsán v následující kapitole. Rozebrána je tam i teorie, která stála za vznikem detektoru. Třetí kapitola pak popisuje provedené změny v hardwaru.

Zbytek práce se zabývá samotnými metodami detekce. Ve čtvrté kapitole jsou rozebrány jednotlivé metody z teoretického hlediska a v páté je nastíněno, jak byly tyto metody implementovány pro potřeby ověření jejich využití.

V kapitole 6 jsou pak zpracovány výsledky z měření a porovnány jednotlivé metody z hlediska úspěšnosti detekce.

Kapitola 2
Výchozí projekt

Původně navržený detektor využíval pro rozhodnutí o přítomnosti automobilu buď magnetický gradient, nebo výpočet vzdálenosti ekvivalentního dipólu.

Pro praktickou realizaci bylo zvoleno použití gradiometru s malou bází a možností měřit homogenní složky magnetického pole. Senzory byly pospojovány po sběrnici RS485 a pro snadnější montáž se mělo propojovacím kabelem vést i napájení. Každý detektor pak měl dvě LED diody pro indikaci obsazenosti parkovacího místa (zelená = volno, červená = obsazeno).

Během vývoje se také testovalo využití magnetické signatury automobilu i pro měření rychlosti. Ta se dá určit využitím vzájemné korelace výstupních signálů z dvojice magnetometrů.

2.1 Úvod do problematiky

2.1.1 Magnetické pole automobilu

Magnetické pole automobilu není jednoduché pro namodelování, neboť různé typy mají různě rozložené a tvarované prvky, karoserie jsou z jiných materiálů apod. Rozdíl je také mezi auty s elektrickým a se spalovacím motorem. Elektromobily vykazují řádově vyšší signaturu v magnetickém poli.

Jako aproximace automobilu byl nakonec použit magnetický dipól. Pro ověření zvoleného modelu bylo provedeno několik měření (pomocí pole magnetometrů měřících magnetickou indukci nad, před a za automobilem) v různých směrech a vzdálenostech.

Výsledky tohoto měření ukázaly značné rozdíly momentů v závislosti na orientaci v zemském magnetickém poli i na tom, zda auto parkovalo popředu nebo pozadu. Přesto se (při simulacích) ukázalo, že může být gradient využit jako práh detekčního algoritmu pro všechny parkovací pozice. Pro rozhodnutí o přítomnosti auta se pak využíval výpočet vzdálenosti dipólu.

Při výpočtu vzdálenosti dipólového zdroje se vycházelo ze vztahů 2.1 a 2.2 (viz [2]):

$$B_z = \frac{\mu_0 m_z}{2\pi z^3}\ [T] \tag{2.1}$$

$$G_z = \frac{\partial B_z}{\partial z} = -3\frac{\mu_0 m_z}{2\pi z^4}\ [T/m], \tag{2.2}$$

kde $B_z\,[T]$ je homogenní složka,
$G_z\,[T/m]$ je gradient,
$z\,[m]$ je axiální vzdálenost,
$m_z\,[Am^2]$ je dipólový magnetický moment v ose z
a $\mu_0\,[Hm^{-1}]$ je permeabilita vakua.

Z vztahů 2.1 a 2.2 pak dostaneme vzdálenost dipólu jako:

$$z = -3\frac{B_z}{G_z}\,[m] \tag{2.3}$$

2.1.2 Astatizace gradiometru

Tento proces zlepšuje přesnost měření a odstraňuje citlivost gradientu na homogenní složky pole. Více podrobností viz [1] a [3].

Střední hodnota homogenního pole:

$$B_z = \frac{B_{z_1} - B_{z_2}}{2}$$

Gradient:

$$G = \frac{B_{z_1} - B_{z_2}}{\Delta z}$$

Astatizovaný gradient:

$$G_{ast} = G - [B_x \quad B_x^2 \quad B_y \quad B_y^2 \quad B_z \quad B_z^2 \quad 1] \cdot C$$

Vektor hodnot C dostaneme jako:

$$C = [B_x \quad B_x^2 \quad B_y \quad B_y^2 \quad B_z \quad B_z^2 \quad 1]^{-1} G,$$

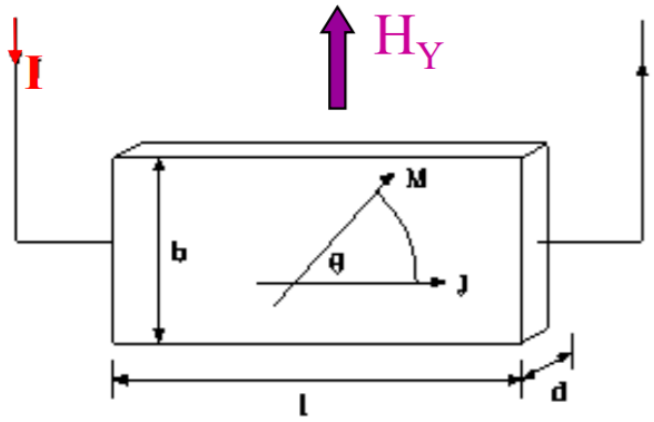
kde B_x, B_y a B_z jsou data zaznamenaná při otáčení senzoru v homogenním magnetickém poli kolem jednotlivých os.

2.1.3 Magnetické senzory

Pro účel projektu bylo potřeba vybrat senzor schopný měřit magnetické pole srovnatelné s velikostí zemského magnetického pole (tedy amplitudou okolo 50 μT). Dalším požadavkem byla schopnost měřit stejnosměrné složky signálu potřebné pro detekci přítomnosti a přinejmenším nízké frekvence pro korelačního měření rychlosti.

Všechny tyto parametry nejlépe splňují senzory typu AMR a fluxgate. V průběhu vývoje bylo testováno několik různých senzorů obou typů a nakonec byl zvolen AMR senzor HMC1021.

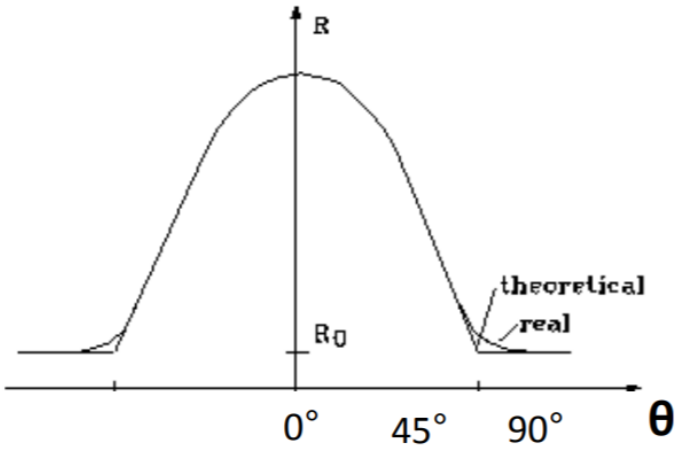
Anizotropní magnetorezistor (AMR)



Obrázek 2.1: Princip činnosti AMR senzoru - odpor závisí na úhlu mezi el. proudem a magnetizací v materiálu (převzato z [6])

Princip funkce AMR senzorů spočívá ve změnách elektrického odporu ve feromagnetiku. Ten závisí na geometrickém úhlu mezi procházejícím elektrickým proudem a směrem magnetizace v materiálu. Tato závislost vychází z kvantových jevů v elektronových obalech atomů feromagnetika, které jsou ovlivněny místním magnetickým polem, což má vliv na střední volnou dráhu elektronu v materiálu.

Normálně má magnetizace materiálu orientaci podél delší osy ($\varphi = 0$). Působením vnějšího pole se však natáčí a tím dojde i ke změně odporu.

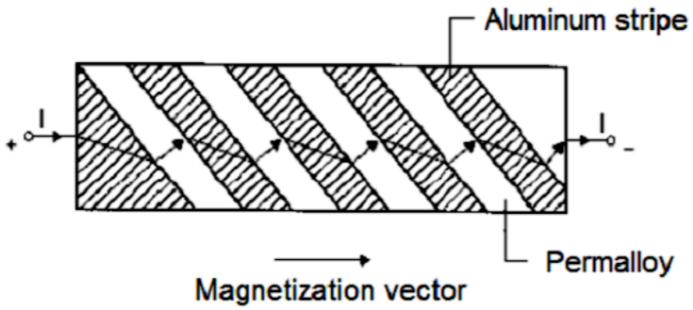


Obrázek 2.2: Unipolární odezva na vnější pole (převzato z [6])

Charakteristika odezvy je však v praxi obtížně použitelná (je stejná pro kladné i záporné pole). Proto je potřeba posunout pracovní bod tak, aby

v klidovém stavu byl na boku charakteristiky, tedy asi v polovině poklesu.

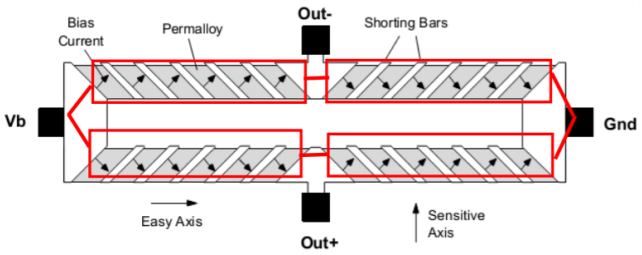
Toho lze nejsnáze dosáhnout technologií tzv. “barber poles”. To znamená, že jsou na feromagnetikum naneseny hliníkové proužky, které mají mnohem vyšší vodivost. Elektrický proud pak teče nejkratší cestou, tedy hliníkem a úseky feromagnetika překonává pod úhlem 45°. Tím je docíleno posunutí pracovního bodu.



Obrázek 2.3: Technika “barber poles”(převzato z [6])

Převodní charakteristiku může ovlivnit zmagnetování materiálu silným vnějším polem. Při vystavení silnému vnějšímu poli může dojít až k převrácení magnetizace materiálu a změně znaménka citlivosti senzoru.

Tomu lze zamezit periodickou obnovou výchozího stavu senzoru. To se provádí pomocí tzv. flipovací cívky. Krátký proudový impuls do cívky uvede senzor do výchozího stavu.



Obrázek 2.4: AMR senzory v plném můstku (převzato z [6])

V praxi jsou senzory zapojovány do plného můstku, což má mnoho výhod, například potlačení offsetu (nulové výstupní napětí v klidovém stavu), čtyřnásobná citlivost, lineární odezva na $\triangle R$ atd.

2.2 Praktická realizace senzoru

V této části je ve stručnosti popsán prototyp senzoru, jak z hlediska hardwaru, tak z hlediska detekce. Více informací o původním projektu viz [1].

2.2.1 Hardware

Jako mikrokontrolér byl vybrán typ STM32L432KB (parametry v tabulce 2.1), který splňoval požadavky na dostatečný výkon a velikost RAM a FLASH paměti, co nejmenší spotřebu a nízkou cenu.

velikost RAM	64 KB
velikost FLASH	128 KB
maximální takt	80 MHz
počet UART rozhraní	3
počet SPI rozhraní	2
počet I2C rozhraní	2
přesnost vnitřního oscilátoru	1 %

Tabulka 2.1: Vybrané parametry mikroprocesoru STM32L432KB

Pro digitalizaci signálu ze senzorů bylo třeba použít AD převodník. Vybrán byl 4-kanálový sigma-delta převodník MCP3912 s rozlišením 24 bitů, který umožňoval současné vzorkování na všech vstupech. Pro zlepšení funkce samotného AMR senzoru pak byl použit flipovací polomůstek s tranzistory.

Kvůli požadavku na rozsah napájecího napětí (8-50 V) bylo nutné použít spínaný měnič. Toho se dosáhlo obvodem TPS54061 s vestavěným spínacím prvkem. Za měnič se pak umístily LDO stabilizátory pro přesnější stabilizaci.

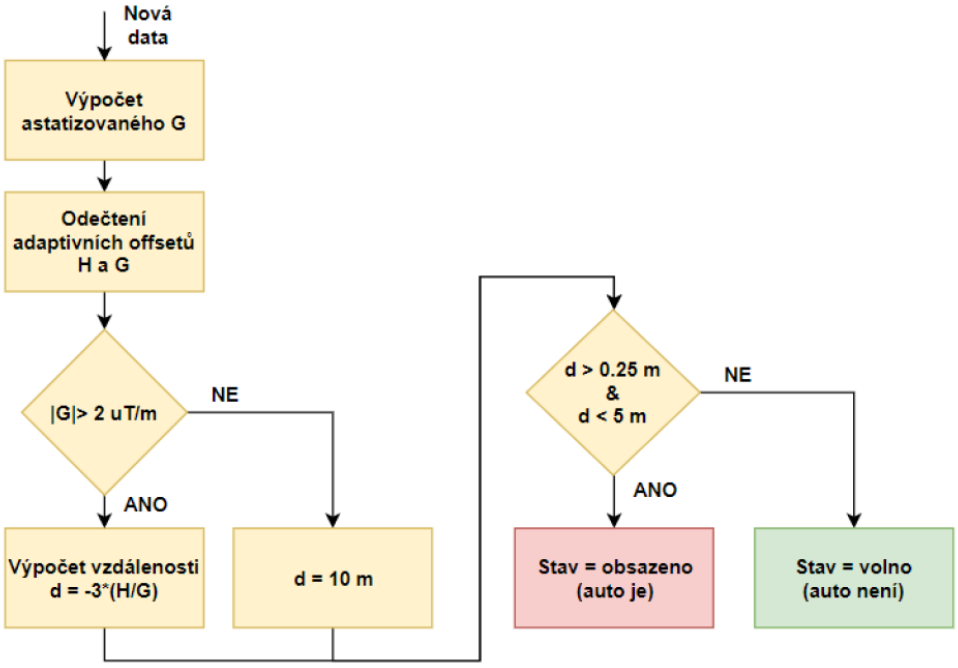
Napájení hlavních gradiometrů bylo řešeno zdrojem proudu, který pro správnou funkci vyžadoval minimální úbytek napětí cca 3 V. Byl tedy navržen diskrétní spínaný zdroj $\pm$ 12 V. Napájení zbylých senzorů (pro měření kolmých složek) pak bylo řešeno ze zdroje napětí (výstup LDO).

2.3 Detekční algoritmus

Pro detekci obsazenosti se využívalo vztahu pro výpočet vzdálenosti magnetického dipólového zdroje (viz vztah 2.3).

Nejprve se ověřovala změna gradientu. Pokud překročila hranici 2 μT/m, počítala se vzdálenost auta (modelovaného dipólem). Pro ostatní případy se nastavila vzdálenost na hodnotu 10 m.

Dál se pak testovalo, jestli vzdálenost leží v intervalu 0.25 až 5 m. Pokud ano, algoritmus vyhodnotil místo jako obsazené (auto je = log. 1), v opačném případě jako volné místo (auto není = log. 0).



Obrázek 2.5: Zjednodušený diagram původního detekčního algoritmu (převzato z [1])

Kapitola 3

Úpravy senzoru

Prototyp detektoru prošel několik změnami. Pořídil se jiný mikrokontrolér a také byl zvolen nový typ AMR senzoru, z čehož vyplynuly i další změny v hardwaru.

Četnými změnami prošel i detekční algoritmus. Bylo rozhodnuto o ověření detekčních možností několika vybraných metod, jako například vzájemné korelace signálů ze dvou AMR senzorů, neuronových sítí a dalších.

Více podrobností o provedených změnách dále v této kapitole.

3.1 Hardware

3.1.1 Prvotní úpravy

Prvotní úpravy detektoru spočívaly v tom, že se nahradily původní AMR senzory HMC1021. Nově použitý typ je tří-osý AMR senzor MMC5883. Má sice větší šum a horší nestabilitu nuly než HMC1021, ale pro naše účely je dostačující a navíc je levnější a plně digitální.

Ze začátku byly pro testovací měření upraveny dva prototypy. První detektor používal dva AMR senzory MMC5883 a na výstupu posílal hodnoty naměřené v jedné ose se vzorkovací frekvencí $f_s = 1000\,Hz$. Druhý měřil magnetické pole ve všech třech osách se vzorkovací frekvencí $f_s = 29.3\,Hz$.

3.1.2 Konečná verze

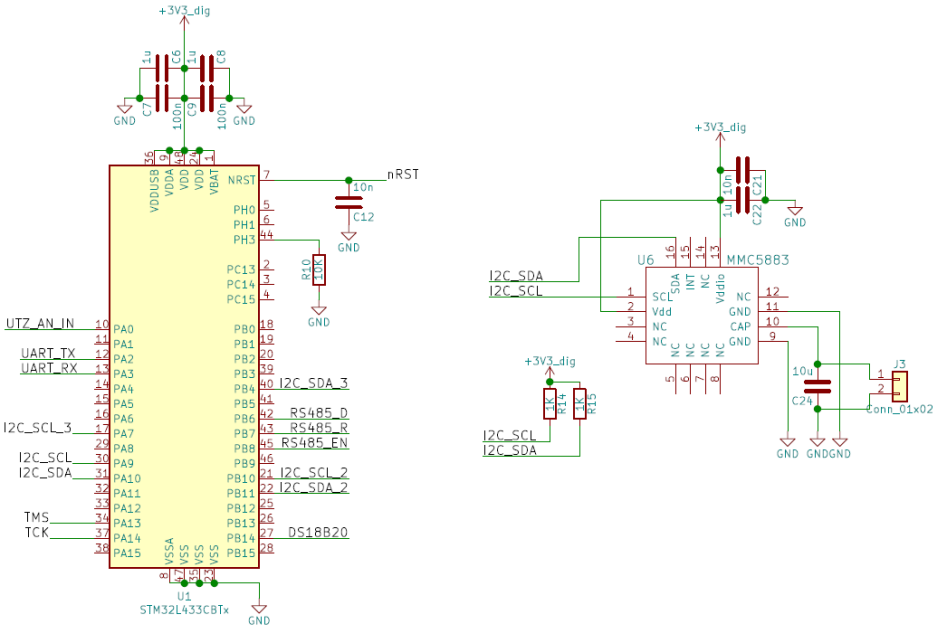
Z důvodu použití plně digitálního senzoru MMC5883, bylo možné odstranit analogové části obvodu. Přesněji AD převodník, jeden z LDO stabilizátorů, pomocné spínané zdroje a flipovací polomůstek.

Po prvotních testovacích měření bylo také rozhodnuto přidat do detektoru ještě třetí AMR senzor, tak že všechny tři senzory byly uspořádány do L.

Kvůli rozhodnutí přidat ještě třetí MMC5883, bylo nutné vybrat i nový mikrokontrolér. Původní mcu totiž měl jen dvě I2C sběrnice, a tudíž by připojení třetího senzoru bylo problematické.

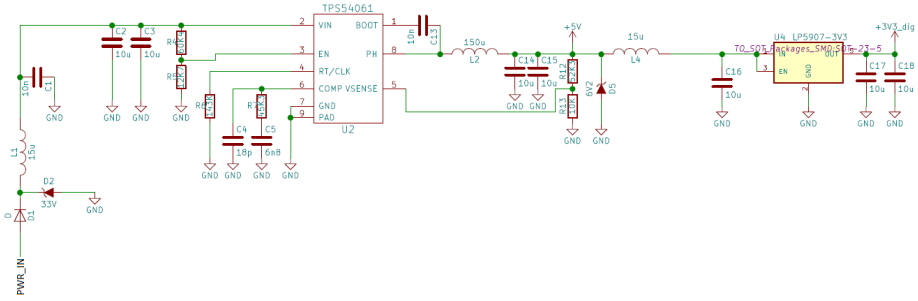
Pro podobnost s původním STM32L432KB byl vybrán mikrokontrolér STM32L433, který má podobné parametry, ale umožňuje připojení všech tří AMR senzorů přes tři I2C sběrnice. To se použitím plně digitálního MMC5883

výrazně zjednodušilo, neboť už nebylo potřeba flipovacího polomůstku ani AD převodníku.

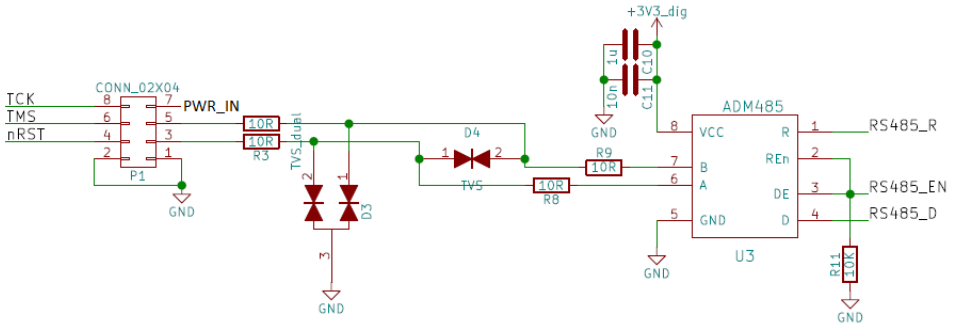


Obrázek 3.1: Mikrokontrolér a zapojení AMR senzorů

Zapojení hlavního spínaného zdroje se nezměnilo. Pouze místo dvou LDO stabilizátorů, následuje za měničem pouze jeden, protože druhý pro analogovou část už není díky novým plně digitálním senzorům třeba.

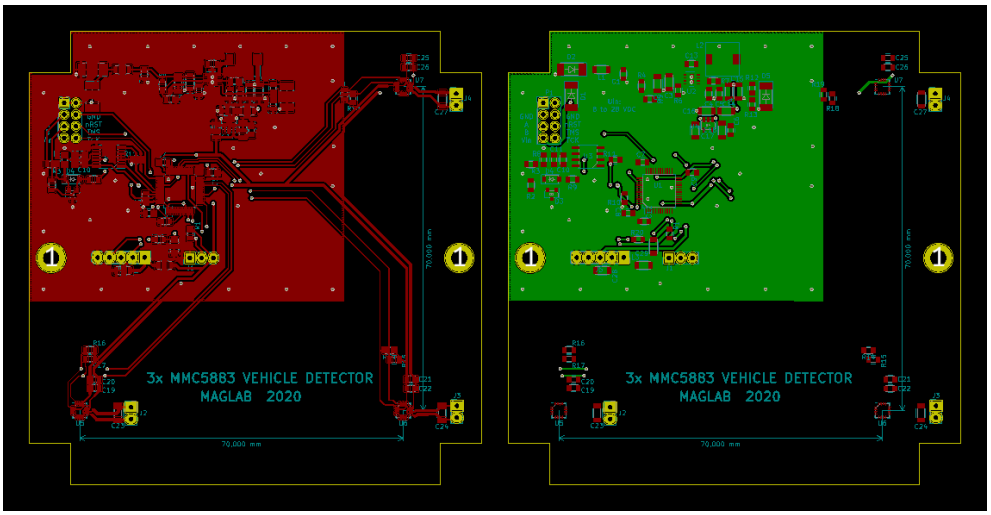


Obrázek 3.2: Hlavní spínaný zdroj a LDO stabilizátor



Obrázek 3.3: Budič sběrnice RS485

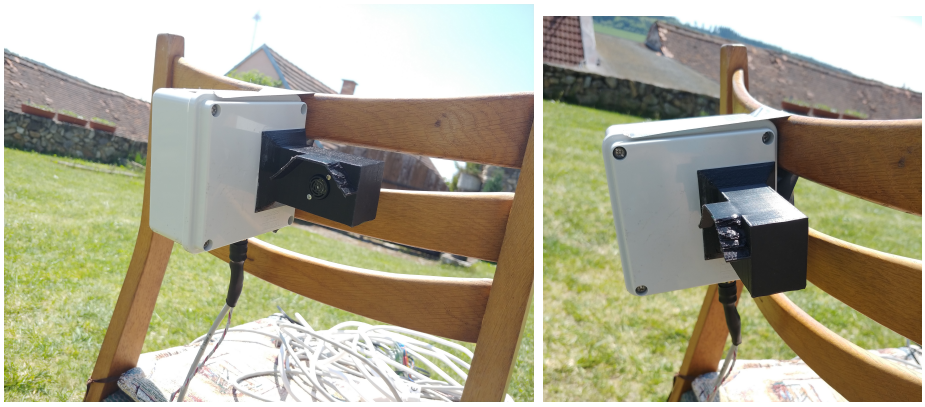
Obvod budiče RS485 sběrnice je na obrázku 3.3. Použitý typ je ADM485. Kromě datových vodičů Rx/Tx (resp. 485_R a 485_D) obsahuje ještě řízení toku (485_EN), které vypíná nebo zapíná vysílací (resp. přijímací) obvod.



Obrázek 3.4: DPS detektorů v KiCADu (strana TOP a BOTTOM)

Návrh plošného spoje s posledními úpravami je na obrázku 3.4. Pro potřeby měření je DPS umístěn do krabičky s jedním přívodním kabelem, který obsahuje napájení i sběrnici RS485, přes kterou detektor posílá naměřená data.

Dále je ke krabičce ze strany připevněn ultrazvukový senzor měřící vzdálenost automobilu. Tento senzor je tam umístěn jako reference pro zjištění reálné přítomnosti automobilu. Celý detektor je vidět na obrázku 3.5.



Obrázek 3.5: Detektor

3.2 Detekční algoritmus

Původním záměrem bylo využít pro detekci přítomnosti automobilu jednoduchou neuronovou síť, která by se učila na datech měřených v reálných podmínkách na různých typech vozů.

Při pokusech s měřením rychlosti automobilu pomocí vzájemné korelace signálů ze dvou AMR senzorů se objevil nápad zkusit využít tuto metodu i pro detekci obsazenosti místa, respektive příjezdu a odjezdu.

Třetí metoda vybraná k vyzkoušení detekce je metoda nejbližších sousedů. Tato metoda je velmi jednoduchá pro implementaci, ale vyžaduje dost paměti, což by mohla být jistá komplikace.

Poslední detekční metoda je binární klasifikace využívající lineární regresi, která rozděluje data do dvou tříd. V případě našeho detektoru jsou to třídy volno a obsazeno.

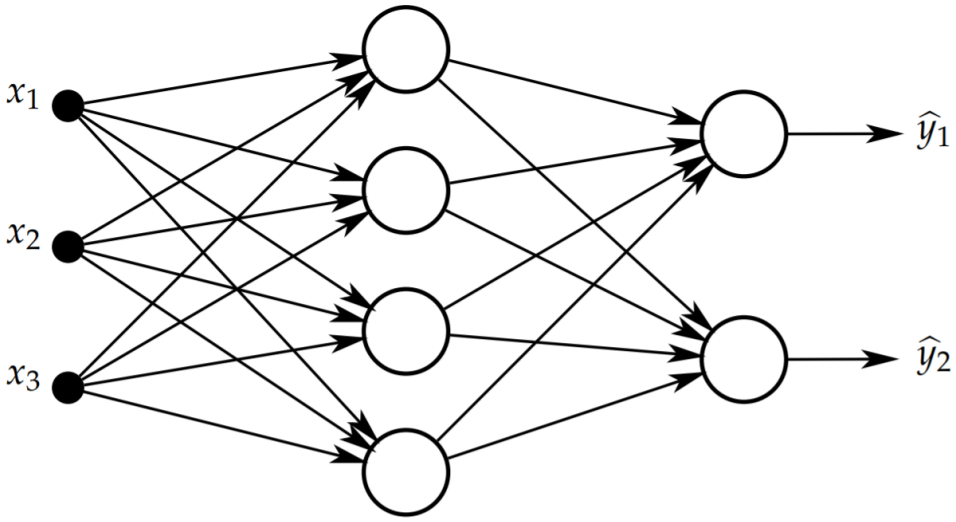
Všechny zvolené metody jsou podrobněji popsány v následující kapitole.

Kapitola 4

Teoretický rozbor metod použitých pro detekci

4.1 Neuronové sítě

Umělé neuronové sítě (NN) jsou inspirované strukturou lidské nervové soustavy. Základním prvkem takové sítě jsou neurony (perceptrony), ty jsou navzájem propojeny a tak si předávají signály. Přitom každý neuron má jeden nebo více váhovaných vstupů, které jsou buď výstupem jiného neuronu, nebo informacemi zvenčí. Výstup má každý neuron pouze jeden, ale ten může být předán hned několika dalším neuronům.



Obrázek 4.1: Neuronová síť s třemi vstupy, dvěma výstupy a jednou skrytou vrstvou

Pro řešení jednoduchých úloh lze vystačit i s jediným neuronem, ale většinou se jich používá více vzájemně propojených a uspořádaných do jednotlivých vrstev.

Vstupní vrstva přijme hodnoty zvenčí a předá je všem neuronům v tzv. skryté vrstvě. Výstupy těchto neuronů jsou pak vstupem pro neurony ve výstupní

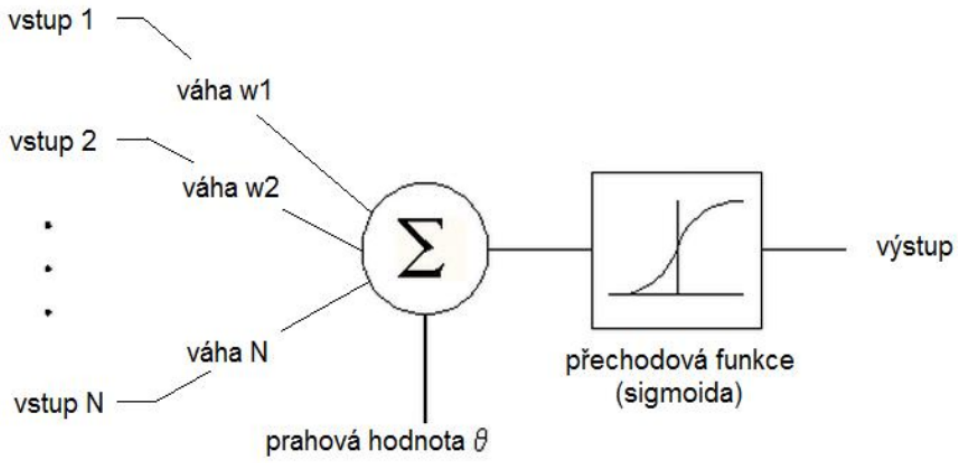
vrstvě, které kódují výstup celé neuronové sítě.

4.1.1 Princip činnosti

Nejprve neuron přijme vstupy a vynásobí je danými váhami. Následně tyto součiny sečte a porovná je se stanovenou prahovou hodnotou. Pokud je součet váhovaných vstupů větší než práh, tak ho transformuje předem danou přenosovou (někdy též aktivní) funkcí a pošle na výstup.

$$\hat{y} = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

x_i jsou vstupy a w_i jejich váhy,
θ je prahová hodnota,
f přenosová funkce
a y je výstup.



Obrázek 4.2: Princip neuronu

4.1.2 Učení neuronové sítě

Máme-li pevně danou strukturu sítě (počet vrstev a neuronů ve vrstvách a jejich vzájemné propojení), pak váhy a prahy, na jejichž hodnotě závisí, jak síť převede vstupy na výstupy, se stanovují v procesu učení.

Jsou dva způsoby, jak učit neuronové sítě – s učitelem a bez učitele. Pro první případ potřebujeme mít množinu vstupů a k nim patřících výstupů. Vstupy se pustí do sítě, výstup se porovná s požadovaným výstupem a následně se provede korekce (změna vah a prahů) tak, aby byl rozdíl mezi skutečným a požadovaným výstupem co nejmenší. Síť se tedy učí ze svých chyb.

Pokud je síť správně nastavená (ani moc velký, ani moc malý počet neuronů, dostatečný vzorek trénovacích dat...), je pak schopná řešit i další úlohy, než jen konkrétně ty, na které byla naučená.

V případě učení bez učitele stačí znát vstupy. Síť v takovém případě provádí tzv. shlukovou analýzu, tedy rozdělí zadané vstupy do konečného množství tříd. Ty pak lze snadněji podrobit další analýze.

4.1.3 Trénování NN na dané množině dat

Při trénování neuronové sítě s pomocí sady vstupů a k nim známým výstupů jde v podstatě o to najít vhodné hodnoty vah $(w_1, w_2, ...)$ tak, aby vypočítaný výstup ze sítě pro zadaná vstupní data odpovídal požadovanému výstupu.

K tomu je potřeba nejprve definovat ztrátovou funkci, která má být minimalizována (např. 4.1).

$$J(w) = \sum_{i=1}^{|T|} E\left(w, x^{(i)}, y^{(i)}\right) = \frac{1}{2} \sum_{i=1}^{|T|} \sum_{k=1}^{C} \left(y_{ik} - \hat{y}_{ik}\right)^2 \tag{4.1}$$

$$E(w, x, y) = \frac{1}{2} \sum_{k=1}^{C} \left(y_k - \hat{y}_k\right)^2$$

$|T|$ je velikost trénovací sady a C je počet výstupů NN.

úloha	vhodná ztrátová funkce
binární klasifikace	$J = -\sum_{i=1}^{|T|} \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - \hat{y}^{(i)}\right)\right]$
regrese	$\sum_{i=1}^{|T|} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$
regrese s více výstupy	$\sum_{i=1}^{|T|} \sum_{k=1}^{C} \left(y_k^{(i)} - \hat{y}_k^{(i)}\right)^2$

Tabulka 4.1: Příklady ztrátové funkce

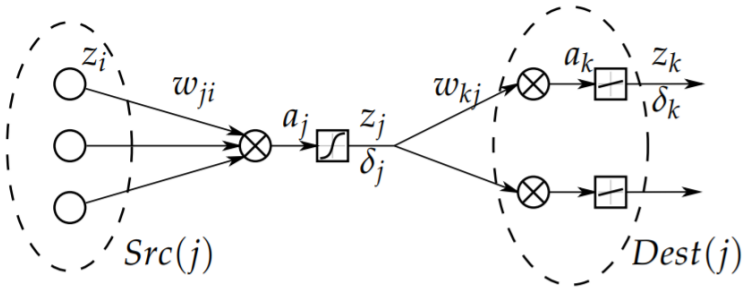
Dále se počítá gradient ztrátové funkce[1] $\nabla E(w) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_W}\right)$ a následně se aktualizují váhy podle vztahu 4.2.

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}, i = 1, 2, ..., W \tag{4.2}$$

Pro výpočet jednotlivých derivací se používá algoritmus backpropagation (z angl. backward propagation of errors, tj. zpětné šíření chyb).

[1] Můžeme uvažovat pouze $\frac{\partial E}{\partial w_i}$, protože $\frac{\partial J}{\partial w_i} = \sum_n \frac{\partial}{\partial w_i} E\left(w, x^{(n)}, y^{(n)}\right)$.

Backpropagation



Obrázek 4.3: Algoritmus backpropagation - příklad

Platí:

$$a_j = \sum_{i \in Src(j)} w_{ji} z_i, \tag{4.3}$$

$$z_j = g(a_j), \tag{4.4}$$

kde g je takzvaná aktivační funkce, která moduluje výstupní signál neuronu. Příklady různých aktivačních funkcí jsou v tabulce 4.2.

lineární	$g(a) = a$
jednotkový skok	$g(a) = \begin{cases} 0, & a < 0 \\ 1, & a \geq 0 \end{cases}$
sigmoida	$g(a) = \sigma(a) = \frac{1}{1+e^{-a}}$
hyperbolický tangens	$g(a) = \tanh(a) = 2\sigma(a) - 1$
usměrněná lineární (ReLU)	$g(a) = \max(0, a) = \begin{cases} 0, & a < 0 \\ a, & a \geq 0 \end{cases}$

Tabulka 4.2: Aktivační funkce neuronu

Z obr. 4.3 lze vyvodit, že E závisí na w_{ji} skrz a_j, tedy:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

Ze vztahu 4.3 dostaneme derivaci jako:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

Výsledná parciální derivace tedy je:

$$\frac{\partial E}{\partial w_{ji}} = \delta_j z_i,$$

kde $\delta_j = \frac{\partial E}{\partial a_j}$ nazýváme chybou.

E závisí na a_k skrz výstup $\hat{y}_k = g(a_k)$:

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial a_k} = \frac{\partial E}{\partial \hat{y}_k} g'(a_k)$$

Jak již bylo zmíněno výše, E závisí na w_{ji} skrz a_j. Na a_j je pak závislý přes a_k následovně:

$$\delta_j = \frac{\partial E}{\partial a_j} = \sum_{k \in Dest(j)} \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_{k \in Dest(j)} \delta_k \frac{\partial a_k}{\partial a_j}$$

Pro a_k platí:

$$a_k = \sum_{j \in Src(k)} w_{kj} z_j = \sum_{j \in Src(k)} w_{kj} g(a_j)$$

A tedy derivace je:

$$\frac{\partial a_k}{\partial a_j} = w_{kj} g'(a_j)$$

S využitím předchozího vztahu dostaneme δ_j jako:

$$\delta_j = g'(a_j) \sum_{k \in Dest(j)} \delta_k w_{kj}$$

Chyba δ_k je tak distribuována do δ_j ve spodní vrstvě podle váhy w_{kj} (která odpovídá rychlosti růstu lineární kombinace a_k) a podle velikosti $g'(a_j)$ (což je rychlost růstu aktivační funkce).

1 **begin**

2 Perform a forward pass for observation x. This will result in values of all a_j and z_j for the vector x.

3 Evaluate the error δ_k for the output layer:

$$\delta_k = \frac{\partial E}{\partial \hat{y}_k} g'(a_k)$$

4 Propagate the errors δ_k back to get all the remaining δ_j:

$$\delta_j = g'(a_j) \sum_{k \in Dest(j)} \delta_k w_{kj}$$

5 Evaluate all the derivatives to get the whole gradient:

$$\frac{\partial E}{\partial w_{ji}} = \delta_j z_i$$

Ukázka kódu 4.1: Error Backpropagation

4.2 Vzájemná korelace

Tato metoda je založena na vyhodnocení vzájemné korelace signálů ze dvou stejných senzorů měřících jeden objekt, ale s jistým zpožděním způsobeným vzdáleností mezi umístěními obou senzorů.

Jak již bylo řečeno výše, výstupní signály z obou senzorů jsou vůči sobě časově posunuté. Kvůli šumu a vlivu jiných rušivých signálů ale nejsou totožné, tedy nelze jejich vzájemné zpoždění určit přesně. Můžeme ale zjistit nejpravděpodobnější hodnotu tohoto zpoždění, a to například využitím korelační funkce.

Vzájemná korelace dvou signálů je definována vztahem:

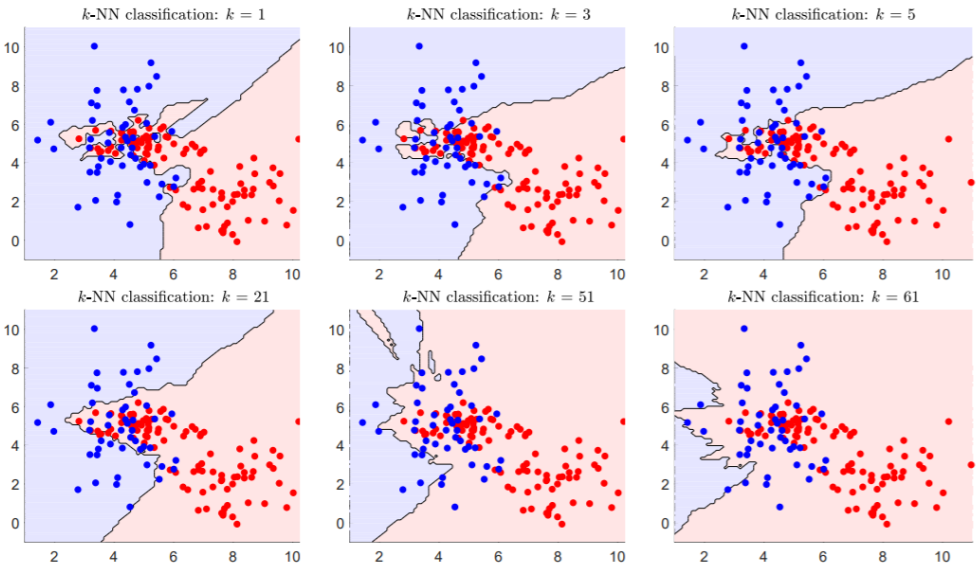
$$K(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f_1(t) f_2(t+\tau) dt,$$

kde $f_1(t)$, $f_2(t)$ jsou časové průběhy signálů,
T je doba trvání signálů (doba měření),
$K(\tau)$ je korelační funkce pro $\tau \in (-T, T)$.

Vzájemně korelační funkce nabývá maxima právě pro τ rovno posunu obou signálů. Nalezením argumentu maximální vzájemné korelace tedy získáme hledaný časový posun.

4.3 Algoritmus *k* nejbližších sousedů

Metoda k nejbližších sousedů, dále jen k-NN (z angl. k-nearest neighbors), je jednoduchá, neparametrická metoda použitelná jak pro klasifikaci, tak pro regresi.



Obrázek 4.4: k-NN klasifikace - příklad

Učení s použitím této metody probíhá velice jednoduše, spočívá pouze v tom, že si musí pamatovat celou sadu trénovacích dat T.

Klasifikace probíhá následovně. Nejprve se pro nový datový bod x určí množina $N_k(x)$ k nejbližších sousedů x v T pomocí měření vzdálenosti mezi jednotlivými datovými body. Dále pak určete předpokládanou třídu $\hat{y} = h(x)$ jako většinový hlas mezi nejbližšími sousedy:

$$\hat{y} = h(x) = \arg\max_{y} \sum_{(x',y')\in N_k(x)} I(y' = y),$$

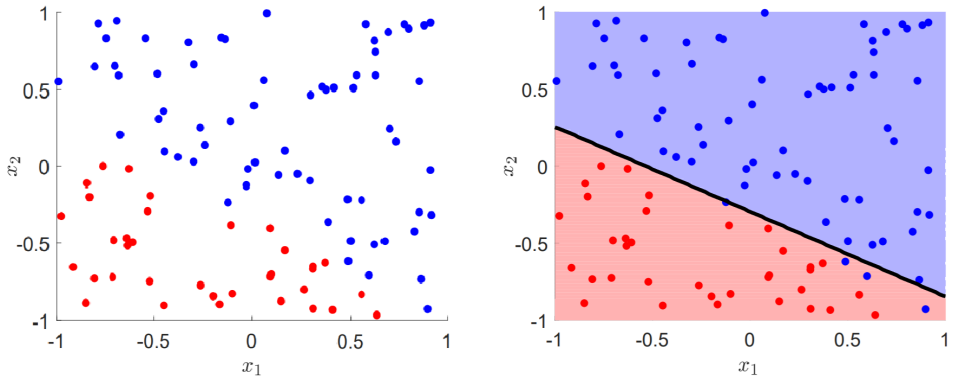
kde $I(P)$ vrací 1, pokud je P *true*, jinak 0.

4.4 Binární klasifikační úloha - lineární klasifikace

Je dána sada trénovací dat $T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(|T|)}, y^{(|T|)})\}$. Každý příklad v této sadě je popsán vektorem charakteristik $x = (x_1, x_2, \ldots, x_D)$ a označen štítkem třídy, do které patří ($y \in \{+1, -1\}$).

Diskriminační funkce $f(x)$ je taková funkce, která umožňuje rozhodnout, do jaké třídy bude klasifikován příklad x. V případě binární klasifikační úlohy jde o dvě třídy a pro diskriminační funkci platí:

$$\left.\begin{array}{l} f(x) > 0 \iff \hat{y} = +1 \\ f(x) < 0 \iff \hat{y} = -1 \end{array}\right\} \quad \hat{y} = \operatorname{sign}(f(x))$$



Obrázek 4.5: lineární klasifikace - příklad

4.4.1 Učení

Jedna z možností, jak získat diskriminační funkci ze sady trénovacích dat je najít vhodný lineární regresivní model. Nejprve potřebujeme vhodnou ohodnocovací funkci J, kterou pak budeme následně minimalizovat vzhledem k w.

$$J(w, T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - f(w, x^{(i)})\right)^2$$

Druhé a lepší řešení je najít lineární diskriminační funkci, která minimalizuje počet chyb, tedy:

$$J(w,T) = \frac{1}{|T|}\sum_{i=1}^{|T|} \mathbb{I}\left(y^{(i)} \neq \hat{y}^{(i)}\right)^2,$$

kde $\mathbb{I}$ je indikační funkce, která vrací 1, pokud je její argument *true*, jinak 0.

4.4.2 Lineární regrese

Lineární regrese je regresní model, který předpokládá lineární vztah mezi vstupy a výstupy:

$$\hat{y} = h(x) = w_0 + w_1 x + \cdots + w_D x_D = w_0 + x w^T,$$

kde $\hat{y}$ je predikce modelu (odhad reálné hodnoty),
$h(x)$ je lineární model,
a $w = (w_0, \ldots, w_D)$ je vektor koeficientů lineární funkce.

Přidáme-li 1 na začátek x tak, že dostaneme $x = (1, x_1, \ldots, x_D)$, můžeme pak přepsat lineární model do jednodušší formy:

$$\hat{y} = h(x) = w_0 \cdot 1 + w_1 x + \cdots + w_D x_D = x w^T,$$

Sadu trénovacích dat můžeme dále uspořádat do matice, takže dostaneme tzv. maticovou notaci:

$$\begin{bmatrix} 1 & x^{(1)} \\ \vdots & \vdots \\ 1 & x^{(T)} \end{bmatrix} \qquad \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(T)} \end{bmatrix}$$

Každý model má dva operační módy:

učení (trénování) uzpůsobení modelu datům (hledání parametrů w)

$$w = \underset{w}{\operatorname{argmin}}\, J(w,T)$$

aplikace (testování) aplikování modelu h na data

$$\hat{y} = h(x,w) = X w^T$$

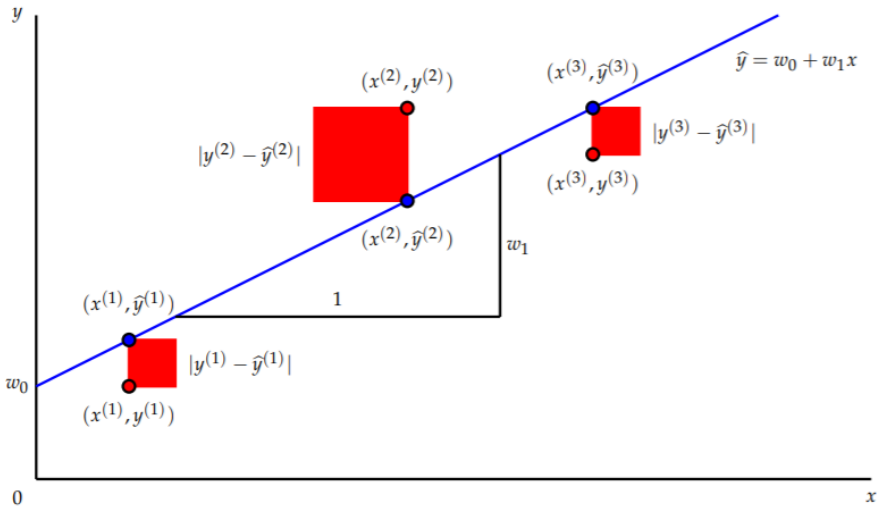
Metody pro hledání vhodného modelu:

1. **numerická optimalizace** $J(w,T)$ - např. metoda nejmenších čtverců

$$J_{MSE}(w,T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

2. **normální rovnice**

$$w = (X^T X)^{-1} X^T y$$



Obrázek 4.6: Nastavování modelu pomocí metody nejmenších čtverců (převzato z [13])

Kapitola 5
Implementace metod pro detekci

Pro vývoj a testování detekčních algoritmů bylo každou verzí senzoru naměřeno několik sad dat s využitím několika modelů automobilů. Takto získaná data byla dál zpracovávána v Matlabu.

Samotné detekční algoritmy byly psané v jazyce C, ve kterém se programuje i mikroprocesor. Výjimkou byla detekce s použitím korelace, ta je řešena v Matlabu, který má pro výpočet vzájemné korelace předdefinovanou funkci.

V Matlabu jsou také vykreslovány výsledky všech detekčních algoritmů, porovnávají se s reálnou přítomností automobilu a počítá se tam i jejich úspěšnost.

5.1 Neuronové sítě

Pro implementaci detekčního algoritmu s neuronovou sítí v jazyce C jsem využila volně dostupnou knihovnu Genann. Její výhodou je jednoduchost a rychlost, protože poskytuje pouze nezbytné funkce pro trénování a testování neuronových sítí.

Ukázka použití knihovny je v kódu níže. Funkce `gennan_init()` slouží k vytvoření nové neuronové sítě. Jejími argumenty jsou počet vstupů, skrytých vrstev, počet neuronů v každé vrstvě a počet výstupů. Funkce `gennan_train()` pak slouží k naučení NN na zadaná data a `gennan_run()` k vyhodnocování.

Dále je možné již vytvořenou neuronovou síť uložit do textového souboru s využitím funkce `gennan_write()` nebo naopak NN ze souboru načíst funkcí `gennan_read()`.

```
1 // Input and expected out data for the XOR function.
2 const double input[4][2] = {{0, 0}, {0, 1}, {1, 0},
     {1, 1}};
3 const double output[4] = {0, 1, 1, 0};
4
5 // New network with 2 inputs, 1 hidden layer of 2
     neurons, and 1 output.
6 genann *ann = genann_init(2, 1, 2, 1);
7
```

```
8  // Train on the four labeled data points many times.
9  int i;
10 for (i = 0; i < 300; ++i) {
11     genann_train(ann, input[0], output + 0, 3);
12     genann_train(ann, input[1], output + 1, 3);
13     genann_train(ann, input[2], output + 2, 3);
14     genann_train(ann, input[3], output + 3, 3);
15 }
16
17 // Run the network and see what it predicts.
18 printf("Output for [%f, %f] is %f.\n", input[0][0],
      input[0][1], *genann_run(ann, input[0]));
19 printf("Output for [%f, %f] is %f.\n", input[1][0],
      input[1][1], *genann_run(ann, input[1]));
20 printf("Output for [%f, %f] is %f.\n", input[2][0],
      input[2][1], *genann_run(ann, input[2]));
21 printf("Output for [%f, %f] is %f.\n", input[3][0],
      input[3][1], *genann_run(ann, input[3]));
22
23 genann_free(ann);
```

Ukázka kódu 5.1: Učení neuronové sítě na funkci XOR s použitím backpropagation

Samotný program řešící detekci s využitím neuronových sítí se spouští se třemi parametry. Prvním se vybírá použitá NN, druhý přepíná mezi učením a testováním a třetí je název testového souboru s uloženými daty.

1. parametr:

   load_nn pro načtení již vytvořené NN ze souboru

   new_nn pro vytvoření nové NN

2. parametr:

   test pro spuštění učení NN

   train pro spuštění testování NN

   train_test pro spuštění učení i testování

Posledním parametrem je název souboru obsahující data, která chceme využít pro učení NN nebo testování její funkčnosti. Aby program fungoval správně, musí být daný soubor ve složce `data`. Výsledky detekce jsou pak ukládány do složky `results`.

Porovnání výsledků detekčního algoritmu s reálnou přítomností následně řeší skript `testing.m` v Matlabu. Tento skript také počítá úspěšnost detekce.

5.2 Vzájemná korelace

V tomto případě byla naměřená data ze senzoru zpracovávána v programu Matlab. Za tímto účelem jsem vytvořila několik skriptů a funkcí, které řešily všechno od načtení dat ze souboru, následné úpravy dat, korelačního algoritmu a testování úspěšnosti detekce.

Hlavní skript spouští ostatní funkce. Nejprve funkci `raw_data()`, jejímž úkolem je načíst hodnoty z textového souboru a upravit je do podoby vhodné pro použití vzájemné korelace (přepočítat je do správných jednotek, odečíst offsety, odfiltrovat šum apod.). Výstupem této funkce jsou naměřené hodnoty z obou senzorů a reálná obsazenost parkovacího místa (jako reference pro určení přesnosti detekčního algoritmu).

Následně je zavolána funkce `corr()`, která obsahuje samotný detekční algoritmus. Jejím cílem je detekovat příjezd/odjezd automobilu podle toho, jak jsou navzájem signály ze senzorů časově posunuty. Více o její funkci později.

Poslední funkce je `corr_test()`. Jejími argumenty jsou vektory s reálnou a odhadovanou přítomností a na výstupu vrací dvě různé přesnosti. Výpočet první spočívá v prostém porovnání obou vektorů, tedy jak moc jsou si odhadovaná a reálná přítomnost podobné. Druhé číslo značí také podobnost, tentokrát se ale vektory porovnávají pouze v místech, kdy je alespoň jeden z nich v log. 1 (tedy obsazeno). Tuto druhou hodnotu udávám z toho důvodu, že v prvním případě vychází přesnost okolo 50 %, i když algoritmus nedetekuje žádný příjezd.

Mezi skripty řešícími korelační detekci jsou ještě další dva pro výpočet gradientů. Funkce `astatization_MMC()` vrací vektor konstant C a volá se ze skriptu `grad_astat`, který načte data ze všech měření a jejich vypočtené gradienty pak uloží pro další použití.

5.2.1 Korelační funkce v Matlabu

Matlab má pro výpočet vzájemné korelace předdefinovanou funkci `xcorr`, která lze použít následovně.

`r = xcorr (x, y)` vrací vzájemnou korelaci dvou diskrétních signálů. Pokud mají x a y různé délky, připojí funkce nuly na konec kratšího vektoru, takže má stejnou délku N.

`[r, lags] = xcorr (x, y)` vrací také vektor se zpožděními, ve kterých se korelace počítá.

```
1 [r, lags] = xcorr(data1, data2);
2 [~, I] = max(abs(r));
3 lagDiff = lags(I);
4 timeDiff = lagDiff/fs;
```

Ukázka kódu 5.2: Výpočet časového zpoždění dvou signálů v Matlabu

Funkce `xcorr()` vrací jednotlivá zpoždění (vektor `lags`), pro která se počítá vzájemná korelace obou signálů, a vektor hodnot korelační funkce (`r`) pro tato zpoždění.

V dalším kroku se zjišťuje maximum korelační funkce pomocí matlabovské funkce `max()`, která vrací maximální hodnotu a index pole, na kterém se tato hodnota nachází. Nicméně, pro další výpočty je potřeba jenom index, není tedy nutné načítat hodnotu maxima do proměnné (Matlab pro tento případ využívá znak ∼).

Časové zpoždění pak lze dopočítat z hodnoty zpoždění na daném indexu (tedy odpovídajícímu maximu korelační funkce) a vzorkovací frekvence.

5.2.2 Detekce příjezdu/odjezdu

Detektor je vůči přijíždějícímu automobilu orientován tak, že jeden z AMR senzorů se nachází ve větší vzdálenosti od auta než ostatní, a tudíž registruje změnu magnetického pole při příjezdu auta s jistým zpožděním. Naopak odjezd auta detekuje vzdálenější senzor dříve, tedy jeho výstup předstihuje signál z bližšího AMR senzoru.

Pomocí vzájemné korelace jsme schopni zjistit, který senzor začal detekovat změnu magnetického pole první, a tedy určit, zda auta na místo právě přijíždí nebo odjíždí.

K získání časového posunu obou signálů se využívá výše zmíněná funkce `xcorr`. Jejími argumenty ale nemohou být jednoduše výstupní signály z obou senzorů, protože ty jsou vůči sobě posunuty při každém příjezdu/odjezdu jinak. Z toho důvodu se vyhodnocuje vzájemnou korelací z obou signálů vždy jen určitě časové okno.

5.3 Algoritmus *k* nejbližších sousedů

Implementace tohoto algoritmu je celkem jednoduchá. Učení spočívá pouze v zapamatování si trénovacích dat a vyhodnocení probíhá tak, že se vstupní hodnoty porovnají se všemi naučenými a hledají se takové, které jsou nejpodobnější. Výstup náležející nalezenému nejbližšímu sousedovi se pak přisoudí současnému vstupu.

Pro porovnání podobnosti dat je využita Euklidovská metrika, která vyjadřuje délku úsečky mezi dvěma body, daná vztahem:

$$m_e(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Algoritmus je implementován pro $k = 1$, tedy pro každé vstupní hodnoty hledá jednoho nejbližšího souseda a podle něj pak rozhoduje o výstupu.

5.4 Lineární regrese a klasifikace

Protože jsem k vývoji detekčních algoritmů hodně využívala Matlab, zvolila jsem pro hledání vhodného lineárního regresivního modelu druhou metodu, tedy normální rovnici:

$$w = (X^T X)^{-1} X^T y$$

Samotná detekce už pak byla jednoduchá. Stačilo vynásobit data ze senzoru nalezeným w, a pokud výsledek vyšel záporný, detekoval algoritmus volno. Pokud vyšel kladný, bylo to obsazeno.

Zobrazení výsledků a výpočet přesnosti detekce se řešilo podobně jako u předchozích metod.

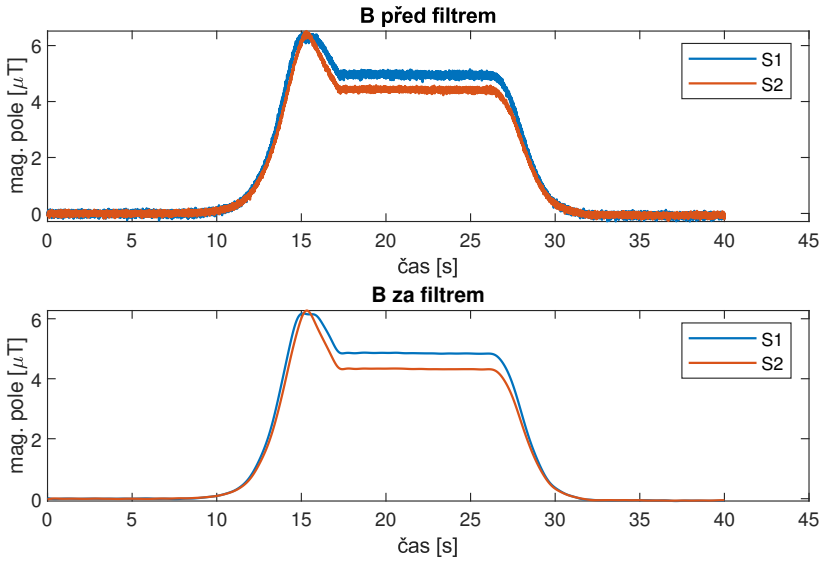
Kapitola 6

Dosažené výsledky

6.1 Pokusy s prvním senzorem

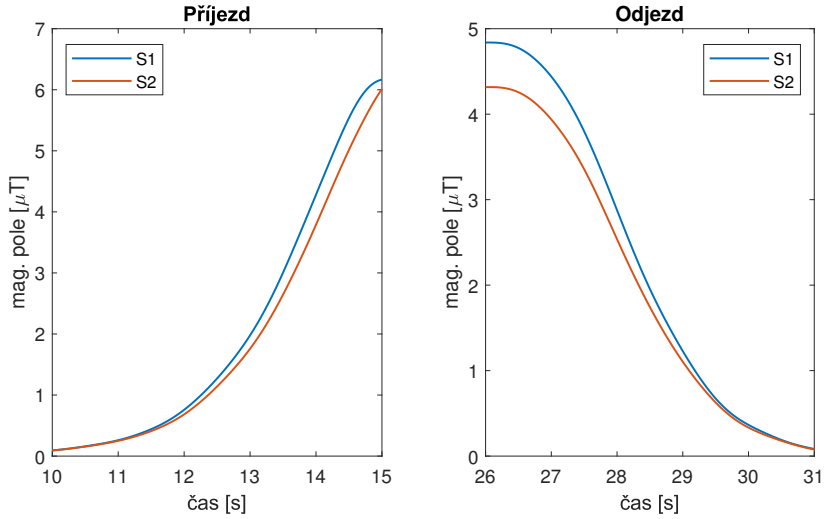
První změny detektoru spočívaly pouze v tom, že byly původní AMR senzory nahrazeny jiným typem. Nová verze detektoru tedy obsahovala dva AMR senzory MMC5883 a na výstupu se posílaly hodnoty magnetického pole v jedné ose naměřené oběma senzory s frekvencí 1000 Hz.

Napájení detektoru se řešilo baterií nebo přes adaptér ze sítě. Komunikace s počítačem probíhala po sériové lince a naměřené hodnoty se zapisovaly do textového souboru. K dispozici jsem měla dva vozy - Škodu Octavia a Škodu Rapid.



Obrázek 6.1: Ukázka vstupních dat používaných pro korelaci (Škoda Rapid)

Na obrázku 6.1 je vidět průběh jednoho měření, kdy vozidlo přijelo na místo, setrvalo tam několik desítek vteřin a následně opět odjelo. Na obrázku 6.2 lze pak vidět detailněji příjezd a odjezd automobilu.



Obrázek 6.2: Příjezd a odjezd automobilu (Škoda Rapid)

Při testování se ukázalo, že výsledky této metody jsou značně závislé na volbě délky okna a velikosti kroku, ale také na nastavení filtru. Rozdíly v úspěšnosti byly velké, v určitých případech i o 20 - 30 % jinde.

Všechna měření prošla testováním s různými kombinacemi výše zmíněných parametrů, přičemž nejlépe vyšla detekce s použitím hodnot v tabulce 6.1. Úspěšnost detekčního algoritmu s tímto nastavením byla 78.7 % podle prvního kritéria hodnocení, respektive 73.3 % podle druhého.

délka časového okna	200 vzorků (0.2 s)
velikost kroku	100 vzorků (0.1 s)

Tabulka 6.1: Parametry korelačního algoritmu - první verze detektoru

Během vývoje detekčního algoritmu jsem narazila na zásadní problém. Při procházení všech naměřených dat, jsem zjistila, že v několika případech jsou signály vůči sobě posunuté přesně naopak. Tedy místo aby se jako první měnil výstup z AMR senzoru označeného jako S1 (ten bližší k automobilu), reaguje na příjezd auta dříve senzor S2.

Tuto skutečnost se mi nepovedlo nijak objasnit. Jistým vysvětlením by mohlo být, kdyby byl detektor při měření omylem otočen, takže by blíže k vozidlu byl senzor S2. Na krabičce, ve které je detektor umístěn, je však dobře poznat, jestli je vůči přijíždějícímu autu orientován správně, takže je značně nepravděpodobné, že by došlo k takovému omylu.

Poněvadž se nepovedlo rozumně vysvětlit, proč jsou v některých případech signály přehozené, rozhodla jsem se vyřešit tento problém při programování detekčního algoritmu. Protože časový posun obou signálů není jediný rozdíl (různá je i amplituda signálu), způsobený odlišnou vzdálenosti AMR senzoru od přijíždějící auta, nebylo obtížné zjistit, který signál by měl být při příjezdu první a naopak při odjezdu opožděný.

Při spuštění detekčního algoritmu se tedy nejprve porovnají oba signály

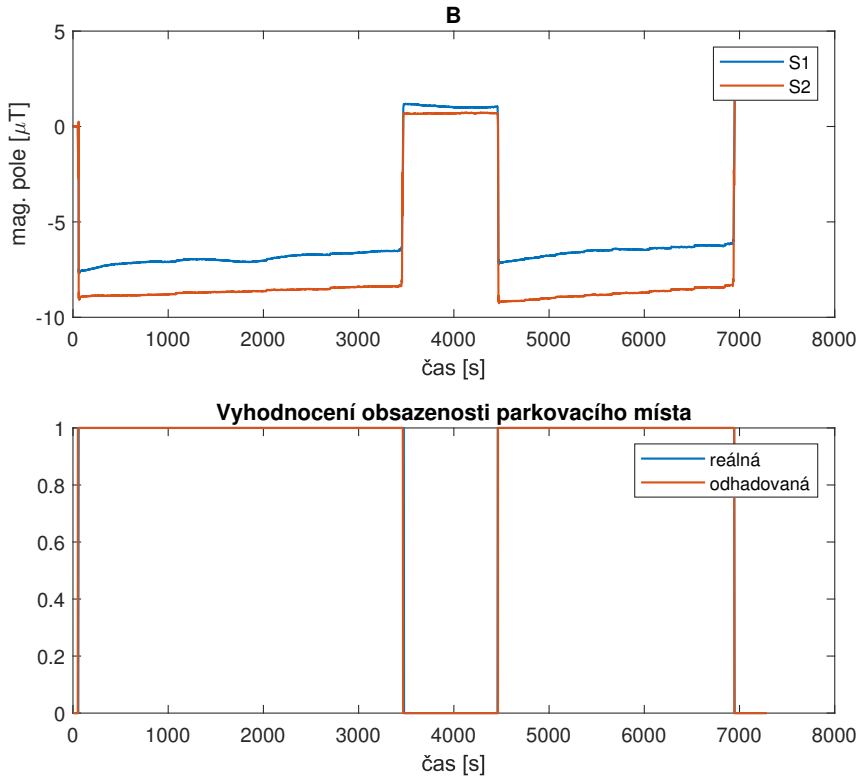
a ten s větší střední hodnotou se bere jako bližší senzor a ten s menší jako vzdálenější. Takto pak může detekce s využitím vzájemné korelace dobře fungovat i pro případy, kdy jsou měřené hodnoty přehozené.

Přestože výše popsaný problém byl vyřešen právě díky faktu, že i velikost amplitudy měřeného signálu je závislá na vzdálenosti senzoru od automobilu (čím blíž, tím je větší), pro normální funkci vzájemné korelace to byla spíše překážka.

Z toho důvodu jsem provedla několik testování funkce `xcorr()` se dvěma signály, jejichž amplitudy jsem násobila různými konstantami nebo k nim přičítala offsety. S takto upravovanými daty jsem pak zkoušela funkci `xcorr()`, abych zjistila, jaké má výsledky, pokud se signály částečně liší.

Tyto pokusy dopadly překvapivě dobře. Ukázalo se, že funkce `xcorr()` je značně odolná proti změnám amplitudy, pokud si data jinak zachovávají svoji podobnost. Přesto nevylučuji, že i tato skutečnost mohla snižovat výslednou úspěšnost detekce.

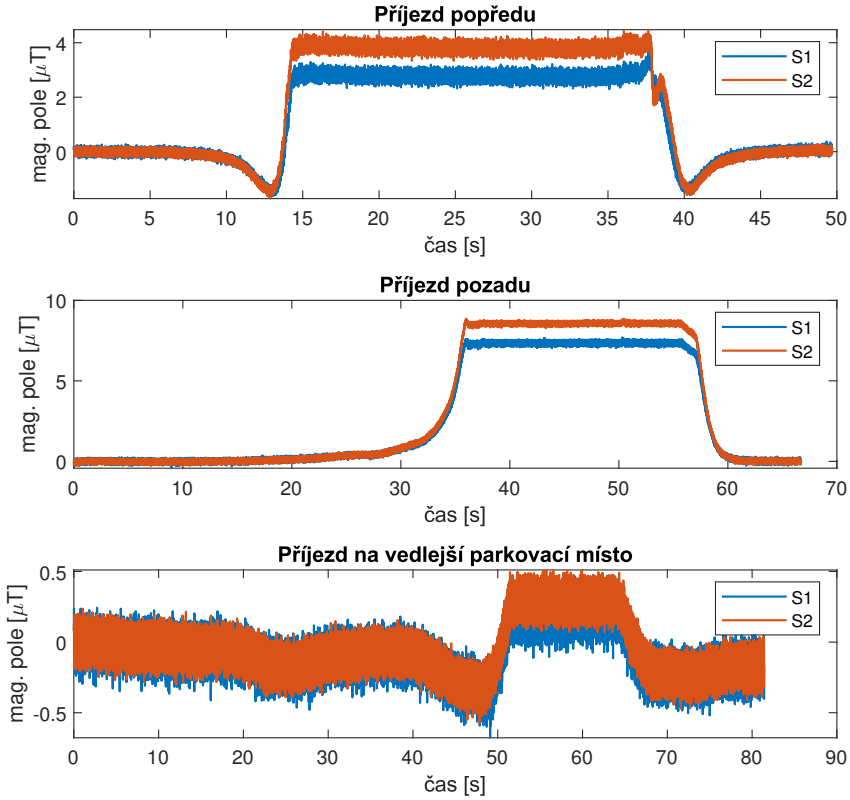
Příklad funkce detekčního algoritmu s vyřešením problému přehozených signálů je na obrázku 6.3. Jde o měření s vozem Škoda Octavia v delším časovém úseku (pár hodin). Tento konkrétní případ detekce proběhl s úspěšností 99.7 %, respektive 99.6 %.



Obrázek 6.3: Ukázka funkce korelačního algoritmu na delším měření (Škoda Octavia)

Výše zmíněný příklad také naznačuje, že v reálných podmínkách (tedy na skutečném parkovišti, kde stojí auta až několik hodin) by tato detekční metoda mohla mít lepší výsledky. Bohužel v domácích podmínkách bylo obtížné provádět více takových dlouhých měření.

Během testování detekčního algoritmu jsem zjišťovala i vliv orientace auta vůči senzoru, tedy rozdíl mezi příjezdem popředu a zacouváním, nebo působení vozidla parkovacího na vedlejším místě.



Obrázek 6.4: Porovnání rozdílů mezi příjezdem popředu a pozadu, vliv příjezdu na vedlejší parkovací místo (Škoda Octavia)

Z obrázku 6.4 je patrné, že mezi příjezdem pozadu a popředu je jistý rozdíl v amplitudě naměřeného signálu. Nicméně, hodnota amplitudy závisí i na tom, jak blízko auto k senzoru přijelo, tedy je možné i to, že při couvání bylo auto prostě jen blíže k senzoru. Každopádně tento rozdíl nemá žádný vliv na detekci, neboť nijak neovlivňuje časové posunutí signálů.

Jistý problém by mohl být automobil přijíždějící na vedlejší parkovací místo. Jak lze vidět na posledním grafu na obrázku 6.4, jeho signatura v magnetickém poli je sice slabá, ale patrná. Pro detekci s použitím ostatním metod by to neměl být problém, protože se ale korelace nedívá na velikost amplitudy, mohlo

by dojít k falešné detekci. V případě, že by se korelační metoda ukázala jako vhodná pro použití v reálných podmínkách, bylo by nejspíš nutné algoritmus upravit tak, aby při detekci pohlížel i na velikost amplitudy.

6.2 Pokusy s druhým senzorem

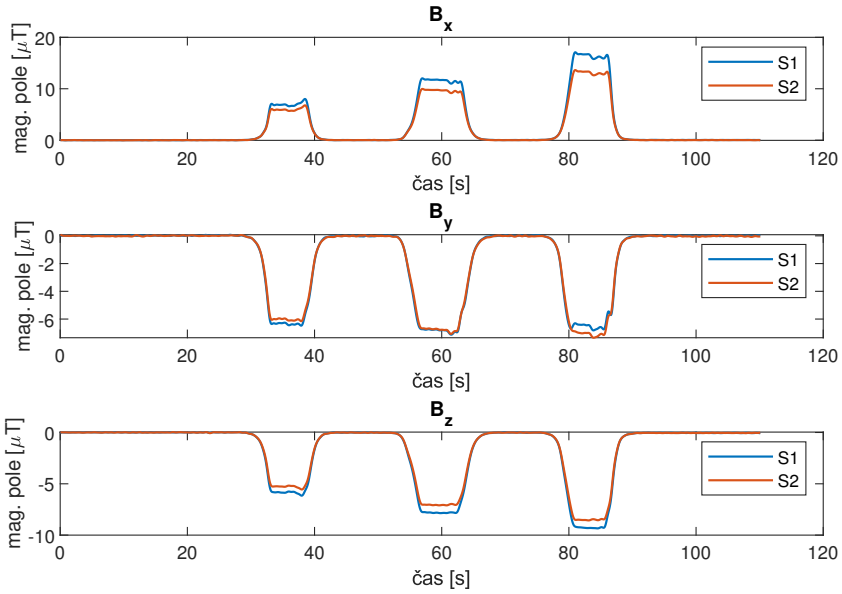
Druhá verze senzoru už měřila magnetické pole ve všech třech osách. Na výstupu tedy posílal detektor šest hodnot, za každý AMR senzor měření ve všech třech osách. To všechno ale na úkor vzorkovací frekvence, která se snížila na 29.3 Hz.

I na tomto senzoru se, stejně jako u první verze, testoval detekční algoritmus s využitím vzájemné korelace, zároveň se ale začalo i s pokusy s jednoduchými neuronovými sítěmi.

délka časového okna	30 vzorků (1 s)
velikost kroku	15 vzorků (0.5 s)

Tabulka 6.2: Parametry korelačního algoritmu - druhá verze detektoru

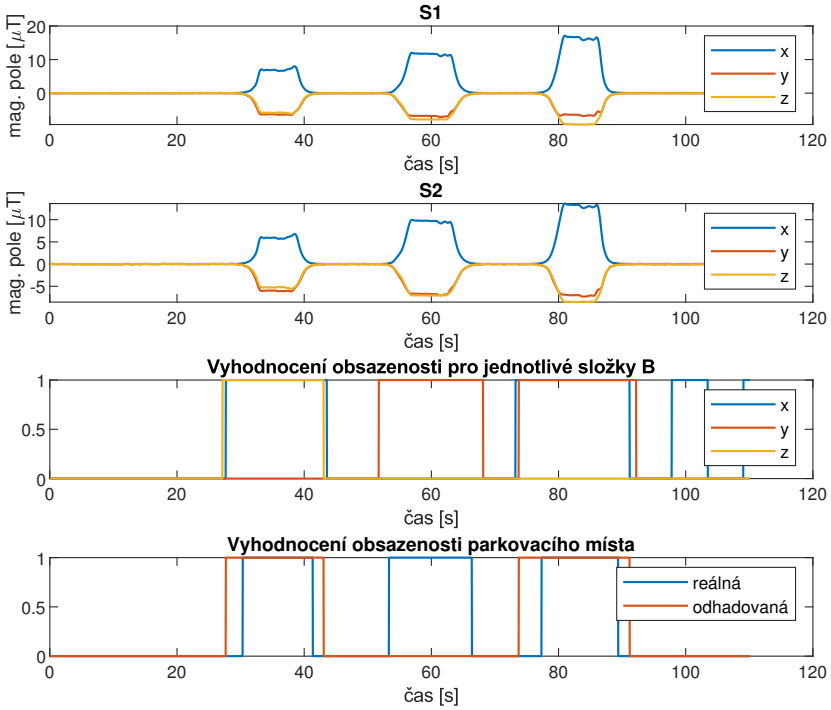
Během testování korelačního algoritmu s touto verzí bylo opět vyzkoušeno mnoho různých kombinací parametrů, přičemž nejlépe vyšly hodnoty v tabulce 6.2. Úspěšnost detekce s tímto nastavením byla 51 % podle prvního kritéria hodnocení a 36.9 % podle druhého.



Obrázek 6.5: Ukázka dat z druhé verze senzoru (Škoda Rapid)

Hodnoty naměřené oběma senzory v jednotlivých osách jsou na obrázku 6.5. Původní záměr využití pro korelaci byl takový, že by se korelačním algoritmem

postupně prohnaly data v jednotlivých osách, následně by se porovnaly, a pokud by alespoň ve dvou osách byla detekována přítomnost, výsledek detekce by bylo obsazeno.



Obrázek 6.6: Detekce s využitím korelace pro jednotlivé osy (Škoda Rapid)

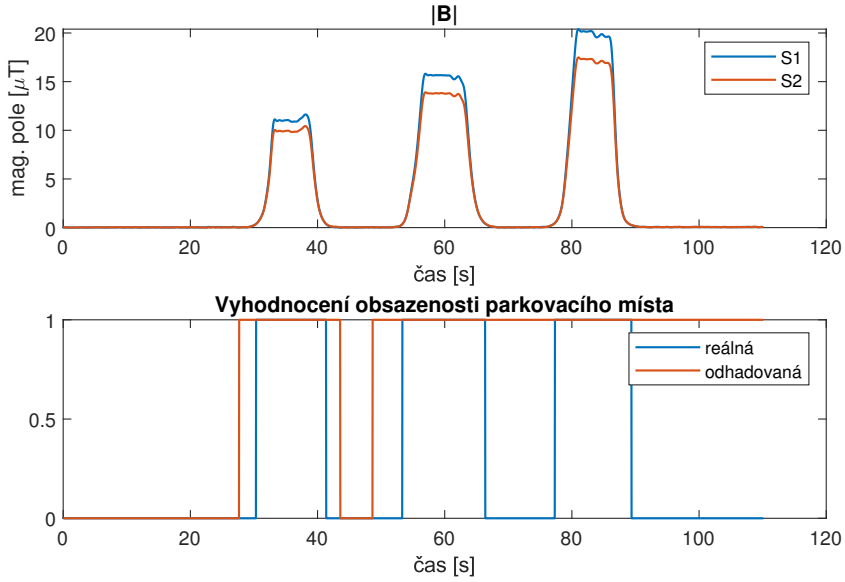
Protože se první způsob detekce příliš neosvědčil (celkově nižší úspěšnost, vyšší výpočetní náročnost), bylo potřeba vymyslet, jak celou úlohu zjednodušit, aby nebylo třeba spouštět korelační algoritmus několikrát.

Jednou z možností by bylo vzít pouze data naměřená v ose x, což by bylo obdobné jako v případě detekce u první verze senzoru. Druhý nápad byl vypočítat z jednotlivých složek celkovou velikost B (tj. $B = \sqrt{B_x^2 + B_y^2 + B_z^2}$).

To má tu výhodu, že změna signálu při příjezdu auta je výraznější a vždy jde do kladných čísel, což sice pro samotnou korelaci nehraje příliš velkou roli, ale využití v ostatních metodách to značně zjednoduší.

Příklad funkce detekčního algoritmu je na obrázku 6.7. Na vyhodnocení obsazenosti lze vidět, že algoritmus správně detekoval první příjezd a odjezd (sice s jistou chybou, ale ta je způsobená i tím, že samotná reálná přítomnost je spíše odhadnutá, protože tahle verze senzoru nemá žádné další referenční měření pro zjištění obsazenosti).

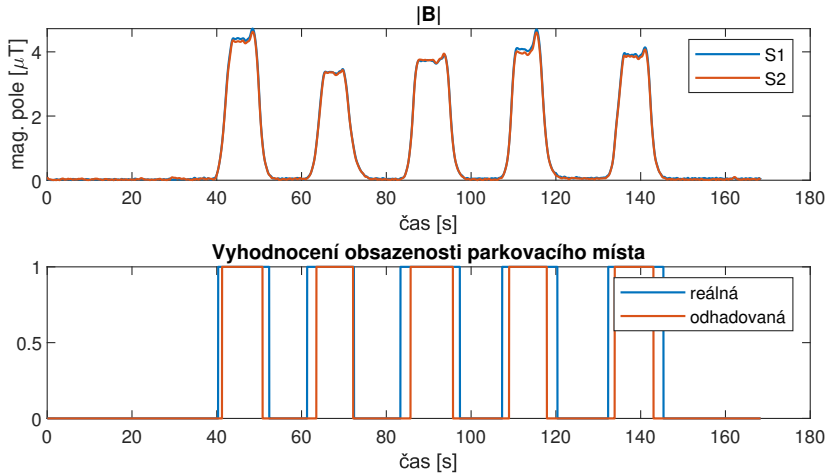
Druhý příjezd detekoval také, ale odjezd už ne. Třetí příjezd tím pádem ani registrovat nemohl, protože bral místo stále jako obsazené. Celková úspěšnost detekce pro toto měření je 62.5 %, resp. 46.7 %.



Obrázek 6.7: Detekce s využitím korelace pro |B| (Škoda Rapid)

Z výsledků lze vidět, že první verze detektoru byla výrazně úspěšnější než druhá. To je způsobeno pravděpodobně tím, že druhý senzor má značně nižší vzorkovací frekvenci, a protože korelační algoritmus běží jen v krátkých časových oknech, nemá dost hodnot pro porovnání.

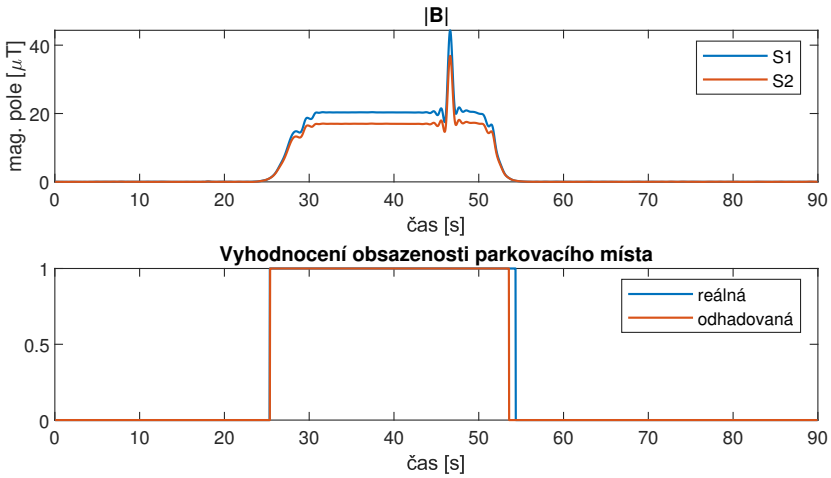
První pokusy s neuronovými sítěmi dopadly o něco nadějněji, než detekce s využitím vzájemné korelace. Pro detekční algoritmus byla vytvořena jednoduchá neuronová síť s dvěma vstupy ($|B_{S1}|$ a $|B_{S2}|$), jednou skrytou vrstvou s dvěma neurony a jedním výstupem (volno = 0, obsazeno = 1).



Obrázek 6.8: Detekce s využitím neuronové sítě (Škoda Octavia)

Celkem bylo provedeno jedenáct měření - šest s Octavií a pět s Rapidem.

Z toho bylo šest měření (tři za každé auto) použito na učení sítě a její funkčnost se pak testovala na všech jedenácti měření. Výsledky detekce se ukládaly do textových souborů, které se dále v Matlabu porovnávaly s reálnou obsazeností.



Obrázek 6.9: Detekce s využitím neuronové sítě (Škoda Rapid)

Příklady detekce jsou na obrázcích 6.8 a 6.9. V prvním příkladu je to pět příjezdů a odjezdů, aniž by byl vypnutý motor. V druhém případě lze před odjezdem vidět špičku, když se motor startoval.

Přesnost detekce se opět zkoumala dvěma způsoby. Celková přesnost po vyhodnocení všech měření byla přibližně 94 %, respektive 84.2 %. Jde o odchylky maximálně v rámci sekund, tedy pro reálné využití by neuronové sítě mohly mít velký potenciál.

6.3 Konečná verze senzoru

Původní záměr byl instalovat několik kusů těchto senzorů na parkoviště FEL ČVUT v Praze. Bohužel kvůli situaci kolem pandemie COVID-19 toto nebylo možné. Všechna měření s poslední verzí senzoru tedy opět proběhla v domácích podmínkách.

Pro měření s tímto senzorem se mi povedlo sehnat více modelů automobilů, konkrétně Škoda Octavia, VW Tiguan, Honda CR-V a Opel Meriva.

6.3.1 Zpracování referenčního měření ultrazvukem

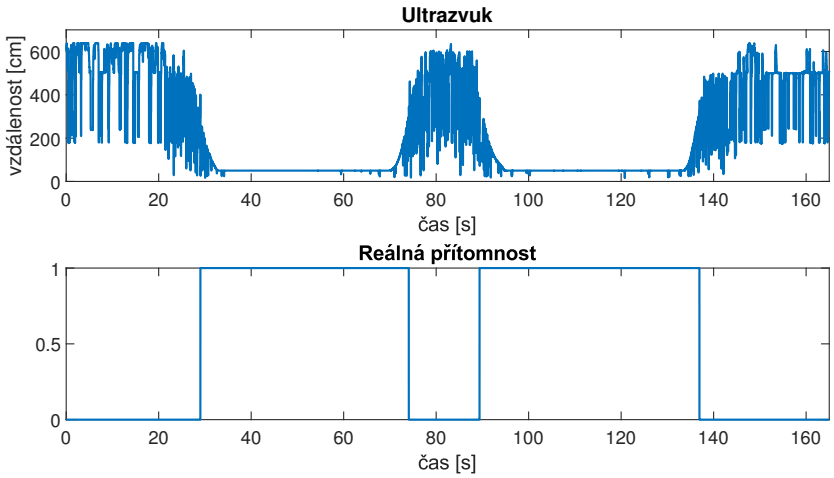
Na boku krabičky s detektorem je umístěn ultrazvukový senzor vzdálenosti. Ten slouží jako reference pro zjištění reálné přítomnosti vozidla na parkovacím místě.

Výstupem ultrazvuku je vzdálenost automobilu od senzoru (v centimetrech). Jako hranici pro rozhodování o přítomnosti beru 3 m, tedy pokud je vzdálenost

naměřená ultrazvukem menší než 300 cm, tak je obsazeno (log. 1), jinak volno (log. 0).

Na horním grafu obrázku 6.10 je vidět, že ultrazvuk ne vždy měří dobře. Jak je vidět na tomto příkladu, v některých případech přeskakovala výstupní vzdálenost na nižší hodnoty, přestože před ultrazvukem žádná překážka nebyla.

Tento problém je řešen v kódu tak, že se projde celý časový úsek měření, a odstraní se z něj změny, které trvaly kratší dobu než několik sekund. Na spodním grafu je vidět zobrazení výsledné obsazenosti.



Obrázek 6.10: Referenční měření ultrazvukem

6.3.2 Vzájemná korelace

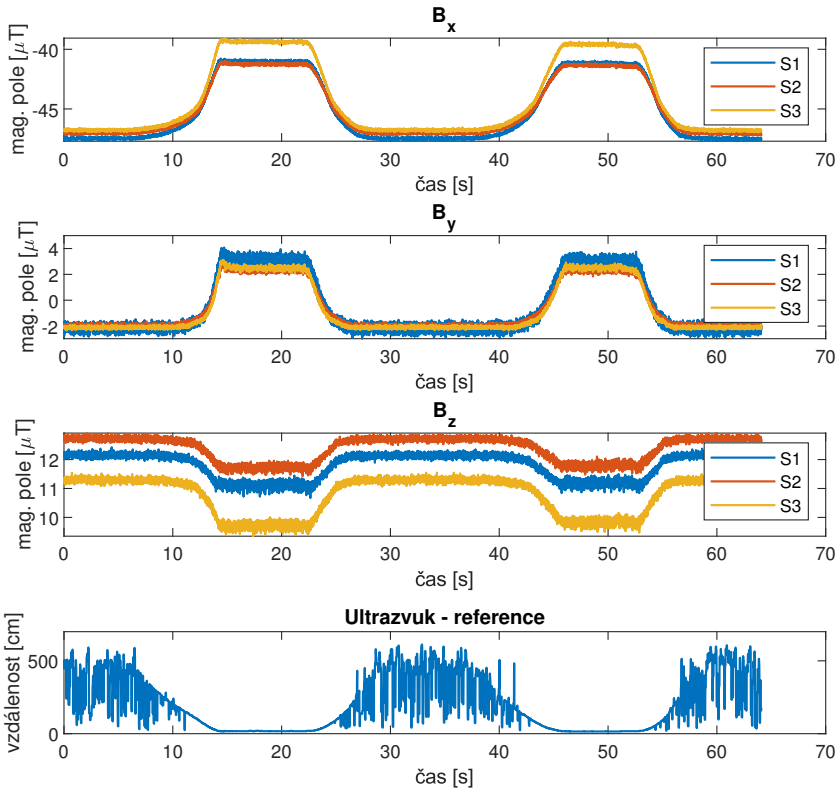
Při detekci s využitím korelace se postupovalo obdobně jako v případě předchozí verze senzoru. Data z měření byla nejprve upravena a pro každý ze tří senzorů byla vypočtena velikost B. Tyto hodnoty se pak uložily pro pozdější použití v ostatních metodách detekce.

Pro samotnou korelaci se opět využilo signálů $|B_{S1}|$ a $|B_{S2}|$. Vyzkoušeno bylo opět několik kombinací nastavení, nejlépe dopadla detekce s parametry v tabulce 6.3. Úspěšnost testování metody na všech měření byla 73.6 % podle prvního kritéria hodnocení a 72.4 % podle druhého.

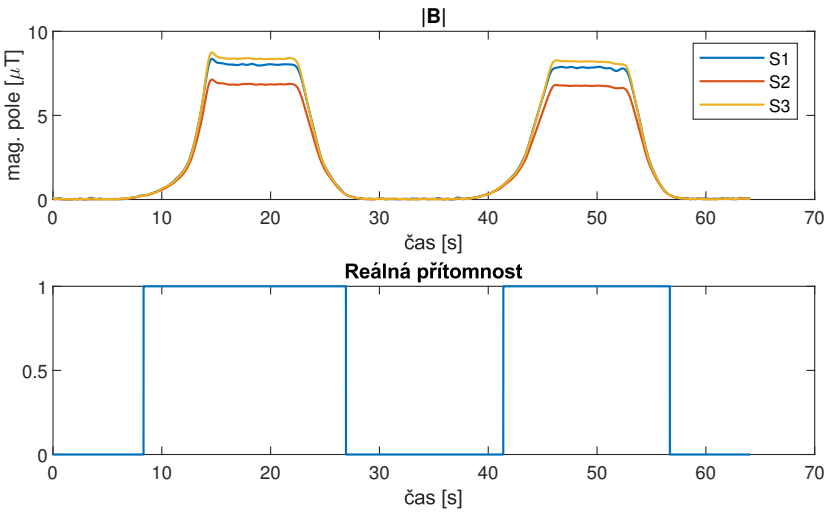
délka časového okna	100 vzorků (cca 0.7 s)
velikost kroku	30 vzorků (0.2 s)

Tabulka 6.3: Parametry korelačního algoritmu - konečná verze detektoru

Na obrázku 6.11 jsou zobrazeny hodnoty získané ze senzorů před úpravami pro další použití. Na obrázku 6.12 jsou to pak stejná data, ale již upravená pro potřeby detekčních algoritmů.

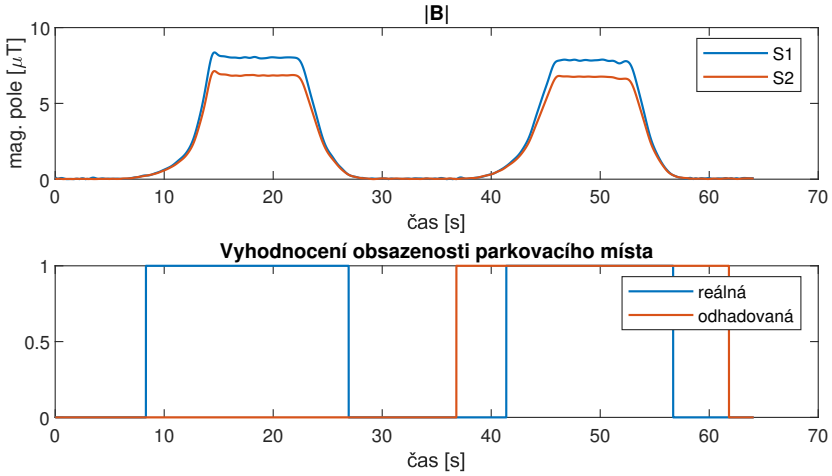


Obrázek 6.11: Data přijatá ze senzorů (Opel Meriva)

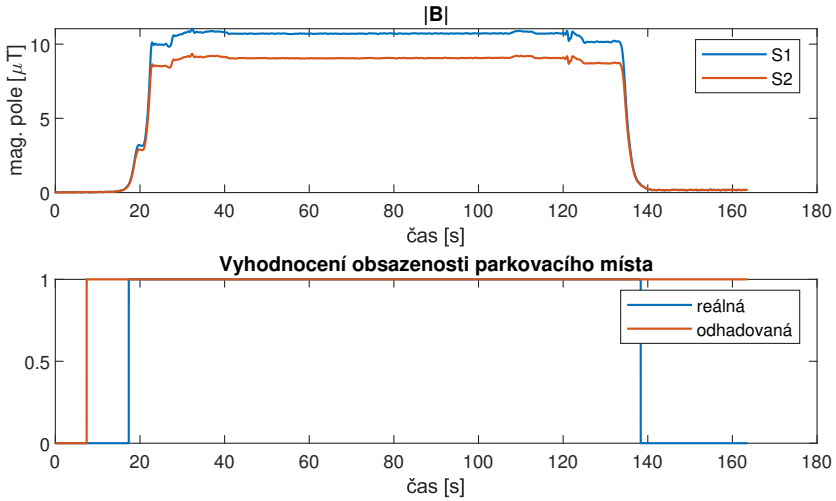


Obrázek 6.12: Data upravená pro použití v detekčních algoritmech (Opel Meriva)

Na obrázcích 6.13 a 6.14 jsou vyobrazeny výsledky detekce pro dvě různá měření.



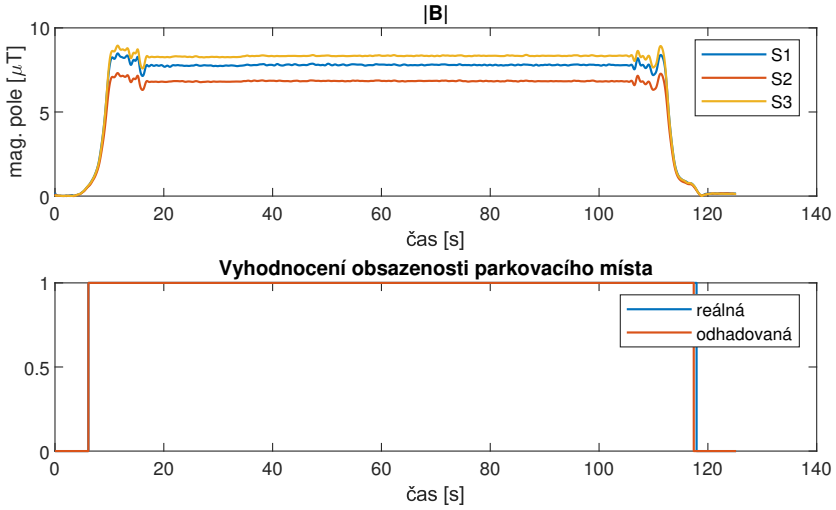
Obrázek 6.13: Ukázka detekce s využitím vzájemné korelace (Opel Meriva)



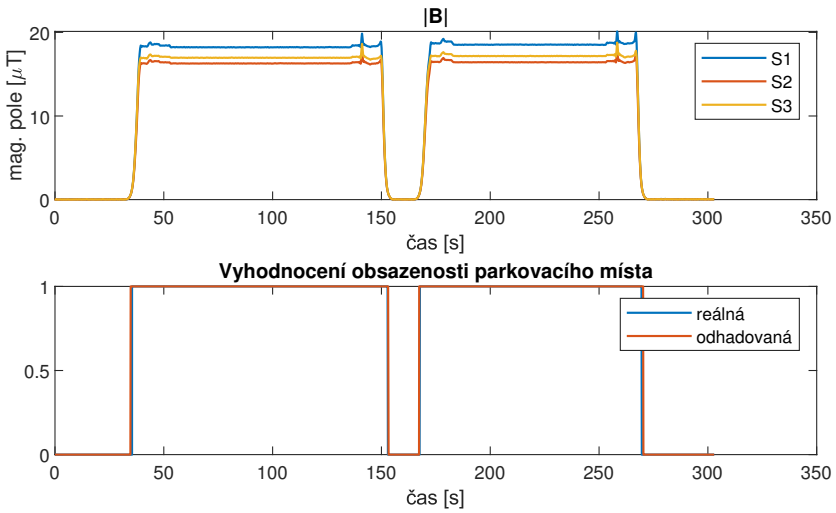
Obrázek 6.14: Ukázka detekce s využitím vzájemné korelace (VW Tiguan)

6.3.3 Neuronové sítě

Výsledky detekce s využitím neuronové sítě dopadly vcelku uspokojivě. Pro osmnáct měření byla úspěšnost detekce 97.2 %, respektive 96.5 %. Přitom stačilo síť naučit několika měřeními se Škodou Octavia, a přesto byla schopná s vysokou přesností detekovat i ostatní typy automobilů.



Obrázek 6.15: Ukázka detekce s využitím neuronové sítě (Opel Meriva)

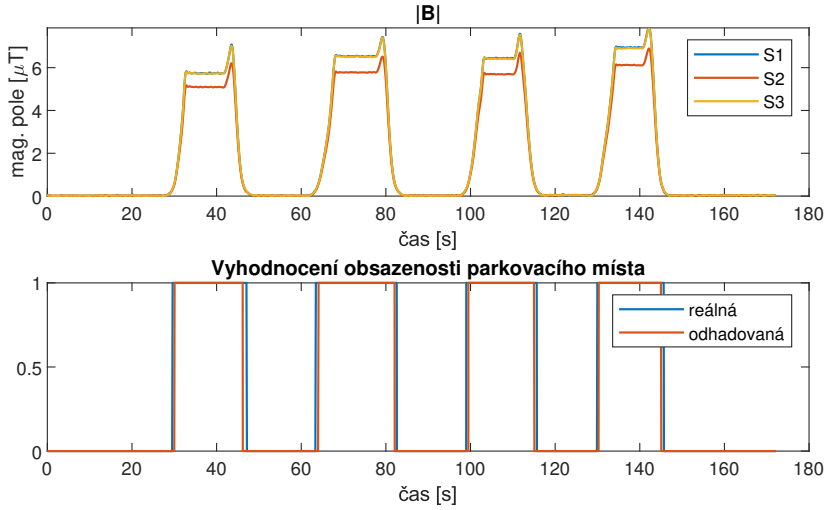


Obrázek 6.16: Ukázka detekce s využitím neuronové sítě (Honda CR-V)

Jak již bylo řečeno, neuronová síť má pro aplikaci v detektoru velký potenciál. Otázkou ale je, jak by si poradila s auty s malou signaturou v magnetickém poli.

Problém je v tom, že senzory registrují i změnu způsobenou autem stojícím vedle a síť naučená na malá auta (respektive auta, jejichž vliv na magnetické pole je menší, což nemusí přímo souviset s jejich velikostí), by pak mohla dojít k falešně pozitivnímu výsledku v případě příjezdu automobilu na vedlejší místo.

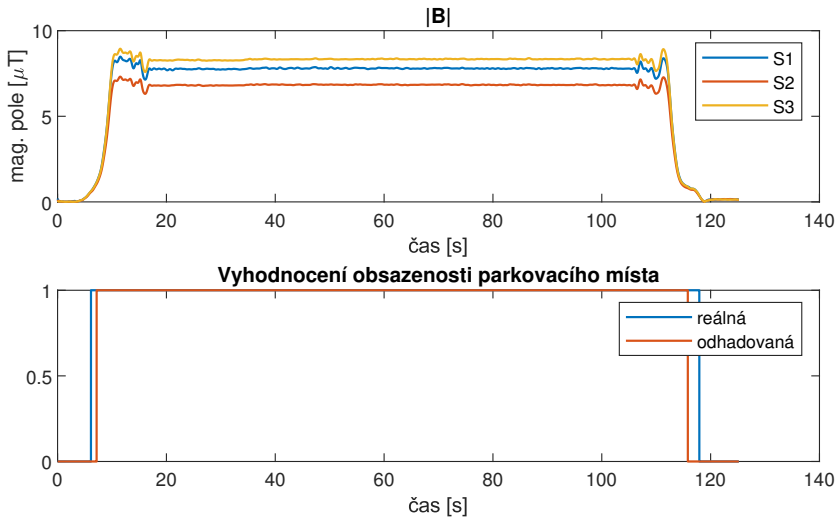
Více o této problematice bude zmíněno dále v této práci.



Obrázek 6.17: Ukázka detekce s využitím neuronové sítě (VW Tiguan)

6.3.4 Nejbližší soused

Také tato metoda dosáhla dobré úspěšnosti 97.4 %, respektive 94.9 %. Příklad detekce je na obrázku 6.18.

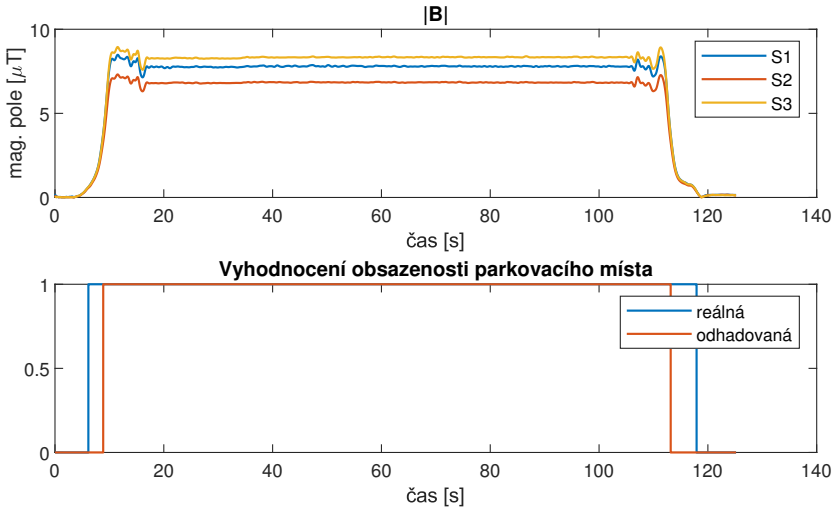


Obrázek 6.18: Ukázka detekce s využitím metody nejbližšího souseda (Opel Meriva)

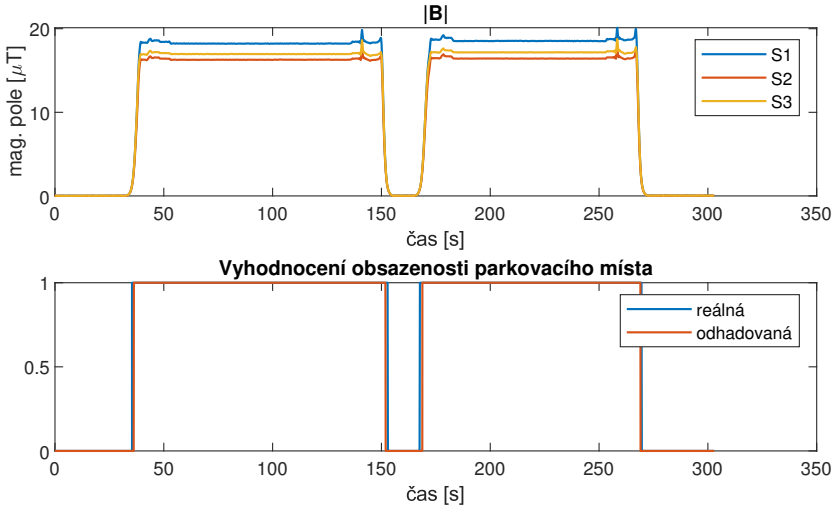
Nicméně, při implementaci a testování tohoto způsobu detekce se ukázalo, že je příliš časově i paměťově náročný, než aby se dal rozumně použít pro aplikaci detektoru.

6.3.5 Lineární regrese a klasifikace

Ještě o něco lepších výsledků než předchozí metody dosáhla detekce s využitím lineární regrese. Pro stejná data jako u předchozích pokusů to bylo 97.6 %, respektive 96.9 %.



Obrázek 6.19: Ukázka lineární klasifikace (Opel Meriva)

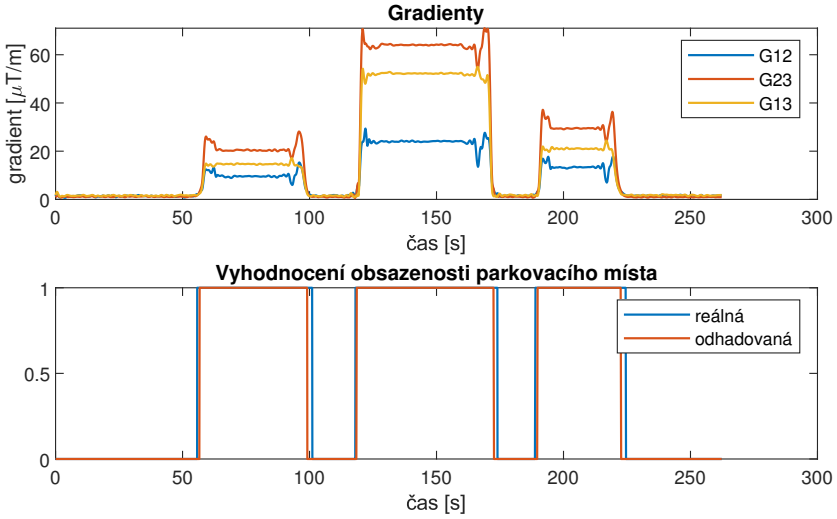


Obrázek 6.20: Ukázka lineární klasifikace (Honda CR-V)

6.4 Detekce s využitím gradientu magnetického pole

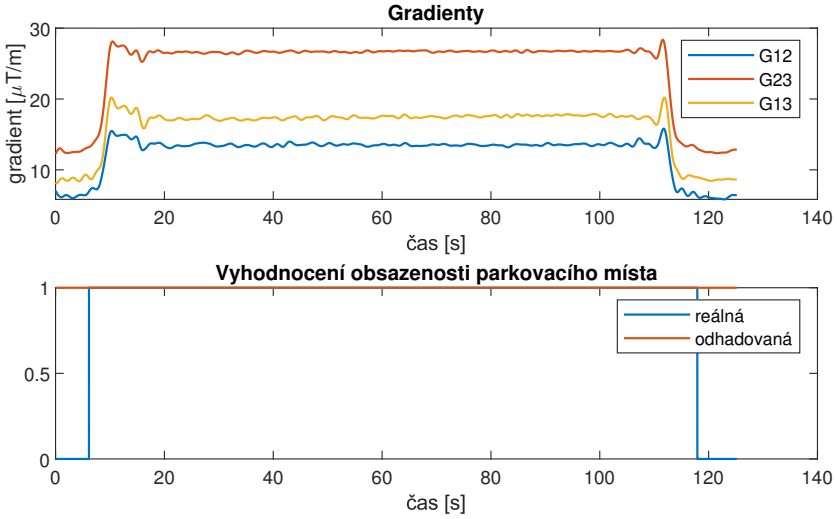
Protože je senzor umístěn v zemském magnetickém poli, může i nepatrná změna jeho polohy způsobit, že se změní offset senzoru. To se dá řešit jen

pravidelnou kalibrací, což by v případě této aplikace mohlo být značně obtížné. Proto se pro detekční algoritmy počítalo s využitím gradientu (viz [1]).



Obrázek 6.21: Neuronová síť (Škoda Octavia)

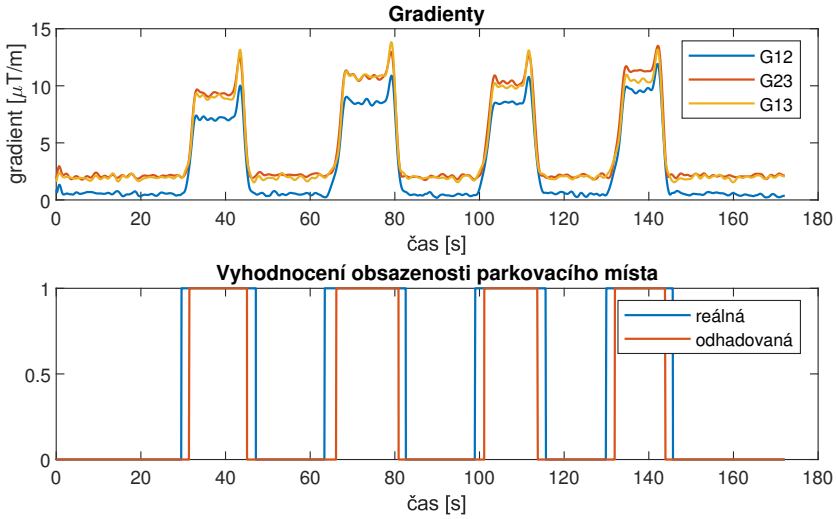
Úspěšnost neuronové sítě byla v tomto případě 98.5 %, respektive 98 %, což je lepší výsledek než u pokusů s původními daty.



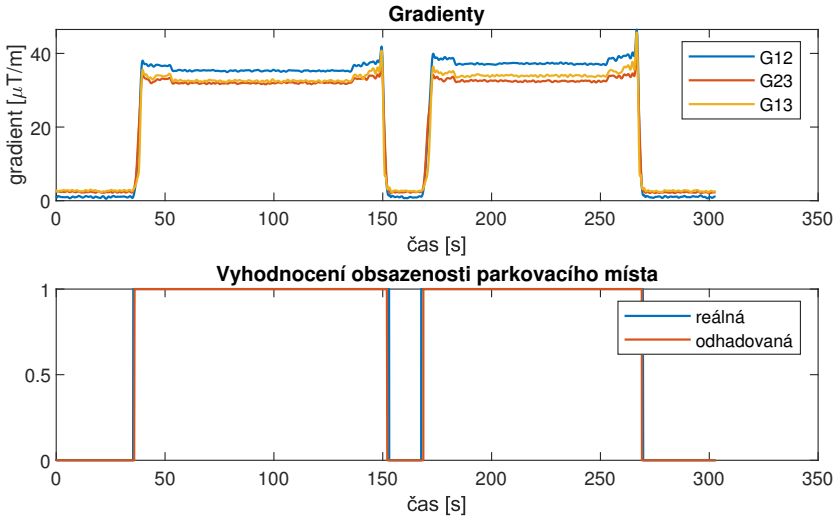
Obrázek 6.22: Neuronová síť (Opel Meriva)

Na obrázku 6.21 je zobrazen příklad úspěšné detekce. Na obrázku 6.22 pak lze vidět případ měření, kdy gradienty nevyšly právě dobře, což následně negativně ovlivnilo i rozhodnutí detekčního algoritmu.

Na obrázcích 6.23 a 6.24 jsou výsledky lineární klasifikace pro gradienty. Tato metoda si tentokrát vedla o něco málo hůře než neuronová síť, dosáhla přesnosti 98.4 %, resp. 97.8 %.



Obrázek 6.23: Lineární klasifikace (VW Tiguan)



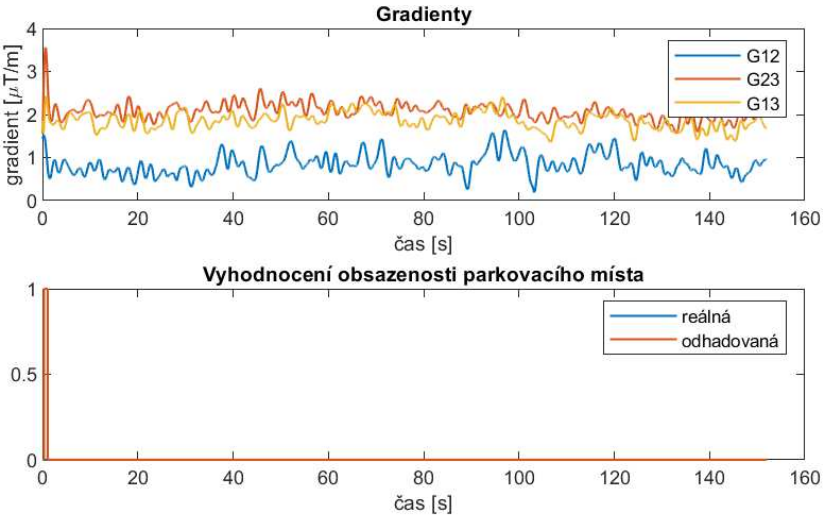
Obrázek 6.24: Lineární klasifikace (Honda CR-V)

6.5 Vliv automobilu přijíždějícího na vedlejší místo

Po otestování všech metod jsem dále zkoušela jejich odolnost vůči příjezdu vozidla na vedlejší parkovací místo.

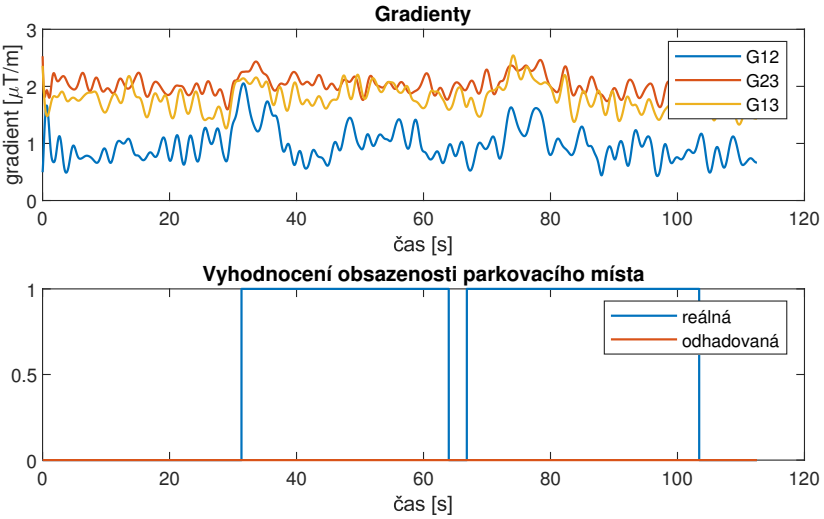
V případě detekce s využitím korelace, algoritmus nedetekoval příjezd na vedlejší místo, ale protože jeho celková úspěšnost nebyla nijak vysoká, je otázka, nakolik je jeho případná odolnost vůči tomuto jevu průkazná.

Naopak u neuronové sítě se dá říct, že její odolnost je celkem průkazná, pokud nedetekovala žádný vedlejší příjezd, neboť předchozí pokusy s ní byly dost přesné. Příklad jednoho měření je na obrázku 6.25.



Obrázek 6.25: Neuronová síť - vliv příjezdu na vedlejší parkovací místo (Škoda Octavia)

Dobrého výsledku také dosáhla lineární klasifikace. Při jejím použití algoritmus nedetekoval jediný falešně pozitivní příjezd. Na obrázku 6.26 je zobrazení jednoho pokusu detekce.

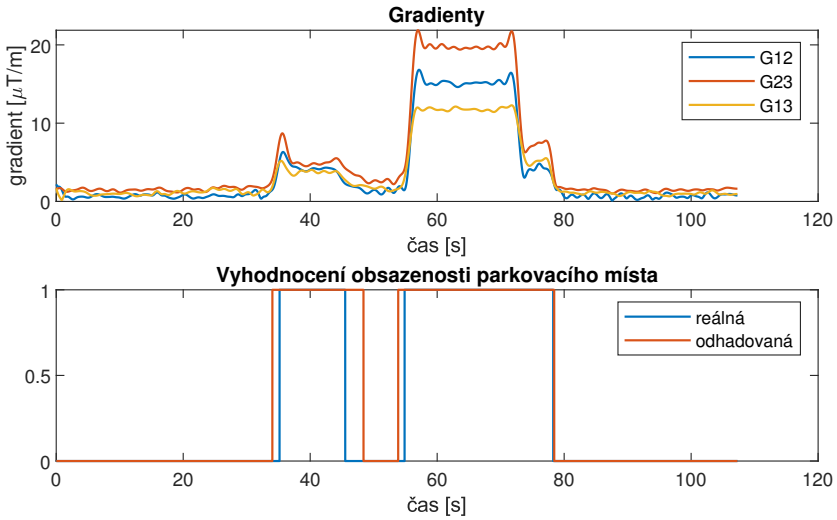


Obrázek 6.26: [Lineární klasifikace - vliv příjezdu na vedlejší parkovací místo (Škoda Octavia)

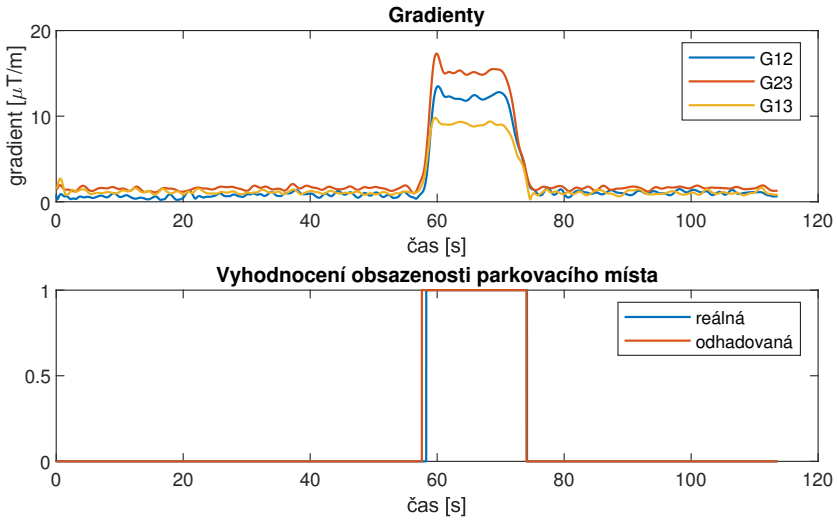
6.6 Podélné parkování

Na závěr jsem se rozhodla ještě otestovat, jak vybrané metody reagují na podélné parkování. K tomu účelu jsem provedla dvě pokusná měření. V prvním případě šlo o zajetí mezi dvě již stojící vozidla, tedy na místo se muselo

zajet pozpátku a pak auto srovnat. V druhém jsem simulovala situaci, kdy je prostor kolem volný, tedy není problém na místo zajet popředu.



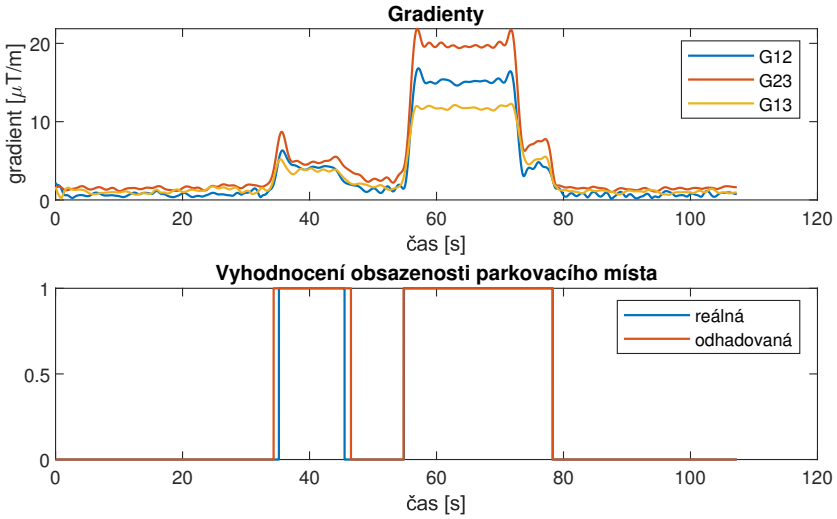
Obrázek 6.27: Ukázka detekce s využitím neuronové sítě (Škoda Octavia)



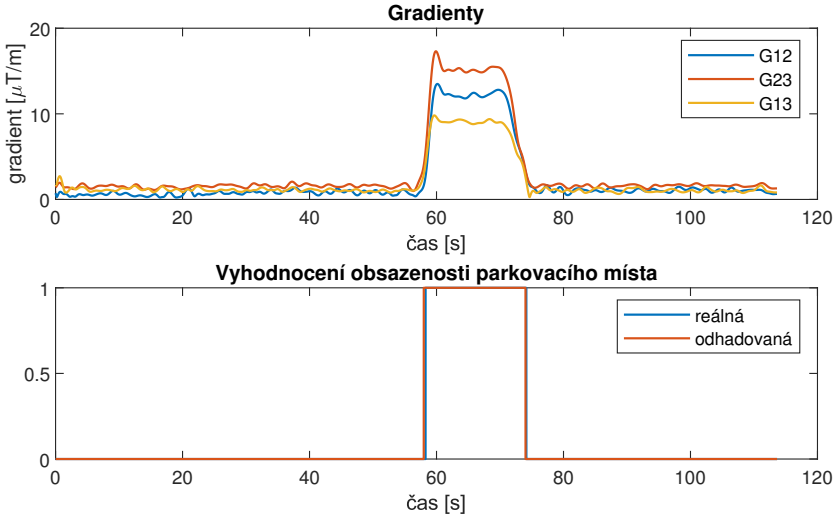
Obrázek 6.28: Ukázka detekce s využitím neuronové sítě (Škoda Octavia)

Na obrázcích 6.27 a 6.28 jsou výsledky detekce s použitím neuronové sítě pro oba výše zmíněné případy. Na obrázcích 6.29 a 6.30 pak stejné příklady, ale s využitím lineární klasifikace.

Je vidět, že si obě metody s podélným parkováním poradily více než slušně, což ukazuje, že použití detektoru by nemuselo být omezené pouze pro parkoviště s příčným parkováním.



Obrázek 6.29: Ukázka detekce s využitím lineární klasifikace (Škoda Octavia)



Obrázek 6.30: Ukázka detekce s využitím lineární klasifikace (Škoda Octavia)

Vzájemná korelace nebyla v tomto případě testována, protože při takovém příjezdu nebyly signály z jednotlivých senzorů dostatečně časově posunuty.

Kapitola 7

Závěr

Přestože průběh práce byl negativně ovlivněn krizovým stavem kolem pandemie COVID-19, podařilo se úspěšně otestovat všechny vybrané metody, které se zdály být vhodné pro využití v aplikaci detektoru.

Výsledky testů ukázaly, že použití vzájemné korelace na signály ze dvou AMR senzorů pro zjištění příjezdu/odjezdu vozidla nemá příliš dobrou úspěšnost, přestože při dřívějších pokusech se korelace s úspěchem využívalo pro měření rychlosti automobilu.

Naopak výrazně lepší přesnost detekce ukázaly ostatní metody. V případě neuronové sítě to byla vysoká úspěšnost detekce pro všechny modely vozů a při dalších pokusech se potvrdilo, že je odolná vůči rušivým prvkům z okolí (příjezd automobilu na vedlejší parkovací místo) a lze použít i pro instalaci vedle automobilu.

Metoda nejbližšího souseda dosáhla také velmi slušných výsledků. Nicméně, i když byla trénovací množina dat relativně malá (jedno měření), potřeboval algoritmus na detekci hodně času. Také paměťová náročnost algoritmu je vysoká, což z ní nedělá právě vhodnou metodu pro tuto aplikaci.

Nadějné výsledky měla také lineární klasifikace. Přesnost detekce byla vyšší než u ostatních metod a zároveň se prokázala i jistá odolnost vůči rušení způsobeného příjezdem automobilu na vedlejší místo. Navíc využití detektoru by nebylo limitováno pouze na příčné parkování.

Lineární klasifikace se jeví jako nejvhodnější metoda pro tuto aplikaci. Měla sice nepatrně horší výsledky než neuronová síť, ale ze všech metod je výpočetně nejméně náročná. Je sice potřeba najít w, ale to lze udělat třeba v Matlabu, takže při detekci samotné stačí jen vynásobit vstupní data a podle výsledku rozhodnout o stavu.

Dalším krokem by určitě mělo být vylepšení měření gradientu a otestování detekčního algoritmu v reálných podmínkách na nějakém parkovišti. To mělo být i součástí této práce, ale bohužel opatření kvůli pandemii COVID-19 toto znemožnila.

Literatura

[1] NOVOTNÝ, David. Magnetický gradiometr pro detekci automobilů a měření rychlosti. Praha, 2018. Diplomová práce. ČVUT.

[2] M. JANOŠEK, A. PLATIL a J. VYHNÁNEK. Simple estimation of dipole source z-distance with compact magnetic gradiometer. IOP Conference Series: Materials Science and Engineering [online]. 2016, 108, 012025- [cit. 2018-05-19]. DOI: 10.1088/1757-899X/108/1/012025. ISSN 1757-8981. Dostupné z: http://stacks.iop.org/1757-899X/108/i=1/a=012025?key=crossref.babc2ab0e6fc636edbf0f52544f2176e6

[3] JANOŠEK, Michal, Jan VYHNÁNEK a Antonín PLATIL. Compact Magnetic Gradiometer and its Astatization. Procedia Engineering [online]. 2015, 120, 1249-1252 [cit. 2018-05-19]. DOI: 10.1016/j.proeng.2015.08.841. ISSN 18777058. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1877705815025114

[4] NOVOTNÝ, D. et al. Vehicle's magnetic field modeling and mapping for its presence detection. In: Magnetic Frontiers 2019: Magnetic Sensors - Abstract Book. Magnetic Frontiers 2019, Lisabon, 2019-06-24/2019-06-27. Lisbon: Instituto Superior Técnico, Technical University of Lisbon, 2019. Dostupné z: https://magfrontiers.sciencesconf.org/data/pages/Magnetic_Frontiers_2019_Abstract_Book_1.pdf

[5] Elektronický kompas - AMR senzor [online]. 3-4 [cit. 2020-03-29]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/210204/mod_folder/content/0/_AMR-navod24.pdf?forcedownload=1

[6] PLATIL, Antonín, Michal JANOŠEK a Pavel RIPKA. Magnetic sensors & applications [online]. 18-24 [cit. 2020-03-29].

[7] Korelační metoda měření rychlosti [online]. [cit. 2020-03-29]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/210197/mod_resource/content/7/A%20-%20Korelacni_mereni_rychlosti.pdf

[8] Matlab Documentation [online]. [cit. 2020-03-29]. Dostupné z: https://www.mathworks.com/help/matlab/

[9] Neuronové sítě [online]. [cit. 2020-03-29]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471

[10] Neuronové sítě a princip jejich fungování [online]. [cit. 2020-03-29]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/

[11] POŠÍK, Petr. Neural Networks [online]. 17-25 [cit. 2020-03-29]. Dostupné z: https://cw.fel.cvut.cz/b182/_media/courses/ui/b12nn-slides.pdf

[12] POŠÍK, Petr. Nearest neighbors. Kernel functions, SVM. Decision trees. [online]. 3-10 [cit. 2020-03-29]. Dostupné z: https://cw.fel.cvut.cz/b182/_media/courses/ui/b04nonlinear-slides.pdf

[13] POŠÍK, Petr. Linear Methods for Regression and Classification [online]. 2-15 [cit. 2020-05-13]. Dostupné z: https://cw.fel.cvut.cz/b182/_media/courses/ui/b02linear-slides.pdf

[14] STM32L432KB STM32L432KC Ultra-low-power Arm® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 256KB Flash, 64KB SRAM, USB FS, analog, audio: Ultra-low-power Arm® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 256KB Flash, 64KB SRAM, USB FS, analog, audio [online]. [cit. 2020-05-20]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32l432kc.pdf

[15] STM32L433xx Ultra-low-power Arm® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 256KB Flash, 64KB SRAM, USB FS, LCD, ext. SMPS [online]. [cit. 2020-05-20]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32l433cc.pdf

[16] MMC5883MA ±8Gauss, High Performance, Low Cost 3-axis Magnetic Sensor [online]. [cit. 2020-05-20]. Dostupné z: https://www.memsic.com/userfiles/files/DataSheets/Magnetic-Sensors-Datasheets/MMC5883MA-RevC.pdf

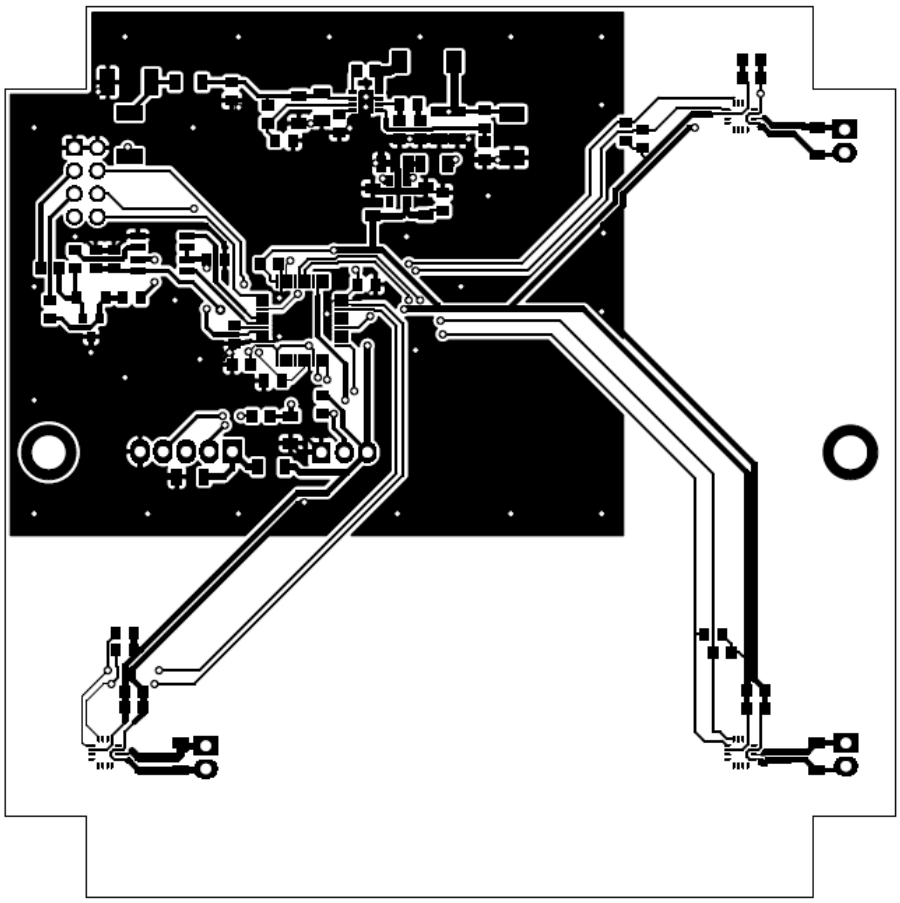
Příloha A

Seznam souborů přiložených na CD

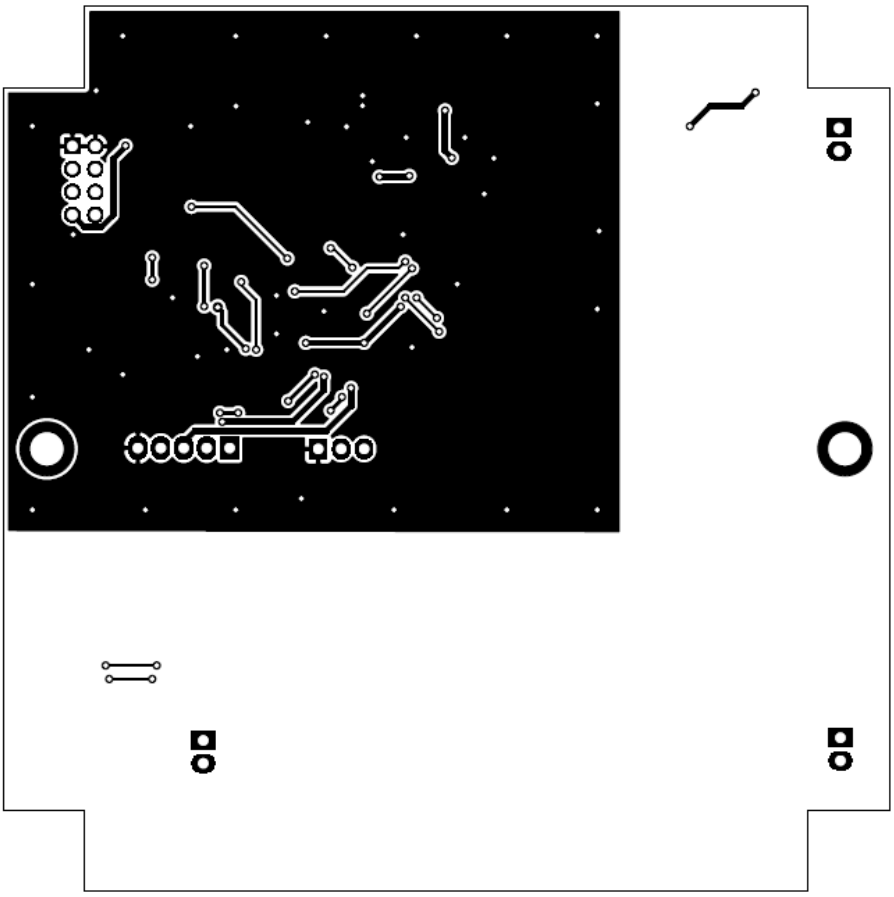
- **detection** - jednotlivé algoritmy a data z měření
  - **kNN** - skripty a data k metodě nejbližších sousedů
  - **linear_classification** - skripty a data pro lineární klasifikaci
    - **data** - data ($|B|$) z poslední verze detektoru
    - **gradienty** - gradienty (poslední verze detektoru)
  - **NN** - skripty a data pro neuronovou síť
    - **senzor2** - data ($|B|$) z druhé verze detektoru
    - **senzor3** - data ($|B|$) z poslední verze detektoru
    - **gradienty** - gradienty (poslední verze detektoru)
  - **xcorr** - skripty a data pro detekci s využitím vzájemné korelace
    - **senzor1** - první verze detektoru
    - **senzor2** - druhá verze detektoru
    - **senzor3** - konečná verze detektoru
- **figures** - obrázky z měření a s výsledky detekce
- **MMC5883_autodet** - projekt v KiCadu s návrhem DPS

Příloha B

DPS konečné verze detektoru



Obrázek B.1: DPS konečné verze detektoru - strana TOP



Obrázek B.2: DPS konečné verze detektoru - strana BOTTOM