České vysoké učení technické v Praze
Fakulta elektrotechnická

Program: Elektronika a komunikace

Analýza dat z automobilového FMCW radaru

Analysis of Data from Automotive FMCW Radar

BAKALÁŘSKÁ PRÁCE

Vypracoval: Jakub Kiňovič
Vedoucí práce: Ing. Viktor Adler, Ph.D.
Rok: 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kiňovič** Jméno: **Jakub** Osobní číslo: **483888**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektromagnetického pole**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Analýza dat z automobilového FMCW radaru

Název bakalářské práce anglicky:

Analysis of Data from Automotive FMCW Radar

Pokyny pro vypracování:

Prostudujte a v jazyce Matlab implementujte zpracování mezifrekvenčních signálů z automobilového FMCW radaru Valeo MRR z testovací jízdy po městském prostředí. U detekovaných cílů použijte metodu MUSIC k určení přesného azimutu cílů. Do dostupných záznamů z palubních kamer z testovací jízdy zobrazte pozice detekovaných cílů.

Seznam doporučené literatury:

Brooker, M., G.: Understanding Millimetre Wave FMCW Radars, Proceedings of 1st International Conference on Sensing Technology, New Zealand, 2005
Sandeep, R.: MIMO Radar, Application Report, Texas Instruments, 2018
Chen Z., Gokeda G., Yu Y.: Introduction to Direction-of-Arrival Estimation, Artech House, 2010
Li, J., Stoica, P.: MIMO Radar Signal Processing, John Wiley & Sons, Inc,, New York, 2009
Bezoušek, P., Šedivý, P.: Radarová technika, Vydavatelství ČVUT, Praha, 2004

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Viktor Adler, Ph.D., katedra elektromagnetického pole FEL

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

Datum zadání bakalářské práce: **24.01.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2022**

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

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v práci a v přiloženém seznamu.

V Praze dne ....

Jakub Kiňovič

Poděkování

Děkuji Ing. Viktoru Adlerovi, Ph.D. a Ing. Milanu Kvičerovi, Ph.D. za jejich trpělivost, čas a cenné rady při vedení mé bakalářské práce.

Jakub Kiňovič

Abstrakt

Tato práce se soustředí na popis principu činnosti FMCW radarů a jejich použití pro registraci radarového obrazu okolí automobilu. Klade si za cíl obecně a názorně popsat princip FMCW radarů, anténních řad pro MIMO kanál a metod pro určení DOA, zpracování vstupního signálu, výpočet vzdálenosti z frekvence a princip detekce rychlosti. V praktické části ukazuje porovnání metody 3D-FFT a MUSIC pro hledání azimutu radarového cíle a seznamuje s postupem a kódem použitým pro spojení radarového obrazu a obrazu palubních videokamer, které bude sloužit pro další zkoumání a vývoj FMCW radarů pro zobrazení okolí automobilu.

Klíčová slova: MIMO, FMCW, automobilový radar, FFT, 3D-FFT, MUSIC, MATLAB

Abstract

This bachelor thesis focuses on the description of the principle of operation of FMCW radars and their use for the registration of the radar image of the car environment. It aims to describe general and clear principle of FMCW radars, antenna arrays for the MIMO channel and methods for DOA estimation, input signal processing, calculation of distance from frequency and the principle of velocity detection. The practical part shows a comparison of 3D-FFT and MUSIC methods for searching the azimuth of the radar target and introduces the procedure and the code to combine the radar image and video of on-board video camera, which will be used for further research and development of FMCW radars for the car environment representation.

Keywords: MIMO, FMCW, automotive radar, FFT, 3D-FFT, MUSIC, MATLAB

Obsah

Seznam použitých zkratek

FMCW Frequency Modulated Continuous Wave

FFT Fast Fourier Transform

MUSIC Multiple Signal Classification

SIMO Single-Input Multiple-Output

MIMO Multiple-Input Multiple-Output

IF Intermediate Frequency

VCO Voltage Controlled Oscillator

PLL Phase-Locked Loop

MCC Measure Cycle Count

DOA Direction Of Arrival

ADC Analog-Digital Converter

Seznam obrázků

Seznam ukázek kódu

Úvod

Pro sepsání této bakalářské práce mě vedl požadavek pro názorné nastínění problematiky FMCW[1] radarů, základní porovnání přesnosti metod 3D-FFT[2] a MUSIC pro určení azimutu radarového cíle a implementace programu, který by umožňoval spojení radarového obrazu a obrazu z palubních kamer automobilu.

FMCW radary jsou využitelné pro široké spektrum oblastí, kde je třeba měřit vzdálenost, rychlost a azimut radarového cíle [4]. Právě univerzálnost použití těchto radarů je motivací pro vytvoření základního přehledu jejich funkce a možností, který se v této práci pokusím nastínit. Jsem přesvědčen, že tento přehled v českém jazyce bude přínosem i pro další zájemce o problematiku. V této práci se zaměřuji na obecný popis funkce FMCW radaru a principů, pomocí kterých je možné měřit vzdálenost a rychlost objektu.

Dále potom v teoretickém rozboru popíši použitý systém zpracování vstupních dat. Pro zjištění azimutu je třeba využít anténních řad a metod pro určení DOA[3], kterým se budu věnovat v samostatných kapitolách. Určení azimutu je využito v samotném praktickém jádru práce, které se zabývá umístěním těchto azimutů do kamerových záznamů z testovacího automobilu pro usnadnění validace radarových výsledků z radaru Valeo MRR. V krátké samostatné kapitole také porovnám přesnosti základní metody určení azimutu 3D-FFT a super-resolution metody MUSIC[4]. Pro zpracování dat byl využit matematický software MathWorks MATLAB.

[1] Frequency Modulated Continuous Wave

[2] Fast Fourier Transform

[3] Direction Of Arrival

[4] Multiple Signal Classification

Teoretický rozbor

V této části se budu věnovat popisu základních pojmů a principů souvisejících s FMCW radarem, dále popíši princip anténních řad použitelných pro MIMO[5] kanál a doplním popisem zpracování vstupních dat.

1 FMCW radar

FMCW radar využívá kontinuálního vysílaného signálu v mikrovlnném pásmu (300 MHz - 300 GHz), který je lineárně frekvenčně namodulovaný pilovitým či trojúhelníkovým průběhem nebo jinými variantami modulací - ty potom slouží zejména při měření více objektů zároveň. Tato metoda svým způsobem vkládá do nosného signálu „časovou značku“, frekvence přijímaného signálu oproti frekvenci vysílaného signálu bude posunuta v závislosti na vzdálenosti cíle.

Mezi jeho výhody patří jednoduchý princip činnosti, schopnost měření vzdálenosti a rychlosti objektu, ovšem jeho nevýhodou je nemožnost detekce více objektů, které se pohybují stejnou rychlostí ve stejné vzdálenosti od radaru v různých azimutech. Tato nevýhoda je řešena použitím anténní řady v principu MIMO, pomocí níž lze určit azimut některou z výpočetních metod.

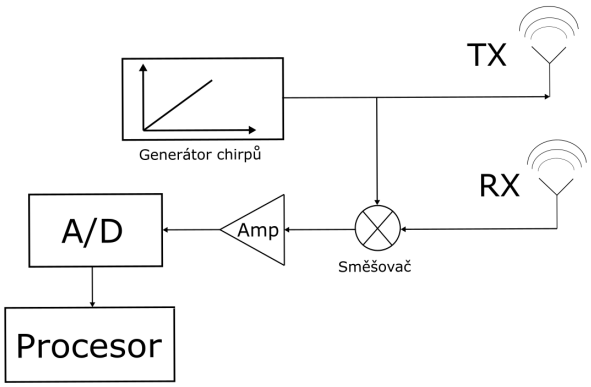
1.1 Popis částí FMCW radaru

Pro vysvětlení složení FMCW radaru jsem zvolil základní blokové schéma, které umožní jednoduchý popis problematiky (obr. 1).

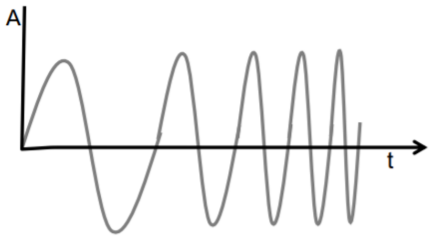
Základním funkčním blokem radaru je generátor chirpů. Chirp je sinusový signál, který je lineárně frekvenčně namodulován v generátoru (obr. 2) a vyslán anténou. Při kontaktu s překážkou je odražen zpět, zachycen anténou (popř. anténami při použití MIMO) a putuje do směšovače společně s chirpem z generátoru [6].

Směšovač je zařízení se dvěma vstupy, do kterých jsou přiváděny směšované signály,

[5] Multiple-Input Multiple-Output

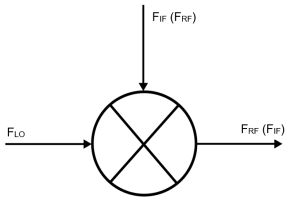


Obrázek 1: Blokový diagram FMCW radaru



Obrázek 2: Časová reprezentace chirpu ([6] str. 4)

a jedním výstupem, ze kterého je signál odebírán (obr. 3). Dle frekvence výstupního signálu rozlišujeme použití jako *up-convertor* (výstupní frekvence je dána součtem vstupních) nebo *down-convertor* (výstupní frekvence je určena rozdílem vstupních).



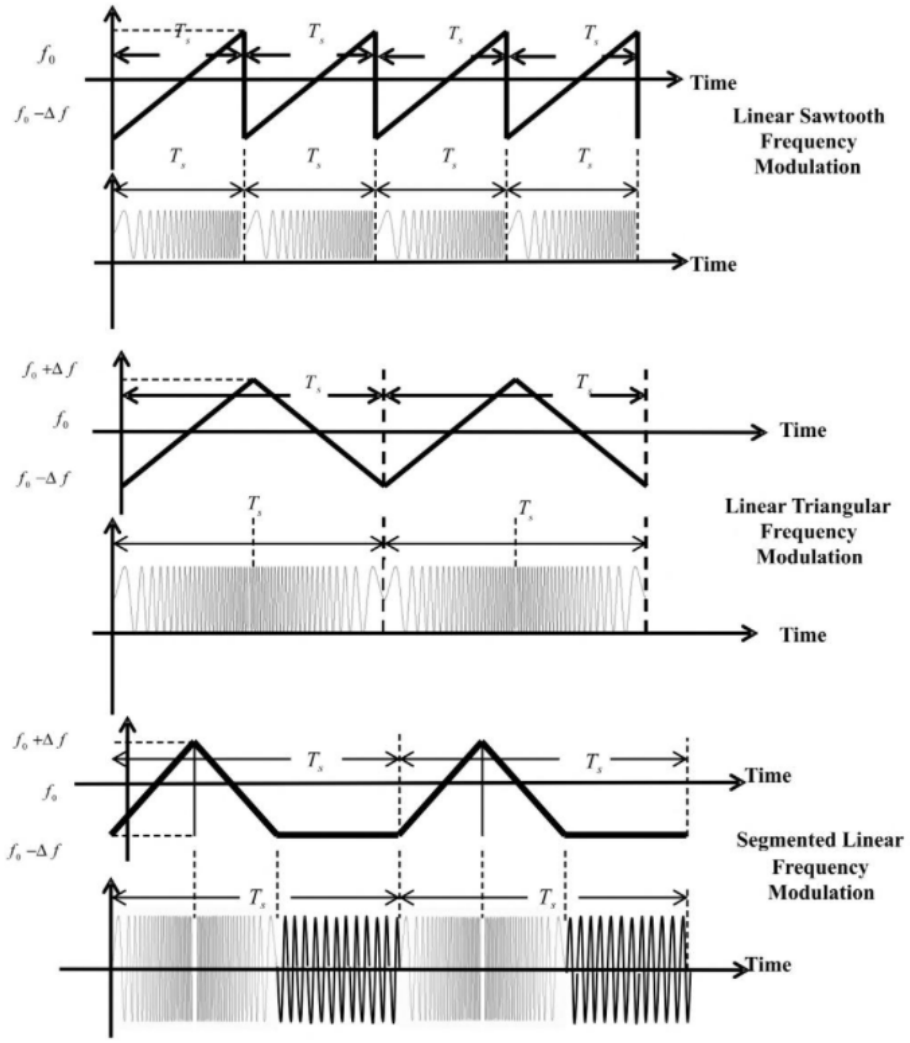
Obrázek 3: Směšovač

Výstupní signál ze směšovače je nazýván IF[6] signál a je zesílen, navzorkován v ADC[7] převodníku a dále zpracován. Mezi jednotlivými bloky se nachází filtry pro zamezení šíření šumu a neužitečných frekvenčních složek.

[6]Intermediate Frequency
[7]Analog-Digital Converter

1.2 Lineární frekvenční modulace

Pro generování chirpu je obvykle využit VCO[8] ve zpětnovazební smyčce PLL[9] (oproti nepříliš využívanému zapojení s Gunnovou diodou uvedenému v [1]). VCO je velmi přesně přeladňován změnou dělícího poměru ve zpětnovazební smyčce. Signál je lineárně frekvenčně modulován pilovým, trojúhelníkovým či jiným průběhem [4] (obr. 4).



Obrázek 4: Ukázky jednotlivých modulací ([5] str. 22)

Podle [11] lze modulaci pilovitým průběhem považovat za tu nejzákladnější, ovšem s nejvíce omezeními. Z povahy pilovitého signálu, který má velmi strmou týlovou hranu pulzu vychází fakt, že se hodí pouze pro použití, kde je třeba měřit vzdálenosti objektů, které jsou statické či je jejich rychlost pohybu zanedbatelná vůči měřené vzdálenosti (např. lodní radary), jelikož při zpětném přijetí vyslaného čirpu není možné určit, zda změna frekvence chirpu koresponduje pouze se vzdáleností, nebo je ovlivněna Dopplerovým posunem.

[8] Voltage Controlled Oscillator

[9] Phase-Locked Loop

Oproti tomu má trojúhelníkový signál tu výhodu, že díky svému tvaru umožňuje určení jak posunu frekvence korespondující měřené vzdálenosti, tak frekvence, danou pohybem cíle. Při zaznamenání odraženého chirpu se jeho průběh nezmění, ovšem frekvenčně se posune a jeho náběžná hrana bude značit součet posunu frekvence na základě vzdálenosti a Dopplerovy frekvence, jeho týlová hrana naopak rozdíl těchto frekvencí. Tímto způsobem je možné je navzájem oddělit.

1.3 Výpočet vzdálenosti z frekvence

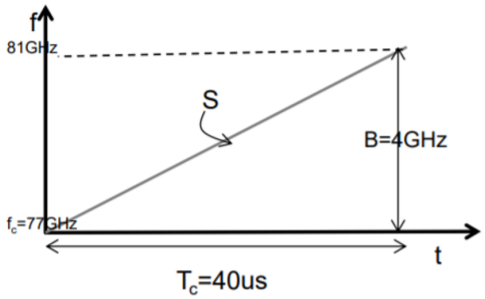
Vzdálenost radarového cíle od radaru je dána vztahem

$$d = \frac{f_{\mathrm{IF}} \cdot c}{2S}, \tag{1}$$

kde f_{IF} je frekvence IF signálu a S je sklon rampy chirpu (obr. 5)

$$S = \frac{B}{T_c}. \tag{2}$$

Frekvence f_{IF} je určena špičkou ve spektru signálu, které je získáno pomocí FFT. Ta je realizována převedením IF signálu do digitální podoby a následným zpracováním v procesoru.



Obrázek 5: f-t reprezentace chirpu ([6] str. 4)

1.4 Detekce rychlosti

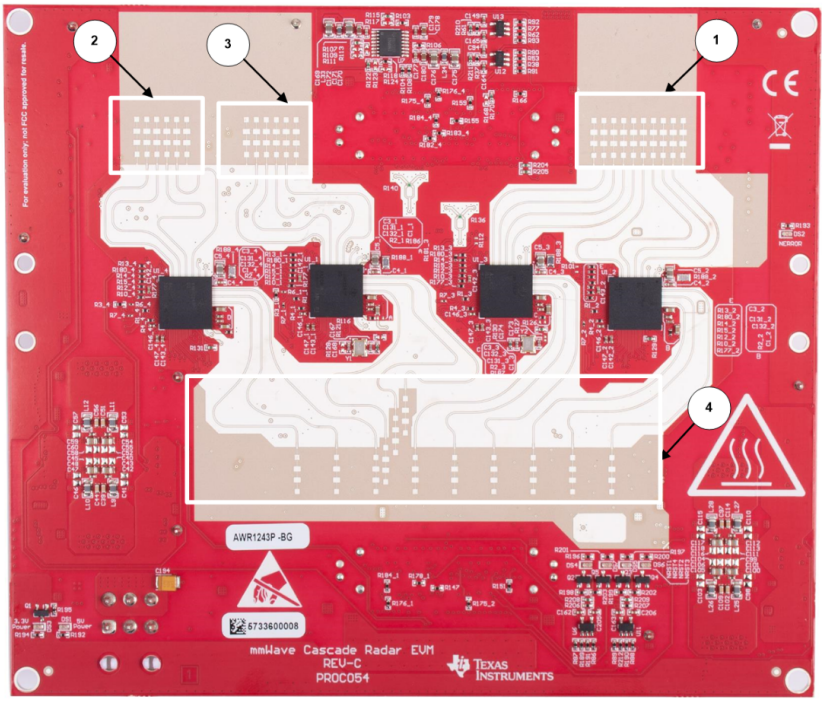
FMCW radar určuje rychlost pohybujících se objektů pomocí vyslání několika chirpů (rámců) za sebou, přičemž následný IF signál podrobí tzv. Doppler-FFT, která určí frekvence, ze kterých je možnost vypočíst hledanou rychlost cíle

$$v = \frac{\lambda\omega}{4\pi T_c}, \tag{3}$$

kde λ je vlnová délka, ω je Dopplerovská frekvence (úhlová rychlost) a T_c perioda chirpů.

2 Princip MIMO

MIMO princip slouží ke zvýšení přesnosti určování azimutu cíle. Jedná se o použití více vysílačů a více přijímačů v uniformní anténní řadě. Oproti SIMO[10], kdy je použita jen jedna vysílací anténa a více antén přijímacích má MIMO výhodu ve snížení počtu potřebných antén (a tedy i celé řady prvků pro zpracování). Na obr. 6 je zobrazeno reálné anténní pole na komerčním modulu MMWCAS-RF-EVM s rozsáhlým MIMO 12×16, tedy 12 TX a 16 RX antén.



Obrázek 6: Ukázkový modul s MIMO anténním polem 12×16 ([9] str. 24)

2.1 Anténní řada

Pro určování azimutů radarových cílů je nutné využít na přijímací straně radaru anténní řadu. Jedná se o skupinu stejných antén v řadě ve vzdálenosti od sebe δ. Tato vzdálenost může být u všech elementů (antén) stejná, ovšem v praxi jsou používány také řady, jejichž elementy mají mezi sebou vzdálenost danou násobky základní vzdálenosti δ. Předpokladem pro funkci takové řady je fakt, že musí být umístěny ve vzdálenosti od vysílacího zdroje tak, aby vlnoplochy elektromagnetického signálu z vysílače bylo možné

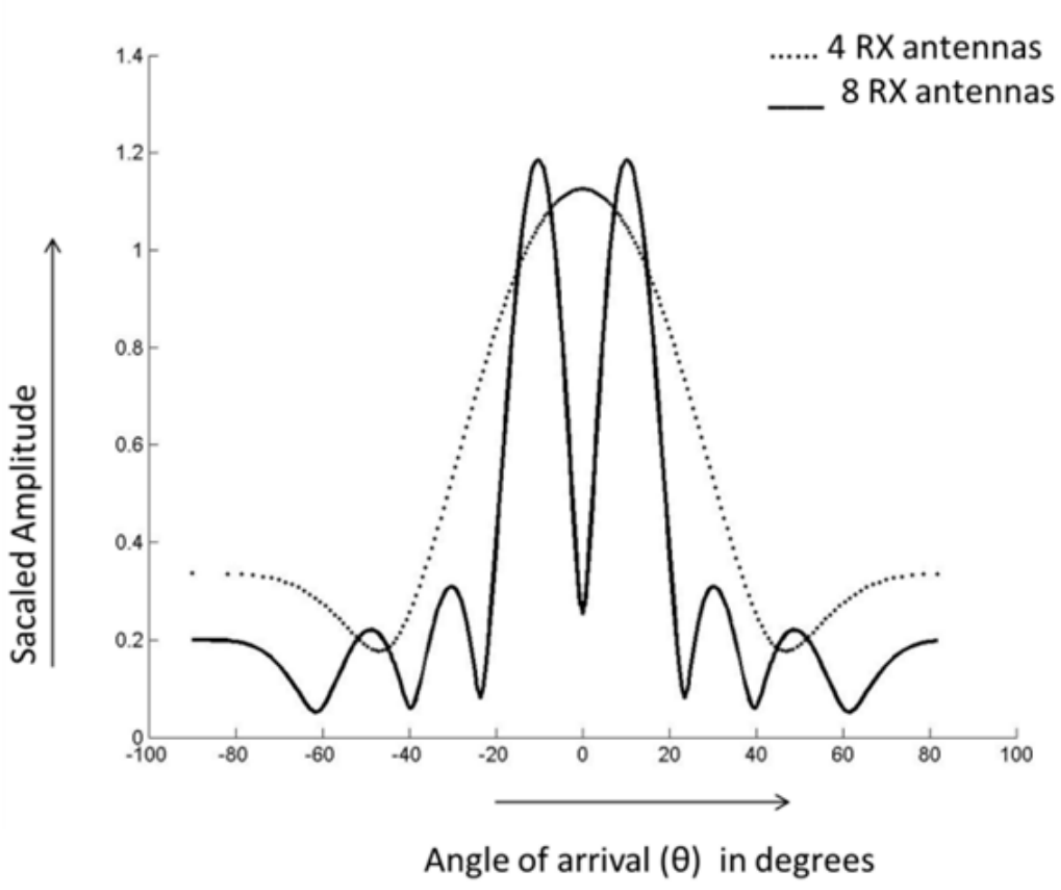
[10] Single-Input Multiple-Output

považovat za rovinné. Potom můžeme celkový signál přijatý elementy zapsat jako

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \dots \end{bmatrix} = \begin{bmatrix} 1 \\ e^{-j\frac{2\pi\Delta}{\lambda}sin\theta} \\ e^{-j2\cdot\frac{2\pi\Delta}{\lambda}sin\theta} \\ \dots \end{bmatrix} s(t) = \begin{bmatrix} 1 \\ e^{-j\mu} \\ e^{-j2\mu} \\ \dots \end{bmatrix} = a(\mu)s(t), \tag{4}$$

kde $a(\mu)$ je tzv. steering vektor.

Minimální rozlišitelný azimut při 3D-FFT zpracování pro SIMO je např. pro 4 RX antény 30° a pro 8 RX antén 15°. Naproti tomu použití MIMO řady o N_{TX} a N_{RX} anténách dává stejné rozlišení jako SIMO řada o $N_{\mathrm{TX}} \cdot \ N_{\mathrm{RX}}$ anténách [7]. Na obr. 7 je naznačena nevýhoda nízké rozlišovací schopnosti radaru, kdy dva cíle mohou splynout do jednoho laloku ve spektru. Azimut se určí ze vzájemného fázového posunu přijatých signálů na jednotlivých anténách řady.



Obrázek 7: Porovnání počtu RX antén ([7] str. 3)

3 Obecný popis metod určení azimutu

Pro určení azimutu radarového cíle při použití technologie MIMO se využívá v dnešní době zejména metoda MUSIC. Dále zde naznačím princip jednoduché metody založené na 3D-FFT, kterou v praktické části využívám pro porovnání s výše zmíněnou metodou MUSIC.

3.1 3D-FFT

Metoda určení azimutu založená na 3D-FFT je jedna z nejméně přesných, ovšem nejjednodušších metod. Určuje tzv. prostorovou frekvenci, podle které se mění fáze přijatých fázorů na jednotlivých anténách v anténní řadě. Dalším přepočtem lze určit spektrum, ve kterém jednotlivé špičky určují hledané azimuty.

3.2 MUSIC

Oproti 3D-FFT je metoda MUSIC jednou z pokročilejších metod, která je numericky a paměťově náročnější, ovšem značně přesnější (dle [3] dosahuje rozlišitelnosti pod 5°). Její dlouhá existence (byla uvedena v roce 1979 [8]) a mnohé výhody, mezi které patří možnost simultánního měření více signálů, vysoká přesnost a schopnost počítání DOA v reálném čase se odráží v její oblíbenosti a vývoji (např. minimum-norm MUSIC).

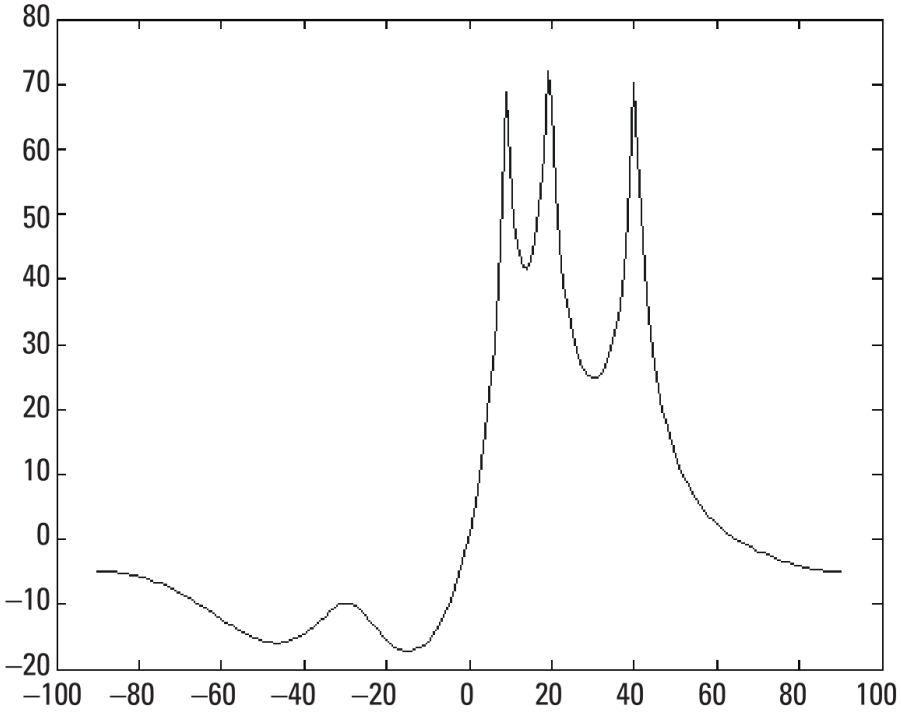
Podle postupu v [2] je nejdříve vytvořena kovarianční matice R_X

$$R_X = E[XX^H], \tag{5}$$

kde X je matice fázorů přijatých signálů a X^H konjungovaná transponovaná matice k X. Následně je kovarianční matice rozložena na dva ortogonální podprostory - signálový V_s a šumový V_n. Poté je vytvořeno MUSIC pseudospektrum

$$P_{MUSIC}(\theta) = \frac{1}{a^H(\theta)V_nV_n^Ha(\theta)}, \tag{6}$$

ve kterém lze najít špičky (obr. 8), které již odpovídají azimutům cílů na daném úhlu θ.

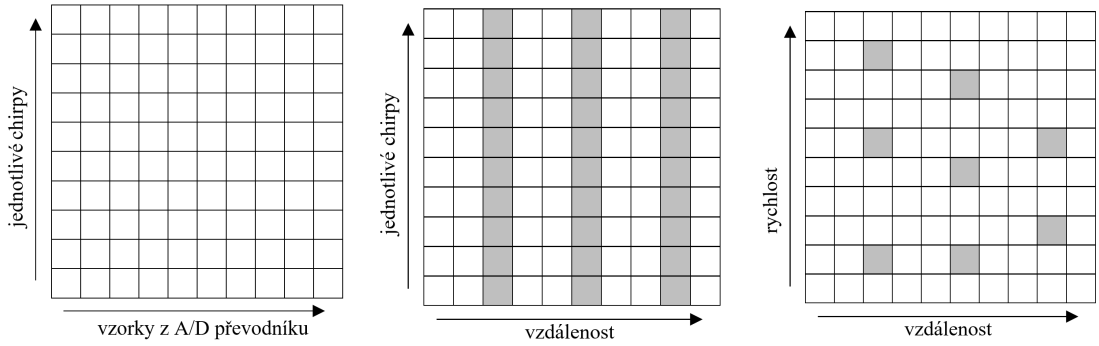


Obrázek 8: Vzorový výstup metody MUSIC se zřetelnými špičkami ([3] str. 60)

4 Zpracování vstupních dat mezifrekvenčního signálu

Tato sekce se bude zabývat jádrem zpracování a výpočtů využitých v této práci.

Klíčovou myšlenkou zpracování navzorkovaných dat mezifrekvenčního signálu je vytvoření matice, jejíž sloupce značí získáné vzorky signálu a řádky jednotlivé chirpy (obr. 9a). Tuto matici máme poté k dispozici pro každý přijímací element anténní řady.



(a) Výchozí stav **(b)** Range-FFT **(c)** Doppler-FFT

Obrázek 9: Schéma zpracování vstupní matice (inspirováno v [6] str. 44)

Tato matice je dále podrobena FFT nejprve přes řádky [10]. Tato operace je nazvána Range-FFT, jelikož slouží k získání tzv. range-binů, které v sobě nesou informaci o vzdálenosti nalezených radarových cílů (obr. 9b). Následně je provedena další FFT, tentokrát přes sloupce - ta je nazývána Doppler-FFT z důvodu, že vytváří tzv. doppler-biny, které

v sobě nesou informaci o rychlosti cíle (obr. 9c).

Ve svém kódu na výslednou matici aplikuji hledání maxima pro získání nejsilnějšího cíle, je získána i jeho rychlost a vzdálenost, tedy jeho fázor. Pro tyto údaje poté zjistím i další fázory z ostatních přijímacích elementů a právě tyto fázory v matici slouží pro určení azimutu. Z nich je poté spočtena autokorelační matice (dle rovnice 5).

Následně je aplikována metoda MUSIC, jejíž vstupy jsou autokorelační matice R dle předchozího odstavce, vzdálenost mezi elementy v anténní řadě δ, umístění elementů v anténní řadě (vektor) a počet cílů, které mají být detekovány.

Praktická část

V této části se budu věnovat nejprve nastínění problematiky a stávajícího řešení. Poté zde popíši způsob měření pro porovnání přesnosti metody založenou na 3D-FFT a metody MUSIC. Následně budu pokračovat k samotnému jádru této práce - softwarovému zpracování mezifrekvenčních signálů, určení azimutů pomocí metody MUSIC a prezentaci výsledných azimutů v záznamech kamer umístěných v automobilu.

5 Současný stav problematiky

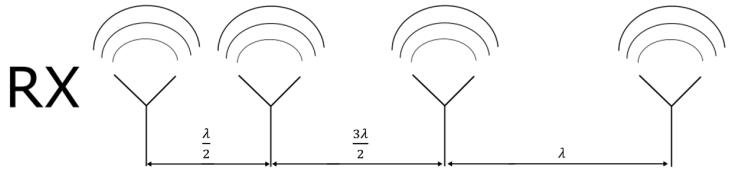
V době vypracování práce existuje specializovaný software pro zpracování signálů z FMCW radarů a použití metod pro určení azimutu, jehož výstupem může být i grafické zobrazení scény okolo radaru. Tento software má ovšem dvě zásadní nevýhody - cenu, pokud nejde o specializovaný software vyvíjený v rámci jediné korporace a uzavřenost pro modifikace, která souvisí obvykle se značnou složitostí jeho kódu.

V návaznosti na tento výčet nevýhod byl vyžádán nástroj, který by v otevřené formě pokryl dílčí oblast zpracování signálů z FMCW radaru, která se zaměřuje na jednoduché zpracování signálu, použití již hotové MUSIC metody a zejména intuitivní zobrazení značek do videoobrazu tam, kde se nacházejí radarové cíle. To pak umožňuje validaci výsledků ostatních nástrojů a také usnadňuje získání celkového přehledu nad scénou.

6 Porovnání 3D-FFT a MUSIC

Tato část má za cíl obecně popsat a porovnat dvě metody určení azimutu - 3D-FFT a MUSIC při měření pohybujícího se koutového odražeče.

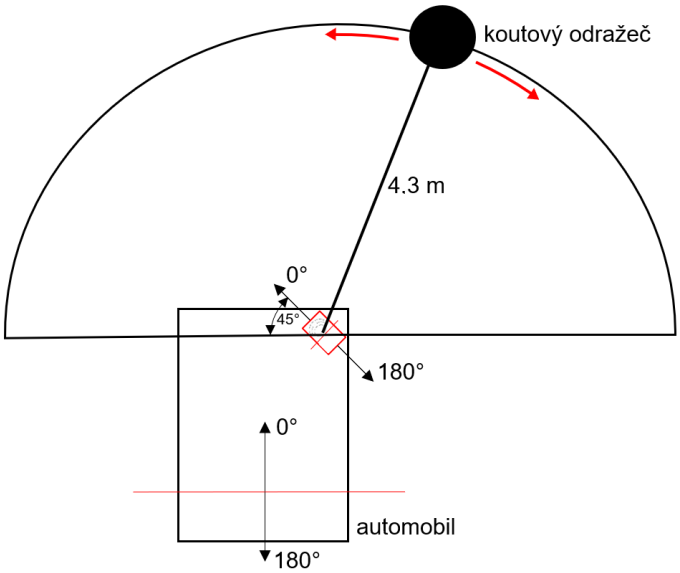
Pro porovnání metod určení azimutu cíle byla naměřena data při detekování koutového odražeče, který se pohyboval po půlkružnici před radarem. Měřicí radar se skládal z přijímací anténní řady čtyř antén s rozestupy $\frac{\lambda}{2}$, $\frac{3\lambda}{2}$ a λ (obr. 10).



Obrázek 10: Uspořádání anténní řady

6.1 Popis uspořádání měření

Měřicí radar byl umístěn v pravém blatníku testovacího automobilu pod instalačním úhlem 45° (viz obr. 11), kde střed zadní osy automobilu byl středem souřadného systému. Nulový azimut byl směrem dopředu v pravotočivém souřadném systému, koutový odražeč se tedy pohyboval ve vzdálenosti 4,3 m od radaru od azimutu 270° do azimutu 90°.



Obrázek 11: Uspořádání měřicí soustavy

6.2 Vstupní data

Část naměřených dat (tzv. detection list) byla v souboru ve formátu *csv* (obr. 12). Hlavní sloupce značily:

- *Azimuth* - azimut vůči radaru
- *LatPos* - souřadnice „šířka“ vůči zadní ose automobilu
- *LonPos* - souřadnice „délka“ vůči zadní ose automobilu
- *Azimuth_car_rad* - azimut vůči automobilu

Druhá část naměřených dat byla v souboru formátu *mat* a obsahovala komplexní data z každé RX antény, např. úhel vůči radaru, rychlost pohybu otáčení ramene s odražečem atd.

	A	B	C	D
1	Azimuth	LatPos_m	LongPos_m	AzimuthCar_rad
2	0.434587	2.383529	6.753958	0.3392635
3	0.1134464	1.624608	10.56016	0.1526464
4	-0.1623156	-0.5832579	11.81437	-0.04932846
5	-0.1169371	-0.5008867	14.44846	-0.03465326
6	0.450295	2.355211	6.559902	0.3446978
7	0.4380777	3.19575	8.456974	0.3612962
8	0.08203048	1.402816	10.6447	0.1310303
9	-0.132645	-0.3010309	11.62333	-0.02589307
10	-1.438151	-6.260301	4.07996	-0.9932133
11	0.1518436	0.4042235	11.20103	0.03607241
12	-0.05759587	-1.292141	11.9383	-0.1078152
13	-1.689479	-12.68543	1.933388	-1.41955
14	-1.954769	-12.40853	-1.340364	-1.678399
15	-0.7714356	-10.36396	13.18892	-0.6660272
16	-1.804671	-15.01625	-0.03650427	-1.573227
17	-1.993166	-16.17834	-3.561649	-1.787489
18	-1.993166	-18.11303	-4.431134	-1.810722
19	-1.802925	-20.52288	-1.312002	-1.634638
20	-1.692969	-22.26081	0.715574	-1.538662

Obrázek 12: Snímek části vstupních dat pro měření s koutovým odražečem

6.3 Zpracování vstupních dat

Zpracování dat zajišťuje krátky kód, který na svém začátku definuje potřebný pracovní prostor, poté načítá vstupní data z *mat* souboru, která odpovídají 2D-FFT. Jsou vytvořeny vektory, které budou ve výsledném grafu představovat x osu. Následně jsou v každém rámci zpracována data 3D-FFT a MUSIC metodou.

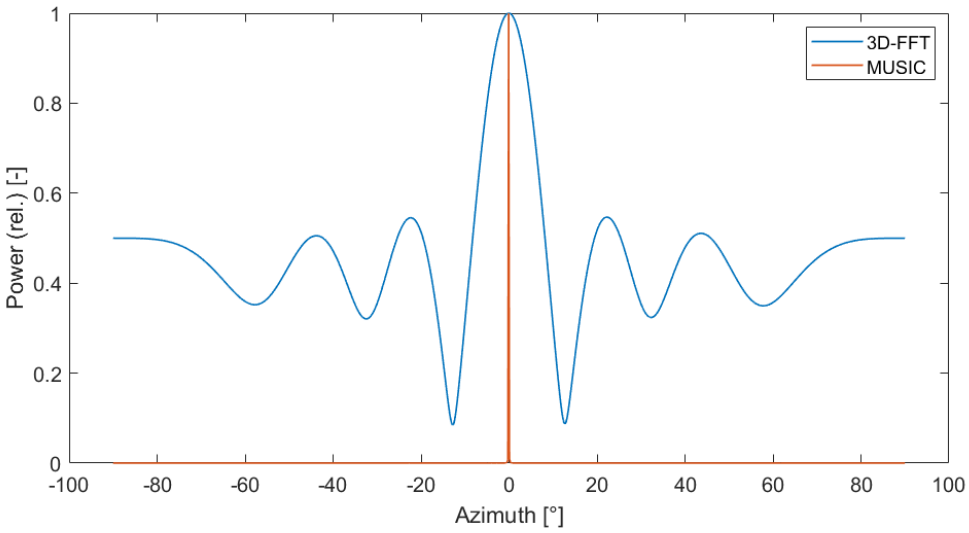
Ukázka kódu 1: Zpracování dat 3DFFT a MUSIC metodou

```
for iFrame = 1:10:N
    % PCur – complex matrix with input data from 2D–FFT
    targetAntPhasors = (PCur(iFrame,:)).'; % transposition for MUSIC

    % zero addition for 3D–FFT – we have field of 32 antennas but data are only from 4 RX
        antennas at 1,2,5,7 indexes
    targetAntPhasorsInterp = zeros(1,nFFTDOA);
    targetAntPhasorsInterp(1) = targetAntPhasors(1);
    targetAntPhasorsInterp(2) = targetAntPhasors(2);
    targetAntPhasorsInterp(5) = targetAntPhasors(3);
```

```
    targetAntPhasorsInterp(7) = targetAntPhasors(4);

    % use of 3D-FFT and saving output for later rendering
    DOAFFTData = fftshift(abs(fft(targetAntPhasorsInterp, nFFTDOA)));
    hLineFFT_DOA.YData = DOAFFTData/max(DOAFFTData);

    % MUSIC
    % calculation of the autocorrelation matrix of recieved phasors
    R = targetAntPhasors*targetAntPhasors';
    % use of external function MUSIC.m
    [~, P_MN] = MUSIC(R,0.5,antennaPos,MUSICTheta,1);
    % saving output for later rendering
    hLineMUSIC_DOA.YData = P_MN/max(P_MN);

    % outputs rendering
    plot(FFTTheta/pi*180, abs(hLineFFT_DOA.YData))
    hold on
    plot(MUSICTheta/pi*180, fliplr(abs(hLineMUSIC_DOA.YData)))
    hold off
    legend('FFT', 'MUSIC');

    pause(0.1)
end
```

6.4 Porovnání přesnosti obou metod

Z obr. 13 lze jednoznačně usoudit vyšší přesnost metody MUSIC. Šířka hlavního laloku 3D-FFT metody je cca. 26°a značí nízkou rozlišovací schopnost, která může zapříčinit detekování dvou blízkých objektů jako objekt jeden. Pro metodu MUSIC vyšla šířka hlavního laloku asi 4°. Lze tedy určit, že metoda MUSIC je při tomto měření 6,5× přesnější, ovšem pro jediný koutový odražeč se zdají obě metody použitelné.



Obrázek 13: Graf porovnání 3D-FFT a MUSIC

7 Registrace radarového a video obrazu

Úkolem této části bylo zpracovat vstupní video tak, aby přes detekované cíle bylo vloženo ID, které značí vzdálenost cíle a usnadňuje kontrolu správné detekce cíle při pouhém sledování videa bez uživatelské kontroly radarových dat.

K řešení zadání byl využit software MathWorks MATLAB R2020b. Vzhledem k obecné nedostupnosti a ceně toolboxů jsem při psaní kódu kladl důraz na použití vlastních funkcí tak, aby žádný toolbox nebyl použit.

7.1 Vstupní data

Pro tuto část byla použita data z radaru Valeo MRR s rozlišovací schopností ve vzdálenosti cca. 30 cm umístěném v jedoucím automobilu a záznamy z palubních kamer.

Vstupní data z radaru byla v hierarchickém formátu *hdf5* s velikostí 2 GB, ve kterém byly uloženy dva datové zdroje pro každý senzor. Jedná se o vektor MCC[11], ve kterém je uložen sled čísel MCC - jsou to čísla radarových cyklů s opakovací frekvencí 30 FPS, lineárně narůstající během sběru dat. V mém programu slouží pro synchronizaci radarových cyklů a zpracování video snímků. Dále se jedná o navzorkovaná data IF signálu z ADC ze všech antén a senzorů, která jsou dělena pro každý ze senzorů zvlášť. Data z radaru se skládala ze 128 jednotlivých chirpů, kdy z každého chirpu bylo odebráno 1024 vzorků.

Záznamy palubních kamer se nachází ve společném videu se snímkovací frekvencí 60 Hz, které je výstupem programu Valeo Development Suite 3.5.1.0. Ve videu také běží MCC radarů, které slouží k synronizaci videozáznamů a běžícího programu - nárůst MCC o jedna proběhne přesně během dvou snímků videa.

7.2 Řešení rozdílných úhlů záběru

Pro správné umístění vloženého ID přes detekovaný cíl v obraze kamery bylo potřeba implementovat přepočet azimutu cíle z radaru na souřadnice v pixelech v obrazu. V tomto směru jsem se omezil pouze na horizontální souřadnici v obraze, jelikož vertikální posun by vyžadoval měření elevace, které senzory neumožňují (vertikální souřadnice byla tedy zvolena ve středu osy v obraze).

Z umístění kamer a senzorů je zřejmé, že úhel záběru ani jeho osa není pro kamery ani senzory společná. Proto jsem k problému přistoupil tak, že pro každou z kamer jsem vybral dvojici senzorů, jejichž záběr měl průnik se záběrem kamery a právě pro tyto senzory poté byly hledány azimuty cílů. Ty byly následně porovnány se záběrem kamery a pouze úhel

[11] Measure Cycle Count

těch, které do záběru náležely byl přepočítán do souřadnic ve snímku videa a zobrazen.

7.3 Umístění kamer a radarových senzorů

V testovacím automobilu se nachází čtyři webkamery sloužící pro monitorování okolí automobilu (obr. 14). Jedná se o běžné webkamery se standardním rozlišením a úhel záběru byl experimentálně určen cca. 66° (tento parametr není výrobcem uváděn).

(a) Obraz přední kamery

(b) Obraz zadní kamery

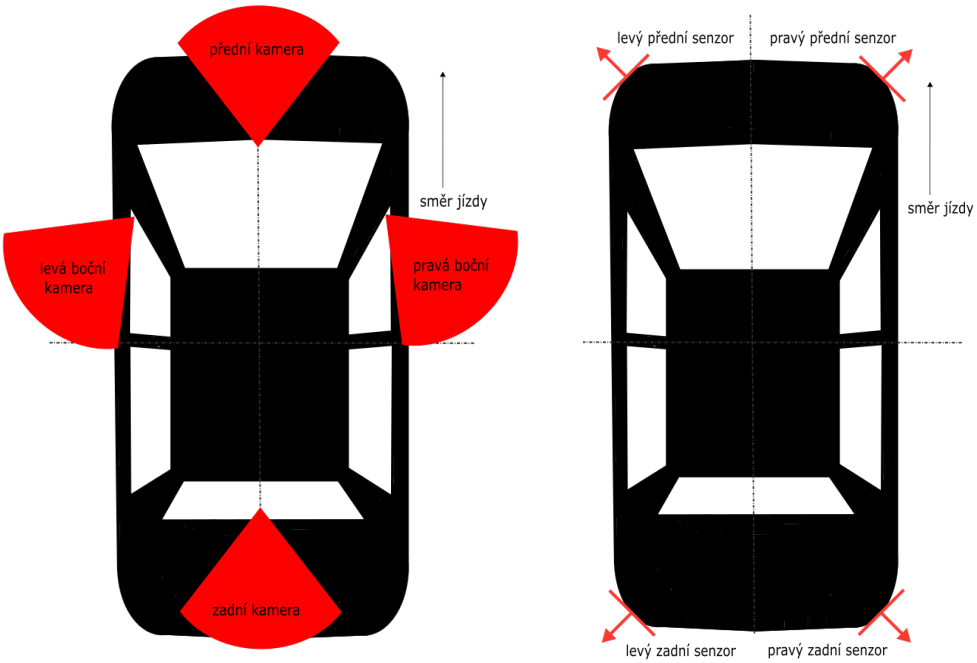
(c) Obraz levé kamery

(d) Obraz pravé kamery

Obrázek 14: Ukázky kamerových obrazů

První kamera je umístěna za předním sklem a slouží pro monitorování situace před automobilem. Obdobně se nachází kamera za zadním sklem pro pohled za automobil. Obě tyto kamery sdílí osu natočení, která se shoduje s podélnou osou automobilu. Dále se nachází dvě kamery po stranách automobilu, které jsou natočeny o 45° od příčné osy automobilu, směrem k jeho zadní části a slouží pro monitorování situace na stranách a zejména mrtvého úhlu (obr. 15a).

Čtyři radarové senzory jsou umístěny v blatnících, jsou natočené o 45° od podélné osy automobilu a jejich úhel záběru je určen jako 180° (obr. 15b).



(a) Umístění kamer **(b)** Umístění senzorů

Obrázek 15: Umístění kamer a senzorů v automobilu

7.4 Obecné schéma kódu

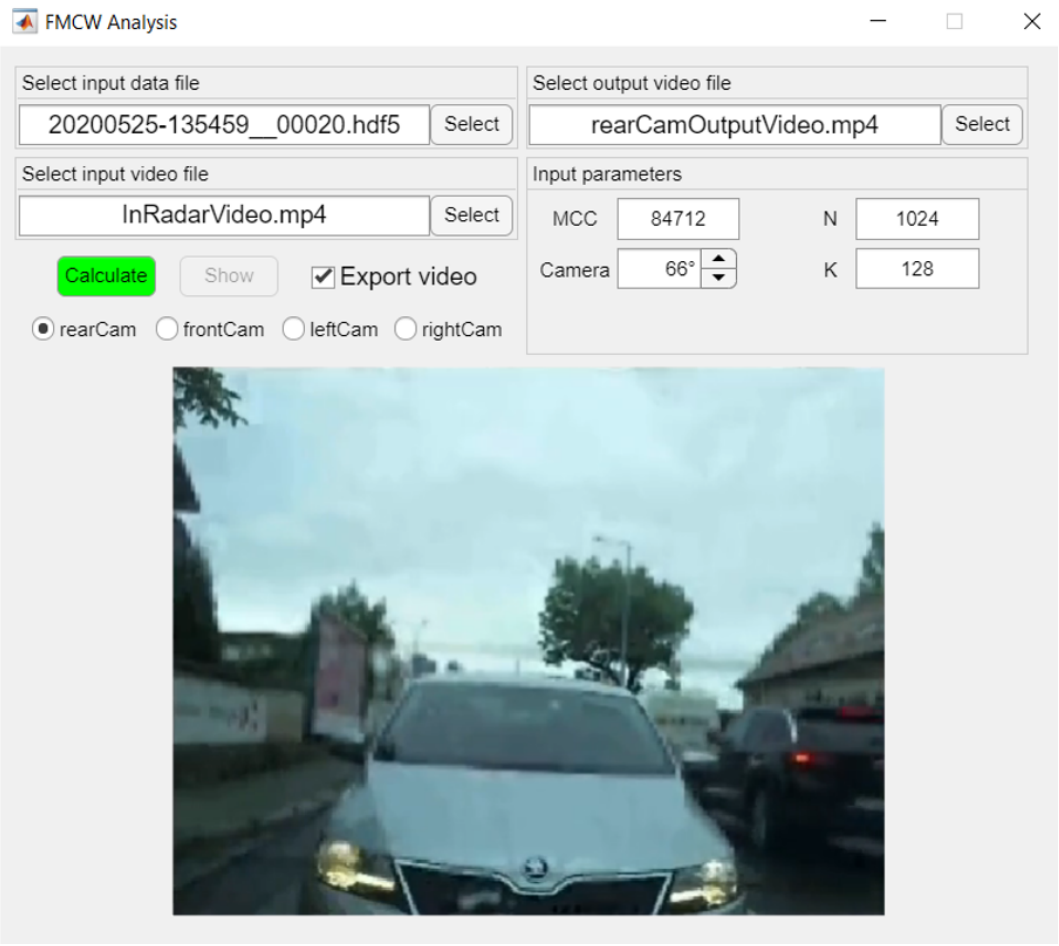
Kód je dělen do hlavní části a několika dílčích externích funkcí.

- *GUI.m* - slouží jako hlavní část kódu a obsahuje GUI
  - *Calculation.m* - zajišťuje veškeré potřebné vlastní a dílčí výpočty
    - *AntPhasors.m* - výpočet fázorů
    - *FFT_2D.m* - příprava pomocí 2D-FFT
    - *MUSIC.m* - vlastní funkční celek metody MUSIC
  - *Show.m* - zajišťuje zobrazení grafických výsledků
    - *NumClass.m* - grafická knihovna číslic

7.5 Grafické rozhraní

Na obrázku 16 je ukázáno GUI používané pro usnadnění zadávání vstupních uživatelských parametrů a celkové ovládání kódu.

Ve vstupních zadávacích polích umožňuje výběr cesty k hierarchickému souboru obsahující vstupní data z A/D převodníku a cesty ke vstupnímu videu vytvořeného v ovládacím softwaru příslušného radaru. Dále je možné vybrat cestu k výstupnímu videu, pokud bude zvolen export videa.



Obrázek 16: Ukázka grafického rozhraní kódu

Panel vstupních parametrů potom umožňuje výběr vstupních parametrů:

- *MCC* - značí pořadové číslo MCC, kterým začíná vstupní video
- *Camera* - určuje úhel záběru použité webkamery v rosahu 0° až 180°
- *N* - počet vzorků na jeden chirp ve vstupních datech
- *K* - počet chirpů v jednom rámci

Vstupní parametry mají implicitní hodnoty nastavené pro použití s testovacími daty.

Pod tlačítky se nachází přepínače, kterými je možné zvolit jednu ze čtyř kamer, sledujících okolí automobilu, na kterou se následující výpočet a zobrazení zaměří.

Pro spuštění výpočtu po nastavení veškerých vstupních hodnot slouží tlačíko **Calculate**, po jehož aktivaci se objeví tzv. progress-bar informující uživatele o průběhu výpočtu.

Po úspěšném běhu tlačíko **Calculate** změní barvu a následně je možné pokračovat stisknutím tlačítka **Show** pro spuštění přehrávání videa, do kterého jsou již vloženy značky,

které v reálném čase kopírují pozici zjištěných radarových cílů. Před spuštěním přehrávání videa si může uživatel zvolit, zda má být přehráváné video s vloženými značkami exportováno do předem určeného souboru.

7.6 Princip funkce kódu

Následující sekci věnuji podrobnějšímu popisu dílčích částí kódu. Budu zde popisovat přístup k tvorbě GUI, výpočtů azimutů a nakonec výslednému zobrazování, popř. exportu videa.

GUI.m

GUI.m slouží k tvorbě grafického rozhraní programu. Nejprve jsou definovány základní grafické struktury a jsou do nich vloženy grafické elementy (tlačítka, editovatelná pole atd.).

Ukázka kódu 2: Základní grafické struktury a elementy

```
%% Structures
hInputDataPanel = uipanel(hFig,'Position',[10, 490, 305, 50],'Title','Select input data file');
hAx = uiaxes(hFig,'Position',[80, 5, 461, 357],'XLim',[0 461],'YLim',[0 357],'XColor','none','YColor','none');
%% Elements
hInputDataEdit = uieditfield(hInputDataPanel,'Position',[2, 2, 250, 25],'FontSize',15,'HorizontalAlignment','center');
hCamSpinner = uispinner(hInputParamsPanel,'Position',[55, 40, 75, 25],'Value',66,'Limits',[0 180],'ToolTip','angle of camera view [0 —180 ]','ValueDisplayFormat','%d ');
```

Následně jsou definovány tzv. callbacks, které zajistí spuštění příslušných funkcí každému z elementů, na základě jeho změny, stiskutí a podobně.

Ukázka kódu 3: Callbacks

```
%% Callbacks
hOutVidButt.ButtonPushedFcn = @(src,event)pressOutVid();
hCalcButt.ButtonPushedFcn = @(src,event)pressCalcButt();
hShowButt.ButtonPushedFcn = @(src,event)pressShowButt();
```

Tyto funkce slouží pro ovládání samotného grafického prostředí, např. změnu barev tlačítek či jejich zpřístupnění uživateli, ale i kontrolu správnosti vstupních dat, která je nedílnou součástí kódu z hlediska prevence proti pádům či nesprávnému chování.

Z výpočetního hlediska jsou zde podstatné funkce, která jsou spuštěny po aktivaci tlačítka **Calculate** (funkce *pressCalcButt*) a následně tlačítka **Show** (funkce *pressShowButt*). Funkce *pressCalcButt* zajišťuje kontrolu vstupních dat, načtění vstupních parametrů a jejich předání externí funkci *Calculation.m*.

Obdobnou funkci má funkce *pressShowButt*, která předává načtené parametry funkci *Show.m*.

Ukázka kódu 4: Funkce *presCalcButt*

```
function pressCalcButt()
    set(hCalcButt,'BackgroundColor',[0.96 0.96 0.96]);
    if InDataPathName
        progress = uiprogressdlg(hFig,'Title','Please Wait','Message','Calculating DOA from
            input data');
        progress.Value = 0;

        fullHDF5 = fullfile(InDataPathName,InDataFileName);

        K = get(hKEdit,'Value');
        N = get(hNEdit,'Value');

        CamSpinner = hCamSpinner.Value;

        [resultAngle, MCC, rangeBins] = Calculation(progress,fullHDF5,K,N,selectedCamera);
        close(progress)

        set(hShowButt,'Enable','on');
        set(hCalcButt,'BackgroundColor','g');
    else
        uialert(hFig,'Please specify input data file','Invalid File');
    end
end
```

Calculation.m

Tato funkce je přímo zodpovědná za veškeré výpočty pro zjištění azimutů cílů, ke kterým využívá dále funkce *FFT_2D.m*, *MUSIC.m* a *AntPhasors.m*.

Na svém začátku zkontroluje, pro jakou kameru v automobilu se budou výpočty provádět a nastaví k výpočtu dané senzory a jejich kalibrace. Poté následuje vytváření pomocných proměnných a již samotný výpočet, který se nachází ve *for* cyklu vnořené funkce *CalcDOA*.

Ukázka kódu 5: Funkce *CalcDOA*

```
function [resultTemp,rangeBinsTemp] = CalcDOA(maxK,SpdoCorrection,prevProg)
    resultTemp = zeros(1,maxK);
    rangeBinsTemp = zeros(1,maxK);
    for nFrames=1:maxK  % 255
        progress.Value = prevProg + nFrames/maxK/2;

        % 128*1024*4
        measData = flip(permute(reshape(hdf5_data1.Beams{nFrames,1}.MeasData{beamId,1}, nrRx
            ,N,K),[3 2 1]),1);

        Fft2D_permuted = FFT_2D(measData,SpdoCorrection,PRS_h1D,PRS_h2D,K,N,nrRx);
```

```
11
12          [targetAntPhasors,targetRangeBin] = AntPhasors(Fft2D_permuted,N);
13          rangeBinsTemp(nFrames) = targetRangeBin;
14          % MUSIC
15          % autocorrelation matrix of recieved phasors
16          R = targetAntPhasors*targetAntPhasors';
17
18          % we use Minimum Norm MUSIC method, which is encrypted in *.p file, in
19          % *.m file is only header with input description
20          [~, P_MN] = MUSIC(R,0.5,antennaPos,MUSICTheta,1);
21          locs = find(P_MN==max(P_MN));
22          resultTemp(:,nFrames) = locs;
23      end
24  end
```

V každém kroku tohoto cyklu jsou načtena data ze senzorů pro jeden rámec, která jsou dále podrobena 2D-FFT (funkce *FFT_2D.m*) a následně výpočtu fázorů, ze kterých je zjištěna pozice nejsilnějšího doppler-binu a range-binu (funkce *AntPhasors.m*).

Z výsledné matice je vytvořena autokorelační matice přijatých fázorů, která je jedním z nejdůležitějších vstupních parametrů pro funkci *MUSIC.m*.

Po proběhnutí metody MUSIC je zjištěna poloha její hlavní špičky a ta je uložena pro další zpracování.

MUSIC.m

Tato funkce je jádrem implementace metody MUSIC, která mi byla poskytnuta v zašifrovaném formátu vedoucím práce, jenž je jejím autorem.

Vstupy funkce jsou samotná autokorelační matice přijatých fázorů, vzdálenost jednotlivých antén od sebe a dále pak vektor úhlů, pro které bude metoda azimuty testovat a počet radarových cílů pro detekci.

Ukázka kódu 6: Hlavička funkce *MUSIC.m*

```
1  %% Implementation of MUSIC method for DOA estimation.
2  % Implemented according to Introduction to Direction-of-Arrival Estimation,
3  % Chen, Gokeda, Tu, Artech House, 2010.
4  %
5  %  INPUTS
6  %   R: Autocorrelation matrix of received data, complex double [M x M]
7  %   Delta: pitch between individual antennas in LUA relative to center
8  %          frequency, 0.5 typically, double [1 x 1]
9  %   antennaPos: positions of individual antennas in Delta units, for LUA it
10 %               would be (0:M-1).', or [0; 1; 2; ...; M-1], double [M x 1]
11 %   theta: for which azimuths the Music spectrum will be computed, rad,
12 %          double [n x 1]
13 %   d: number of targets for detection
14 %
15 %  OUTPUTS
```

```
%   P: spectrum computed using classical MUSIC, double [n x 1]
%   P_MN: spectrum computed using Minimum Norm MUSIC, double [n x 1]
%
%    2020, Viktor Adler, CTU in Prague, adlervik@fel.cvut.cz
```

Show.m

Funkce *Show.m* zajišťuje zpracování videa a vložení ID nalezených cílů, jeho zobrazení uživateli pro náhled a případný export videa do určeného souboru.

Na začátku funkce je zjištění použité kamery a transformace vypočtených azimutů od jednotlivých senzorů na základě jejich natočení vůči ose kamery a vybrání azimutů, které se shodují s jejím maximálním úhlem záběru.

Následně je použit cyklus *while*, který pro každý snímek videa zajišťuje vyříznutí správného okénka s pohledem kamery, vložení ID na správné místo a export snímku do souboru.

Ukázka kódu 7: Hlavní část funkce *Show.m*

```
while hasFrame(videoFileReader)
    myMCC = myMCC + 1;
    % 1 MCC per 2 frames
 defaultFrame = readFrame(videoFileReader);
    readFrame(videoFileReader);

    % cuts from original video
    selCam = defaultFrame(84:440,lCut:rCut,:);     % 356x460 pixels

    if any(myMCC == MCC)
        if counter < length(resultPixelCleaned)
            for id=1:resultPixelRows % id of peak
                id_digNum = max(ceil(log10(abs(rangeBins(id,counter)))),1); % num of digits
                    in id
                id_vect = num2str(rangeBins(id,counter)) - '0'; % vector from id
                if ~isnan(resultPixelCleaned(id,counter))
                    if ~isnan(rangeBins(id,counter))
                    % rear camera
                         for digNum=1:id_digNum
                            if resultPixelCleaned(id,counter)+digNum*9 <= 462
                                selCam(150:150+9,...
                                    resultPixelCleaned(id,counter)+(digNum-1)*9:...
                                    resultPixelCleaned(id,counter)+digNum*9,:)...
                                    = Nums.pickNum(id_vect(digNum));
                            end
                        end
                     end
                end
            end
            counter = counter + 1;
        end
```

```
    end

    % display video frame to screen
    set(videoFig,'CData',flip(selCam));
    % if needed, export video to a file
    if Checked
        writeVideo(v,selCam);
    end
    pause(0.001);
end
```

Vkládané ID značí index sloupce azimutu cíle v matici přijatých fázorů, který značí vzdálenost cíle v násobcích rozlišovací schopnosti radaru ve vzdálenosti.

Pro zamezení použití toolboxů v kódu je vytvořena třída *NumClass.m*, ve které jsou definovány matice číslic, které jsou pak dle daného indexu range-binu naskládány za sebe, aby vytvořily dané číslo.

Po skončení běhu funkce *Show.m* může uživatel program ukončit či například změnit vstupní parametry, kameru a znovu spustit výpočet a export videa pro další informace.

7.7 Náhled do zpracování

Při běhu kódu dochází k mnoha výpočtům a pro zobrazení a lepší kontrolu dílčích sekcí kódu jsem využil tvorbu kontrolních grafů.

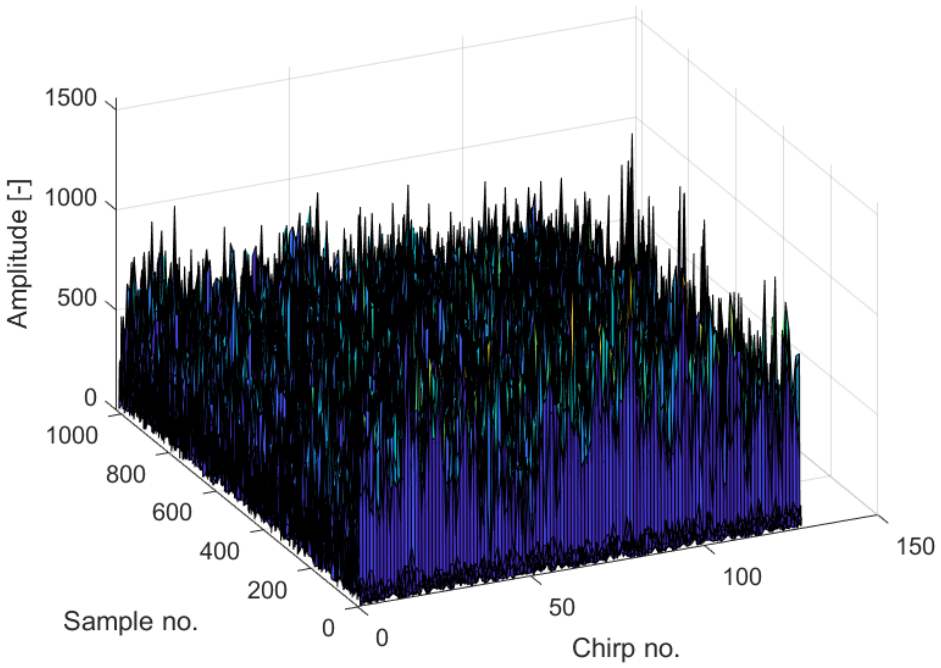
Vstupní signál a použití FFT

V této části předvedu jednotlivé kroky zpracování od IF signálu po výsledné MUSIC spektrum. Tato ukázka IF signálu a několikanásobného použití FFT náleží 169. rámci na první anténě, kdy výsledný azimut a range-bin odpovídá ID 39 na obr. 21.

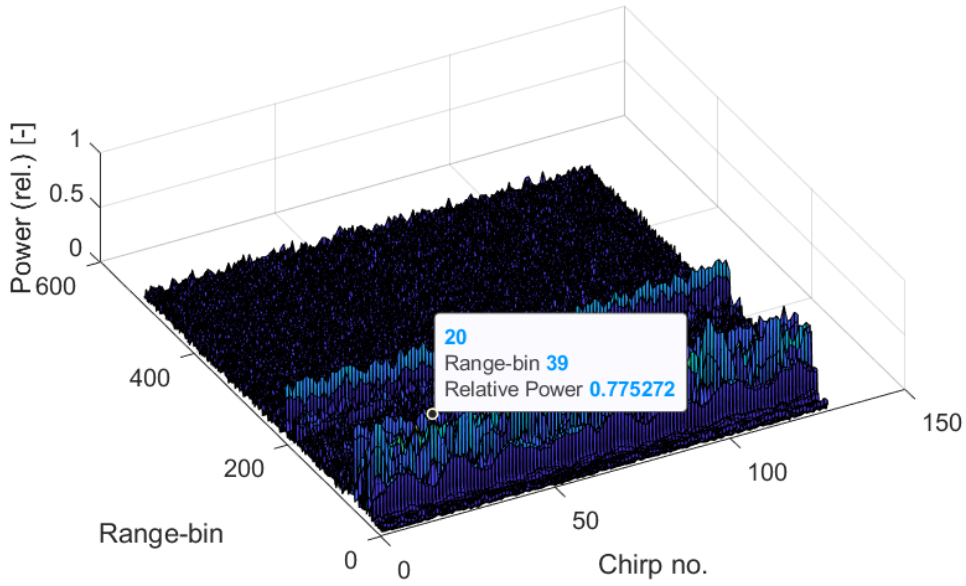
Na obr. 17 se nachází graf modulu komplexního IF signálu. Je zde patrné, že se skládá z 1024 vzorků v každém ze 128 chirpů.

Dále je na obr. 18 zobrazena absolutní hodnota komplexního spektra IF signálu po aplikaci FFT přes jednotlivé sloupce, tj. vzorky v každém chirpu. Takto vznikl tzv. range profil, ve kterém jsou viditelné špičky v range-binech (při pohledu shora jako na obr. 9b).

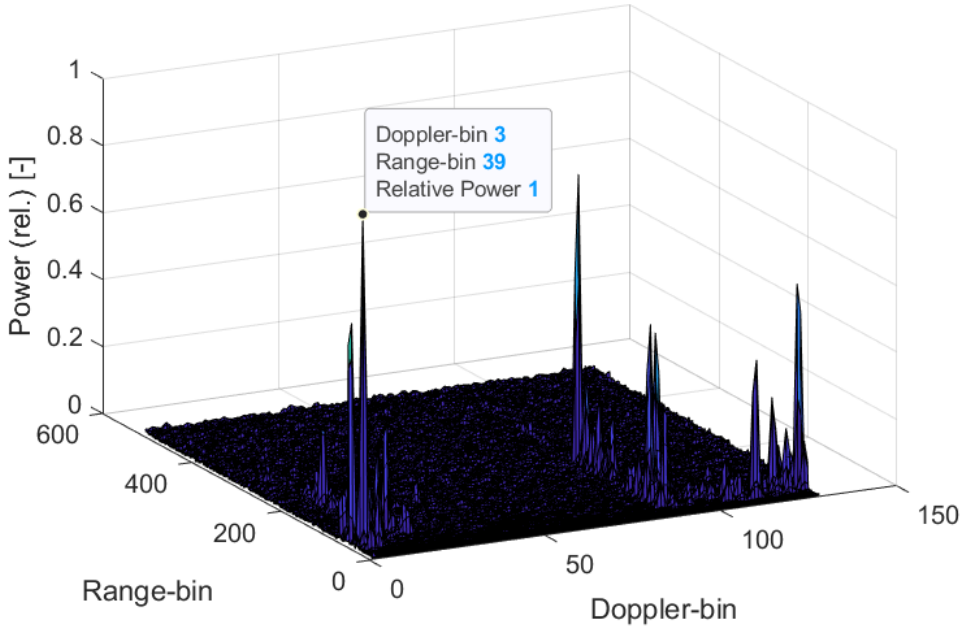
Po aplikaci FFT přes řádky (tj. chirpy) vzniká tzv. doppler profil na obr. 19, kde jsou viditelné již jednotlivé špičky s udaným range- a doppler-binem (při pohledu shora jako na obr. 9c).



Obrázek 17: Časový průběh vzorového IF signálu



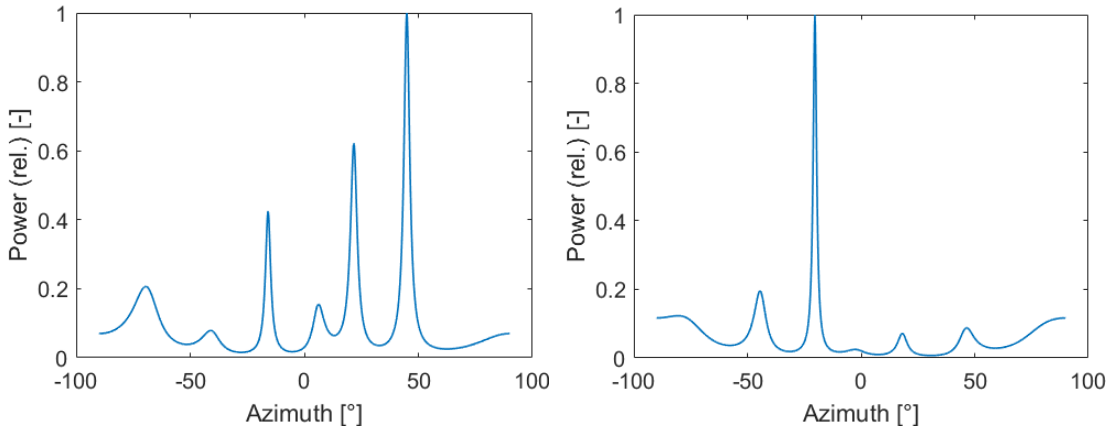
Obrázek 18: Modul spektra Range-FFT



Obrázek 19: Modul spektra Doppler-FFT

Vyhledání fázorů a použití metody MUSIC

Při následném detekování nejsilnějšího cíle, nalezení jeho range- a doppler-binu a vyhledání odpovídajících fázorů na ostatních anténách (funkce *AntPhasors.m*) lze použít metodu MUSIC. Na obr. 20a je její výstupní spektrum, z něhož je vybrána špička (tedy cíl), nalezen jeho příslušný range-bin a úhel, který je poté přepočítán z hlediska perspektivy a umístěn do snímku na obr. 21 jako ID 39. Následný ID 26 je zpracován obdobně, jedná se o výstup zpracování rámce č. 170 (obr. 20b).



(a) Spektrum pro cíl ID 39 **(b)** Spektrum pro cíl ID 26

Obrázek 20: Vzorová spektra MUSIC pro dva cíle na obr. 21

První cíl s ID 39 značí úhel 45° z pohledu levého předního radaru - ten je pak přepočítán pro kameru na 0°, čehož je možné si všimnout na snímku 21, kde se ID nachází uprostřed horizontální osy.

Druhý cíl s ID 26 značí úhel -20° z pohledu pravého předního radaru - ten je pak přepočítán pro kameru na 25° a toho je možné si povšimnout na snímku 21, kde se ID nachází v pravé části horizontální osy (kamera má úhel záběru 66°, pravý okraj snímku tedy odpovídá úhlu 33°).

7.8 Výstupní videa

Jako výstup kódu slouží exportovaná videa nebo v případě potřeby zbežné kontroly pouze náhled při použití tlačitka **Show**. Následující obrázky zobrazují vybrané snímky výstupního videa pro demonstraci funkčnosti kódu.

Na obr. 21 je snímek videa z kamery umístěné za předním sklem.

Obrázek 21: Ukázkový snímek s vloženými ID - přední kamera, dvě detekce

Při pohledu na snímek jsou patrná dvě vložená ID - jedno ID značí automobil s pořadím range-binu 39, toto ID automobil však nesleduje, jelikož se ve snímku nachází velké množství objektů u vozovky, které často mají větší odražený výkon. Druhé vložené ID náleží lampě veřejného osvětlení umístěné u vozovky.

Na dalším obrázku č. 22 je opět snímek z přední kamery, ovšem zde ID poměrně přesně značí umístění patníku v křižovatce.

Obrázek 22: Ukázkový snímek s vloženými ID - přední kamera, jedna detekce

Při zkoumání jednotlivých videí je patrné, že v programu bylo nastaveno vyhledávání pouze nejsilnějšího cíle výstupu z MUSIC metody. To umožnilo snadnou orientaci při zpracování, ovšem pro zvýšení počtu detekovaných cílů promítnutých do videa by bylo potřeba zvolit mírně odlišný postup. Dle mého názoru jsou charakteristiky vysílacích antén takové, že nejvíce výkonu směřují dopředu radaru, tzn. vždy v úhlu 45° od podélné osy automobilu. Kamery ovšem míří jiným směrem a to vede k závěru, že většina detekovaných cílů leží mimo záběr kamer.

Z jednotlivých videí je tento jev zřejmý. Zatímco ve výstupech přední a zadní kamery jsou okamžiky, kdy se v záběru nenachází žádný z cílů, či jsou v záběru cíle dva z důvodu rozdílných nasměrování levého a pravého radaru, v případě bočních kamer se v obraze nachází cíl často jeden a to zejména od radaru zadního, se kterým kamera sdílí větší průnik úhlů záběrů.

Na obrázku č. 23 je snímek ze zadní palubní kamery a zobrazuje detekci stromu u vozovky. Toto ID určitou dobu daný cíl kopíruje a ve videu byl vidět postupný nárůst jeho hodnoty značící postupné vzdalování cíle.

Obrázek 23: Ukázkový snímek s vloženými ID - zadní kamera, jedna detekce

Závěr

V práci byly stručně popsány funkce, výhody a nevýhody FMCW radarů. Dále byl popsán princip anténních řad pro MIMO, metod pro určení DOA a zejména princip zpracování dat mezifrekvenčního signálu z FMCW radaru.

V praktické aplikaci byl úspěšně implementován program pro registraci radarových dat do záznamů palubních kamer umístěných v testovacím voze a ukázán rozdíl v přesnosti základních a pokročilých metod určení azimutu radarového cíle.

Umístění značky cíle do obrazu je poměrně přesné, ovšem pro další vývoj programu navrhuji zaměření právě na tuto oblast, ve které vidím možná zlepšení z hlediska lepší kalibrace senzorů, zpřesnění úhlů záběru kamer a doplnění možnosti vertikálního pohybu vložené značky na základě vzdálenosti cíle a jeho elevace.

Použitá literatura

[1] BROOKER, Graham M. *Understanding Millimetre Wave FMCW Radars*. 1 st International Conference on Sensing Technology [online]. New Zealand: IEEE, 2005, , 152-157 [cit. 2020-12-09]. Dostupné z: doi:10.1.1.590.1430

[2] HONGHAO, Tang. *DOA estimation based on MUSIC algorithm* [online]. Växjö, Švédsko, 2014 [cit. 2021-5-19]. Dostupné z: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A724272dswid=-1444. Bakalářská práce. Linnaeus University.

[3] CHEN, Zhizhang, Gopal K. GOKEDA a Yiqiang YU. *Introduction to Direction-of-Arrival Estimation*. 1. Norwood: Artech House, 2010. ISBN 978-1-59693-089-6.

[4] IVANOV, S. I., V. D. KUPTSOV a A. A. FEDOTOV. *The signal processing algorithm of automotive FMCW radars with an extended range of speed estimation*. Journal of Physics: Conference Series [online]. 2019, 1236 [cit. 2021-5-18]. ISSN 1742-6588. Dostupné z: doi:10.1088/1742-6596/1236/1/012081

[5] JANKIRAMAN, M. *FMCW Radar Design*. Norwood,MA: Artech House, 2018. ISBN 978-1-63081-567-7.

[6] RAO, Sandeep. *Introduction to mmwave Sensing: FMCW Radars* [online]. In: . 20.4.2017, s. 70 [cit. 2021-01-06]. Dostupné z: https://training.ti.com/sites/default/files/docs/mmwaveSensing-FMCW-offlineviewing_4.pdf

[7] RAO, Sandeep. *MIMO Radar* [online]. 2018 [cit. 2020-12-09]. Dostupné z: https://www.ti.com/lit/pdf/swra554

[8] SCHMIDT, Ralph O. *Multiple emitter location and signal parameter estimation*. IEEE Transactions on Antennas and Propagation [online]. 1986, 34(3), 276-280 [cit. 2021-5-19]. ISSN 0096-1973. Dostupné z: doi:10.1109/TAP.1986.1143830

[9] TEXAS INSTRUMENTS. *AWRx Cascaded Radar RF Evaluation Module: User's Guide*. 2020. Dostupné také z: https://www.ti.com/lit/ug/swru553a/swru553a.pdf

[10] WOJTKIEWICZ, Andrzej, Jacek MISIUREWICZ, Marek NAŁE¸CZ, Konrad JE¸DRZEJEWSKI a Krzysztof KULPA. *Two-dimensional signal processing in FMCW radars* [online]. Instytut Podstaw Elektroniki [cit. 2021-5-18]. Dostupné z: https://www.researchgate.net/profile/Konrad-Jedrzejewski/publication/235926794_Two-dimensional_Signal_Processing_in_FMCW_Radars/links/0046352f20849d9850000000/Two-dimensional-Signal-Processing-in-FMCW-Radars.pdf

[11] WOLFF, Christian. *Frequency-Modulated Continuous-Wave Radar (FMCW Radar)* [online]. In: . [cit. 2021-5-21]. Dostupné z: https://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.htmlabs2

Přílohy

GUI.m

```
function GUI()
    %% Variables
    figSize = [650, 550];
    selectedCamera = 'rearCam';
    InDataFileName = [];
    InDataPathName = [];
    InVideoFileName = [];
    InVideoPathName = [];
    OutVideoFileName = 'myVideo';
    OutVideoPathName = pwd;
    CamSpinner = [];
    MCC = [];
    Checked = [];
    resultAngle = [];
    rangeBins = [];

    %% Main window
    screenSize = get(groot, 'ScreenSize');

    hFig = uifigure('Position', [(screenSize(3:4) - figSize)/2 figSize],...
                    'Name', 'FMCW Analysis','Resize',false);

    %% Structures
    hInputDataPanel = uipanel(hFig, 'Position', [10, 490, 305, 50],...
        'Title', 'Select input data file');
    hInputVideoPanel = uipanel(hFig, 'Position', [10, 435, 305, 50],...
        'Title','Select input video file');
    hOutputVideoPanel = uipanel(hFig, 'Position', [320, 490, 305, 50],...
        'Title','Select output video file');
    hInputParamsPanel = uipanel(hFig, 'Position', [320, 365, 305, 120],...
        'Title', 'Input parameters');

    hRadioButtonGroup = uibuttongroup(hFig,'Position',[20 370 300 40],...
        'BorderType','none');

    hAx = uiaxes(hFig, 'Position', [80, 5, 461, 357], 'XLim', [0 461],...
        'YLim', [0 357], 'XColor', 'none', 'YColor', 'none');

    %% Elements
    hInputDataEdit = uieditfield(hInputDataPanel, 'Position',...
```

```
        [2, 2, 250, 25],'FontSize', 15, 'HorizontalAlignment', 'center');
    hInputVideoEdit = uieditfield(hInputVideoPanel, 'Position',...
        [2, 2, 250, 25],'FontSize', 15, 'HorizontalAlignment', 'center');
    hOutputVideoEdit = uieditfield(hOutputVideoPanel, 'Position',...
        [2, 2, 250, 25],'FontSize', 15, 'Value', 'myVideo',...
        'HorizontalAlignment', 'center');
    hMCCEdit = uieditfield(hInputParamsPanel, 'numeric', 'Position',...
        [55, 70, 75, 25],'Value', 84712, 'ValueDisplayFormat', '%d',...
        'HorizontalAlignment', 'center','ToolTip', 'starting MCC in video');
    hNEdit = uieditfield(hInputParamsPanel, 'numeric', 'Position',...
        [200, 70, 75, 25],'Value', 1024, 'ValueDisplayFormat', '%d',...
        'HorizontalAlignment', 'center','ToolTip', 'number of samples');
    hKEdit = uieditfield(hInputParamsPanel, 'numeric', 'Position',...
        [200, 40, 75, 25],'Value', 128, 'ValueDisplayFormat', '%d',...
        'HorizontalAlignment', 'center','ToolTip', 'number of chirps');

    hInDataButt = uibutton(hInputDataPanel, 'Text', 'Select',...
        'Position', [252, 2, 50, 25]);
    hOutVidButt = uibutton(hOutputVideoPanel, 'Text', 'Select',...
        'Position', [252, 2, 50, 25]);
    hCalcButt = uibutton(hFig, 'Text', 'Calculate', 'Position',...
        [35, 400, 60, 25]);
    hShowButt = uibutton(hFig, 'Text', 'Show', 'Position',...
        [110, 400, 60, 25],'Enable','off');
    hInVidButt = uibutton(hInputVideoPanel, 'Text', 'Select',...
        'Position', [252, 2, 50, 25]);

    hCheck = uicheckbox(hFig,'Value', false,'Text', 'Export video',...
        'FontSize', 15,'Position', [190,397, 180, 30]);

    hCamSpinner = uispinner(hInputParamsPanel,'Position',[55, 40, 75, 25],...
        'Value',66,'Limits',[0 180],'ToolTip',...
        'angle of camera view [0  —180  ]','ValueDisplayFormat','%d  ');

    hMCCLabel = uilabel(hInputParamsPanel, 'Text', 'MCC',...
        'HorizontalAlignment', 'Center', 'Position', [5, 73, 50, 20],...
        'FontSize', 12);
    hNLabel = uilabel(hInputParamsPanel, 'Text', 'N',...
        'HorizontalAlignment', 'Center', 'Position', [160, 73, 50, 20],...
        'FontSize', 12);
    hKLabel = uilabel(hInputParamsPanel, 'Text', 'K',...
        'HorizontalAlignment', 'Center', 'Position', [160, 42, 50, 20],...
        'FontSize', 12);
    hCamSpinnerLabel = uilabel(hInputParamsPanel, 'Text', 'Camera',...
        'HorizontalAlignment', 'Center', 'Position', [5, 42, 50, 20],...
        'FontSize', 12);

    hRearCam = uiradiobutton(hRadioButtonGroup,'Position',[1 2 100 20],...
        'Text','rearCam');
    hFrontCam = uiradiobutton(hRadioButtonGroup,'Position',[76 2 100 20],...
        'Text','frontCam');
    hLeftCam = uiradiobutton(hRadioButtonGroup,'Position',[153 2 100 20],...
        'Text','leftCam');
    hRightCam = uiradiobutton(hRadioButtonGroup,'Position',[221 2 100 20],...
```

```
        'Text','rightCam');

    %% Callbacks
    hInDataButt.ButtonPushedFcn = @(src, event)pressInDataButt();
    hInVidButt.ButtonPushedFcn = @(src, event)pressInVidButt();
    hOutVidButt.ButtonPushedFcn = @(src, event)pressOutVid();
    hCalcButt.ButtonPushedFcn = @(src, event)pressCalcButt();
    hShowButt.ButtonPushedFcn = @(src, event)pressShowButt();
    hCheck.ValueChangedFcn = @(src, event)checkCheckbox();
    hCamSpinner.ValueChangedFcn = @(src, event)changeCamSpinner();
    hMCCEdit.ValueChangedFcn = @(src, event)changeButtColor();
    hNEdit.ValueChangedFcn = @(src, event)changeButtColor();
    hKEdit.ValueChangedFcn = @(src, event)changeButtColor();
    hRadioButtonGroup.SelectionChangedFcn = @(src, event)changeRadioButt(event);

    %% Functions
    function pressInDataButt()
        [fileName, pathName] = uigetfile('*.hdf5', 'Select the hdf5-file');
        if fileName
            set(hInputDataEdit, 'Value', fileName);
            InDataFileName = fileName;
            InDataPathName = pathName;
            set(hCalcButt,'BackgroundColor',[0.96 0.96 0.96]);
            set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
        else
            set(hInputDataEdit, 'Value', '');
            InDataFileName = 0;
            InDataPathName = 0;
        end
    end

    function pressInVidButt()
        [fileName, pathName] = uigetfile('*.mp4', 'Select the video file');
        if fileName
            set(hInputVideoEdit, 'Value', fileName);
            InVideoFileName = fileName;
            InVideoPathName = pathName;
            set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
        else
            set(hInputVideoEdit, 'Value', '');
            InVideoFileName = 0;
            InVideoPathName = 0;
            set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
        end
    end

    function pressOutVid()
        [fileName, pathName] = uiputfile('*.mp4', 'Select output file');
        if fileName
            OutVideoFileName = fileName;
            OutVideoPathName = pathName;
            set(hOutputVideoEdit, 'Value', fileName);
            set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);

```

```
        else
            OutVideoFileName = 0;
            OutVideoPathName = 0;
            set(hOutputVideoEdit, 'Value', '');
            set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
        end
    end

    function pressCalcButt()
        set(hCalcButt,'BackgroundColor',[0.96 0.96 0.96]);
        if InDataPathName
            progress = uiprogressdlg(hFig,'Title','Please Wait', 'Message',...
                'Calculating DOA from input data');
            progress.Value = 0;

            fullHDF5 = fullfile(InDataPathName,InDataFileName);

            K = get(hKEdit,'Value');
            N = get(hNEdit,'Value');

            CamSpinner = hCamSpinner.Value;

            [resultAngle, MCC, rangeBins] = Calculation(progress,fullHDF5,...
                K, N,selectedCamera);
            close(progress)

            set(hShowButt,'Enable','on');
            set(hCalcButt,'BackgroundColor','g');
        else
            uialert(hFig,'Please specify input data file','Invalid File');
        end
    end

    function pressShowButt()
        set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
        set(hShowButt,'Enable','off');
        if InVideoPathName
            videoFileReader = VideoReader(fullfile(InVideoPathName,...
                InVideoFileName));

            myMCC = get(hMCCEdit,'Value');

            fullOutVideo = fullfile(OutVideoPathName,OutVideoFileName);

            Show(resultAngle,rangeBins,MCC,hAx,videoFileReader,...
                fullOutVideo,myMCC,CamSpinner,selectedCamera,Checked);

            set(hShowButt,'BackgroundColor','g');
            set(hCalcButt,'BackgroundColor','g');
        else
            uialert(hFig,'Please specify input video file','Invalid File');
        end
        set(hShowButt,'Enable','on');
    end
```

```
203
204     function checkCheckbox()
205         if hCheck.Value
206             if OutVideoFileName
207                 Checked = true;
208             else
209                 uialert(hFig,'Please specify output video filename',...
210                     'Invalid File');
211                 hCheck.Value = 0;
212             end
213         else
214             Checked = false;
215         end
216     end
217
218     function changeCamSpinner()
219         CamSpinner = hCamSpinner.Value;
220         changeButtColor()
221     end
222
223     function changeRadioButt(event)
224         changeButtColor()
225         selectedCamera = event.NewValue.Text;
226     end
227
228     function changeButtColor()
229         set(hCalcButt,'BackgroundColor',[0.96 0.96 0.96]);
230         set(hShowButt,'BackgroundColor',[0.96 0.96 0.96]);
231     end
232
233 end
```

Calculation.m

```
1  %% Calculation for FMCW Analysis program.
2  %
3  %  INPUTS
4  %   progress: reference to the progress bar used to inform user about
5  %   progress of calculation
6  %   fullHDF5: complete path to .hdf5 file with data from ADC
7  %   K: number of chirps
8  %   N: number of samples
9  %   selectedCamera: text value of which camera is selected
10 %
11 %  OUTPUTS
12 %   resultAngle: matrix of two rows which belong to selected sensors and
13 %   columns which represent angles of radar targets
14 %   MCC: vector of MCC for resultAngle and rangeBins variable
15 %   synchronization
16 %   rangeBins: matrix of two rows which belong to selected sensors and
17 %   columns which represent range bins of radar targets
18
19 function [resultAngle, MCC, rangeBins] = Calculation(progress,fullHDF5,K,N,...
20     selectedCamera)
```

```
21
22      switch selectedCamera
23          case 'rearCam'
24              sensorId1 = 4; % rear left
25              sensorId2 = 3; % rear right
26              Spdo1 = [0;-145.57;115.84;-76.59]*pi/180;
27              Spdo2 = [0;-142.69;118.31;-104.62]*pi/180;
28          case 'frontCam'
29              sensorId1 = 1; % front left
30              sensorId2 = 2; % front right
31              Spdo1 = [0;-120.7;108.74;-72.75]*pi/180;
32              Spdo2 = [0;-153.4;117.07;-108.0]*pi/180;
33          case 'leftCam'
34              sensorId1 = 3; % rear left
35              sensorId2 = 1; % front left
36              Spdo1 = [0;-145.57;115.84;-76.59]*pi/180;
37              Spdo2 = [0;-120.7;108.74;-72.75]*pi/180;
38          case 'rightCam'
39              sensorId1 = 2; % front right
40              sensorId2 = 4; % rear right
41              Spdo1 = [0;-153.4;117.07;-108.0]*pi/180;
42              Spdo2 = [0;-142.69;118.31;-104.62]*pi/180;
43      end
44
45      % sensorId = 1 . . . front left
46      %          = 2 . . . front right
47      %          = 3 . . . rear left
48      %          = 4 . . . rear right
49      % radars are directed to the corners of the car
50
51      beamId = 2; % 1=225MHz beam (long range,limited field of view towards front)
52                  % 2=450MHz (wide beam across field of veiw)
53
54      % placement of Rx element: lam/2, 3/2lam, lam (spaces between antennas)
55      % |_|___|__|
56      % 0 1   4  6
57      antennaPos = [0 1 4 6].';
58
59      hdf5_data1 = h5read(fullHDF5,strcat('/SensorHST_',num2str(sensorId1)));
60      hdf5_data2 = h5read(fullHDF5,strcat('/SensorHST_',num2str(sensorId2)));
61
62      nrRx = 4; % number of RX antennas
63
64      MCC1 = hdf5_data1.MCC;
65      MCC2 = hdf5_data2.MCC;
66
67      MUSICTheta = linspace(-pi/2, pi/2, N);
68
69      %Hamming window for 2DFFT
70      PRS_h1D = 0.54-0.46*cos(2*pi*(0:N-1)/N);
71      PRS_h2D = 0.54-0.46*cos(2*pi*(0:K-1)/K);
72
73      SpdoCorrection1 = shiftdim(exp(1i*Spdo1), -2);
74      SpdoCorrection2 = shiftdim(exp(1i*Spdo2), -2);
```

```
75
76      % Calculation/graphical part
77      maxK1 = length(hdf5_data1.Beams);
78      maxK2 = length(hdf5_data2.Beams);
79
80      [result1,rangeBins1] = CalcDOA(maxK1,SpdoCorrection1,0);
81      [result2,rangeBins2] = CalcDOA(maxK2,SpdoCorrection2,1/2);
82
83      % Preparation for video processing
84      resultAngle1 = interp1([1,1023],[-90,90],result1);% converts indexes of
85                                                        % first sensor to angles
86      resultAngle2 = interp1([1,1023],[-90,90],result2);% converts indexes of
87                                                        % second sensor to angles
88
89      resultAngle1 = resultAngle1 - 45; % left
90      resultAngle2 = resultAngle2 + 45; % right
91
92      if MCC1(1) > MCC2(1)
93          rangeBins1 = [NaN(size(rangeBins1,1),MCC2(1)-MCC1(1)) rangeBins1];
94          rangeBins2 = [rangeBins2 NaN(size(rangeBins2,1),MCC2(1)-MCC1(1))];
95          resultAngle1 = [NaN(size(resultAngle1,1),MCC2(1)-MCC1(1)) resultAngle1];
96          resultAngle2 = [resultAngle2 NaN(size(resultAngle2,1),MCC2(1)-MCC1(1))];
97
98          MCC = [MCC2; MCC1(MCC2(length(MCC2))-MCC1(1)+2:length(MCC1))];
99      elseif MCC1(1) < MCC2(1)
100         rangeBins2 = [NaN(size(rangeBins2,1),MCC2(1)-MCC1(1)) rangeBins2];
101         rangeBins1 = [rangeBins1 NaN(size(rangeBins1,1),MCC2(1)-MCC1(1))];
102         resultAngle2 = [NaN(size(resultAngle2,1),MCC2(1)-MCC1(1)) resultAngle2];
103         resultAngle1 = [resultAngle1 NaN(size(resultAngle1,1),MCC2(1)-MCC1(1))];
104
105         MCC = [MCC1; MCC2(MCC1(length(MCC1))-MCC2(1)+2:length(MCC2))];
106     end
107
108     resultAngle = [resultAngle1; resultAngle2];
109     rangeBins = [rangeBins1;rangeBins2];
110
111     function [resultTemp,rangeBinsTemp] = CalcDOA(maxK,SpdoCorrection,prevProg)
112         resultTemp = zeros(1,maxK);
113         rangeBinsTemp = zeros(1,maxK);
114         for nFrames=1:maxK  % 255
115             progress.Value = prevProg + nFrames/maxK/2;
116
117             measData = flip(permute(reshape(...         128*1024*4
118                 hdf5_data1.Beams{nFrames,1}.MeasData{beamId,1}, nrRx, N, K),...
119                 [3 2 1]),1);
120
121             Fft2D_permuted = FFT_2D(measData, SpdoCorrection, PRS_h1D,...
122                 PRS_h2D, K, N, nrRx);
123
124             [targetAntPhasors, targetRangeBin] = AntPhasors(Fft2D_permuted);
125             rangeBinsTemp(nFrames) = targetRangeBin;
126             % MUSIC
127             % autocorrelation matrix of recieved phasors
128             R = targetAntPhasors*targetAntPhasors';
```

```
129
130             % we use Minimum Norm MUSIC method, which is encrypted in *.p file,
131             % in *.m file is only header with input description
132             [~, P_MN] = MUSIC(R, 0.5, antennaPos, MUSICTheta, 1);
133             locs = find(P_MN==max(P_MN));
134             resultTemp(:,nFrames) = locs;
135         end
136     end
137 end
```

FFT_2D.m

```
1  %% Component for the 2D-FFT.
2  %
3  %  INPUTS
4  %   measData: input matrix of input data of selected frame
5  %   SpdoCorrection: calibration matrix
6  %   PRS_h1D: Hamming window for the first FFT
7  %   PRS_h2D: Hamming window for the second FFT
8  %   K: number of chirps
9  %   N: number of samples
10 %   nrRx: number of RX antennas
11 %
12 %  OUTPUTS
13 %   Fft2D_permuted: matrix of 2D-FFT of input data
14
15 function Fft2D_permuted = FFT_2D(measData, SpdoCorrection, PRS_h1D, PRS_h2D,...
16     K, N, nrRx)
17     % just to match the dimensions needed
18     VideoChirp2D = permute(measData,[2 1 3]);
19
20     FirstFftData = 16*fft(double(VideoChirp2D).*repmat(PRS_h1D',1,K,nrRx)...
21         ,N,1)/N;
22
23     % discard negative range bins
24     % range x doppler x channel
25     ComplexFft2D = 16*fft(double(FirstFftData(1:N/2,:,:)).*repmat(PRS_h2D,...
26         N/2,1,nrRx),K,2)/K;
27     % calibration
28     Fft2D_temp = ComplexFft2D.*(repmat(SpdoCorrection,[N/(2),K,1]));
29
30     % just to match the dimensions needed
31     Fft2D_permuted = permute(Fft2D_temp,[2 1 3]);
32 end
```

AntPhasors.m

```
1  %% Component for the MUSIC preprocessing.
2  %
3  %  INPUTS
4  %   Fft2D_permuted: matrix with 2D-FFT data
5  %
6  %  OUTPUTS
7  %   targetAntPhasors: 3D matrix with selected phasors
```

```
%   targetRangeBin: range bin of selected target

function [targetAntPhasors, targetRangeBin] = AntPhasors(Fft2D_permuted)
    % fftshift through first dimension (rows) - velocity, for the zero
    % velocity to be in the middle of the range
    thisRVData = fftshift(Fft2D_permuted, 1);
    thisRVData(:,1:5,:) = 0;
    % finding at which position of cut R-V plot for 1. antenna does the
    % strongest target is placed (result index is linear)
    [~, indTarget] = max(abs(thisRVData(:, :, 1)), [], 'all', 'linear');

    % it is necessarry to convert te linear index to numbers of rows and
    % columns
    [targetSpeedBin,targetRangeBin] = ind2sub(size(thisRVData,[1 2]),indTarget);

    % finding, which are the recieved phasors at the corresponding antennas
    % for the detected target
    targetAntPhasors = squeeze(thisRVData(targetSpeedBin, targetRangeBin, :));

end
```

Show.m

```
%% Function for showing the video with IDs and for video export.
%
%  INPUTS
%   resultAngle: matrix of two rows which belong to selected sensors and
%   columns which represent angles of radar targets
%   rangeBins: matrix of two rows which belong to selected sensors and
%   columns which represent range bins of radar targets
%   MCC: vector of MCC for resultAngle and rangeBins variable
%   synchronization
%   hAx: number of RX antennas
%   videoFileReader: instance of videoFileReader for the input video
%   processing
%   fullOutVideo: complete path to input video file with all camera records
%   myMCC: MCC from the user input
%   CamSpinner: selected angle of camera view
%   selectedCamera: label of the camera selected by user
%   Checked: Boolean value whether the video is to be exported

%  no OUTPUTS

function Show(resultAngle,rangeBins,MCC,hAx,videoFileReader,fullOutVideo,myMCC,...
    CamSpinner,selectedCamera,Checked)

switch selectedCamera
    case 'rearCam'
        % modification to meet the camera max view angle
        resultAngle(abs(resultAngle) > CamSpinner/2) = NaN;
        % basic conversion from angles to pixel indexes in video frames
        resultPixel = round(interp1([-CamSpinner/2,CamSpinner/2],[1,458],resultAngle));
        lCut = 30;
        rCut = 490;
```

```
    case 'frontCam'
        resultAngle(abs(resultAngle) > CamSpinner/2) = NaN;
        resultPixel = round(interp1([-CamSpinner/2,CamSpinner/2],[1,458],resultAngle));
        lCut = 1420;
        rCut = 1880;
    case 'leftCam'
        resultAngle(resultAngle > -45+CamSpinner/2) = NaN;
        resultAngle(resultAngle < -45-CamSpinner/2) = NaN;
        resultPixel = round(interp1([-45-CamSpinner/2,-45+CamSpinner/2],[1,458],resultAngle)
            );
        lCut = 957;
        rCut = 1417;
    case 'rightCam'
        resultAngle(resultAngle > 45+CamSpinner/2) = NaN;
        resultAngle(resultAngle < 45-CamSpinner/2) = NaN;
        resultPixel = round(interp1([45-CamSpinner/2,45+CamSpinner/2],[1,458],resultAngle));
        lCut = 493;
        rCut = 953;
end

resultPixelCleaned = [];
for row=1:size(resultPixel,1)
   if min(isnan(resultPixel(row,:))) == 0
       resultPixelCleaned = [resultPixelCleaned; resultPixel(row,:)];
   end
end

% Video processing
resultPixelRows = size(resultPixelCleaned,1);
counter = 1;
Nums = NumClass;

if Checked
    v = VideoWriter(fullOutVideo,'MPEG-4');
    open(v)
end

videoFig = image(hAx, 'CData', NaN(357,461,3));

while hasFrame(videoFileReader)
    myMCC = myMCC + 1;
    % 1 MCC per 2 frames
 defaultFrame = readFrame(videoFileReader);
    readFrame(videoFileReader);

    % cuts from original video
    selCam = defaultFrame(84:440,lCut:rCut,:);     % 356x460 pixels

    if any(myMCC == MCC)
        if counter < length(resultPixelCleaned)
            for id=1:resultPixelRows % id of peak
                % num of digits in id
                id_digNum = max(ceil(log10(abs(rangeBins(id,counter)))),1);
                id_vect = num2str(rangeBins(id,counter)) - '0'; % vector from id
```

```
                if ~isnan(resultPixelCleaned(id,counter))
                    if ~isnan(rangeBins(id,counter))
                    % rear camera
                         for digNum=1:id_digNum
                            if resultPixelCleaned(id,counter)+digNum*9 <= 462
                                selCam(150:150+9,...
                                    resultPixelCleaned(id,counter)...
                                    +(digNum-1)*9:resultPixelCleaned(id,...
                                    counter)+digNum*9,:)= Nums.pickNum(...
                                    id_vect(digNum));
                            end
                        end
                     end
                end
            end
            counter = counter + 1;
        end
    end

    % display video frame to screen
    set(videoFig, 'CData', flip(selCam));
    % if needed, export video to a file
    if Checked
        writeVideo(v,selCam);
    end
    pause(0.001);
end

if Checked
    close(v)
end

end
```

NumClass.m

```
%% Class of numbers for inserting.
%
%  contains matrixes of numbers from 0 to 9

classdef NumClass
   properties
      Value {mustBeNumeric}
   end
   methods (Static)
       function r = pickNum(Value)
          if Value == 1
             r = NumClass.makeOne();
          elseif Value == 2
             r = NumClass.makeTwo();
          elseif Value == 3
             r = NumClass.makeThree();
          elseif Value == 4
             r = NumClass.makeFour();
```

```
        elseif Value == 5
           r = NumClass.makeFive();
        elseif Value == 6
           r = NumClass.makeSix();
        elseif Value == 7
           r = NumClass.makeSeven();
        elseif Value == 8
           r = NumClass.makeEight();
        elseif Value == 9
           r = NumClass.makeNine();
        elseif Value == 0
           r = NumClass.makeZero();
        end
     end
    function r = makeOne()
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 255 255 0 0 0 0;
                    0 0 0 255 0 255 0 0 0 0;
                    0 0 255 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 255 255 255 255 255 0 0];
    end
    function r = makeTwo()
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 0 255 255 255 255 0 0 0;
                      0 0 255 0 0 0 255 0 0 0;
                      0 0 255 0 0 0 255 0 0 0;
                      0 0 0 0 0 0 255 0 0 0;
                      0 0 0 0 0 255 0 0 0 0;
                      0 0 0 0 255 0 0 0 0 0;
                      0 0 0 255 0 0 0 0 0 0;
                      0 0 255 0 0 0 0 0 0 0;
                      0 0 255 0 0 0 0 0 0 0;
                      0 0 255 255 255 255 255 0 0 0];
    end
    function r = makeThree()
       r = zeros(10,10,3);
       r(:,:,1) = [0 0 0 0 255 255 0 0 0 0;
                   0 0 0 255 0 0 255 0 0 0;
                   0 0 255 0 0 0 0 255 0 0;
                   0 0 0 0 0 0 255 0 0 0;
                   0 0 0 0 0 255 0 0 0 0;
                   0 0 0 255 255 0 0 0 0 0;
                   0 0 0 0 0 255 0 0 0 0;
                   0 0 0 0 0 0 255 0 0 0;
                   0 0 255 0 0 0 255 0 0 0;
                   0 0 255 255 255 255 0 0 0 0];
    end
    function r = makeFour()
```

```
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 0 0 0 0 255 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 255 0 0 0 0 0;
                    0 0 0 255 0 0 0 0 0 0;
                    0 0 255 0 0 255 0 0 0 0;
                    0 0 255 255 255 255 255 255 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0;
                    0 0 0 0 0 255 0 0 0 0];
    end
    function r = makeFive()
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 0 255 255 255 255 255 0 0;
                    0 0 0 255 0 0 0 0 0 0;
                    0 0 0 255 0 0 0 0 0 0;
                    0 0 0 255 0 0 0 0 0 0;
                    0 0 0 255 255 255 255 0 0 0;
                    0 0 0 0 0 0 0 255 0 0;
                    0 0 0 0 0 0 0 255 0 0;
                    0 0 0 0 0 0 0 255 0 0;
                    0 0 0 0 0 0 255 0 0 0;
                    0 0 0 255 255 255 0 0 0 0];
    end
    function r = makeSix()
         r = zeros(10,10,3);
         r(:,:,1) = [0 0 0 0 255 255 255 0 0 0;
                    0 0 0 255 0 0 0 0 0 0;
                    0 0 255 0 0 0 0 0 0 0;
                    0 0 255 0 0 0 0 0 0 0;
                    0 0 255 0 0 0 0 0 0 0;
                    0 0 255 0 255 255 255 0 0 0;
                    0 0 255 255 0 0 0 255 0 0;
                    0 0 255 0 0 0 0 255 0 0;
                    0 0 0 255 0 0 255 0 0 0;
                    0 0 0 0 255 255 0 0 0 0];
    end
    function r = makeSeven()
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 255 255 255 255 255 255 255 0;
                     0 0 0 0 0 0 0 0 255 0;
                     0 0 0 0 0 0 0 0 255 0;
                     0 0 0 0 0 0 0 0 255 0;
                     0 0 0 0 0 0 0 0 255 0;
                     0 0 0 0 0 0 0 255 0 0;
                     0 0 0 0 0 0 255 0 0 0;
                     0 0 0 0 0 255 0 0 0 0;
                     0 0 0 0 255 0 0 0 0 0;
                     0 0 0 255 0 0 0 0 0 0];
    end
    function r = makeEight()
        r = zeros(10,10,3);
        r(:,:,1) = [0 0 0 0 255 255 0 0 0 0;
```

```
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 0 0 255 255 0 0 0 0;
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 0 255 255 255 255 0 0 0];
      end
      function r = makeNine()
         r = zeros(10,10,3);
         r(:,:,1) = [0 0 0 0 255 255 0 0 0 0;
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 0 255 255 255 255 255 0 0;
                        0 0 0 0 0 0 0 255 0 0;
                        0 0 0 0 0 0 0 255 0 0;
                        0 0 0 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 255 0 0 0;
                        0 0 0 255 255 255 0 0 0 0];
      end
      function r = makeZero()
         r = zeros(10,10,3);
         r(:,:,1) = [0 0 0 0 255 255 0 0 0 0;
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 255 0 0 0 0 255 0 0;
                        0 0 0 255 0 0 255 0 0 0;
                        0 0 0 0 255 255 0 0 0 0];
      end
   end
end
```

MUSIC.m (k této funkci náleží zašifrovaný .p soubor se samotnou funkcí)

```
%% Implementation of MUSIC method for DOA estimation.
% Implemented according to Introduction to Direction-of-Arrival Estimation,
% Chen, Gokeda, Tu, Artech House, 2010.
%
%  INPUTS
%   R: Autocorrelation matrix of received data, complex double [M x M]
%   Delta: pitch between individual antennas in LUA relative to center
%          frequency, 0.5 typically, double [1 x 1]
%   antennaPos: positions of individual antennas in Delta units, for LUA it
%               would be (0:M-1).', or [0; 1; 2; ...; M-1], double [M x 1]
%   theta: for which azimuths the Music spectrum will be computed, rad,
%          double [n x 1]
%   d: number of targets for detection
```

```
14 %
15 %  OUTPUTS
16 %   P: spectrum computed using classical MUSIC, double [n x 1]
17 %   P_MN: spectrum computed using Minimum Norm MUSIC, double [n x 1]
18 %
19 %    2020, Viktor Adler, CTU in Prague, adlervik@fel.cvut.cz
```