



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

## SYSTÉM PRO SLEDOVÁNÍ A KLASIFIKACI OBJEKTŮ NA OBLOZE

SYSTEM FOR TRACKING AND CLASSIFICATION OF OBJECTS IN THE SKY

### SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

#### AUTOR PRÁCE

AUTHOR

Bc. Jakub Franka

#### VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Ilona Janáková, Ph.D.

BRNO 2023



# Semestrální práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

**Student:** Bc. Jakub Franka

**ID:** 220976

**Ročník:** 2

**Akademický rok:** 2023/24

## NÁZEV TÉMATU:

### Systém pro sledování a klasifikaci objektů na obloze

#### POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je navrhnout systém počítačového vidění pro detekci, trasování a klasifikaci létajících objektů na obloze. Zadání zahrnuje řešení po HW i SW stránce. Předpokládá se klasifikace do kategorií typu: letadlo, dron, pták, hmyz, případně dalších. Pro klasifikaci bude testováno i využití konvolučních neuronových sítí.

1. Seznamte se s danou problematikou. Proveďte rešerši existujících přístupů.
2. Navrhněte vhodnou kombinaci a uspořádání hardwarových komponent – kamera s optikou + PC/minipočítač.
3. Navržené zařízení realizujte.
4. Pořídeť dostatečně rozsáhlou a pestrou databázi reálných snímků/sekvencí.
5. Navrhněte algoritmy detekce a trasování objektů.
6. Navrhněte klasifikátor objektů.
7. Implementujte zvolené algoritmy do navrženého zařízení.
8. Vše otestujte. Definujte omezuječí podmínky. Zhodnotěte.

SP = 1,5,6.

#### DOPORUČENÁ LITERATURA:

JUREČKA, Tomáš. Detekce a klasifikace létajících objektů. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, FEKT, Ústav automatizace a měřicí techniky. Vedoucí práce Ilona Janáková.

SCHUMANN, Arne, Lars SOMMER, Johannes KLATTE, Tobias SCHUCHERT a Jurgen BEYERER. Deep cross-domain flying object classification for robust UAV detection. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2017, s. 1-6. ISBN 978-1-5386-2939-0. Dostupné z: doi:10.1109/AVSS.2017.8078558

**Termín zadání:** 18.9.2023

**Termín odevzdání:** 4.1.2024

**Vedoucí práce:** Ing. Ilona Janáková, Ph.D.

**doc. Ing. Petr Fiedler, Ph.D.**

předseda rady studijního programu

#### UPOZORNĚNÍ:

Autor semestrální práce nesmí při vytváření semestrální práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

## **ABSTRAKT**

Abstrakt práce v originálním jazyce

## **KLÚČOVÉ SLOVÁ**

Klíčová slova v originálním jazyce

## **ABSTRACT**

Překlad abstraktu (v angličtině, pokud je originálním jazykem čeština či slovenština; v češtině či slovenštině, pokud je originálním jazykem angličtina)

## **KEYWORDS**

Překlad klíčových slov (v angličtině, pokud je originálním jazykem čeština či slovenština; v češtině či slovenštině, pokud je originálním jazykem angličtina)

FRANKA, Jakub. *Systém pro sledování a klasifikaci objektů na obloze*. Semestrálna práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024. Vedúci práce: Ing. Ilona Janáková, Ph.D.

## Vyhľásenie autora o pôvodnosti diela

**Meno a priezvisko autora:** Bc. Jakub Franka

**VUT ID autora:** 220976

**Typ práce:** Semestrálna práca

**Akademický rok:** 2023/24

**Téma záverečnej práce:** Systém pro sledování a klasifikaci objektů  
na obloze

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno .....  
.....  
podpis autora\*

---

\*Autor podpisuje iba v tlačenej verzii.

## POĎAKOVANIE

Rád bych poděkoval vedoucímu bakalářské/diplomové/disertační práce panu Ing. XXX YYY, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

# Obsah

<b>Úvod</b>	<b>12</b>
<b>1 Detekcia objektov v obraze</b>	<b>13</b>
1.1 Prahovanie . . . . .	13
1.1.1 Globálne prahovanie . . . . .	13
1.1.2 Lokálne prahovanie . . . . .	15
1.2 Detekcia hrán . . . . .	15
1.2.1 Detekcia horizontu pomocou Houghovej transformácie . . . . .	17
1.3 Detekcia pohybu . . . . .	18
<b>2 Popis objektov</b>	<b>21</b>
2.1 príznaky . . . . .	21
<b>3 Klasifikácia</b>	<b>22</b>
3.1 Konvolučné neurónové siete . . . . .	22
<b>4 Detekcia a klasifikácia v jednom kroku</b>	<b>24</b>
4.1 R-CNN . . . . .	24
4.2 YOLO . . . . .	25
<b>5 Návrh kombinácie a usporiadania hardvérových komponentov</b>	<b>26</b>
5.1 Jednodoskový počítač . . . . .	26
5.2 Kamera . . . . .	26
5.3 Ovládacie prvky . . . . .	27
5.4 Uchytenie kamery . . . . .	27
<b>6 Tvorba datasetu</b>	<b>28</b>
6.1 Získavanie dát . . . . .	28
6.2 Úprava, rozdelenie a rozšírenie datasetu . . . . .	28
6.3 Testovanie použiteľnosti datasetu . . . . .	29
<b>7 Návrh softvéru</b>	<b>31</b>
7.1 Použitý jazyk a knižnice . . . . .	31
7.2 Štruktúra aplikácie . . . . .	31
7.3 Implementácia . . . . .	32
7.4 Návrh grafického rozhrania . . . . .	33
<b>Závěr</b>	<b>36</b>

<b>Literatúra</b>	<b>37</b>
<b>Zoznam symbolov a skratiek</b>	<b>38</b>
<b>Zoznam príloh</b>	<b>39</b>
<b>A Některé příkazy balíčku <i>thesis</i></b>	<b>40</b>
A.1 Příkazy pro sazbu veličin a jednotek . . . . .	40
A.2 Příkazy pro sazbu symbolů . . . . .	40
<b>B Druhá příloha</b>	<b>41</b>
<b>C Příklad sazby zdrojových kódů</b>	<b>42</b>
C.1 Balíček <i>listings</i> . . . . .	42
<b>D Obsah elektronické přílohy</b>	<b>45</b>

# Zoznam obrázkov

1.1	Funkcia prahovania s prahom 0.5 pre svetlé pozadie . . . . .	13
1.2	Jednoduché prahovanie . . . . .	14
1.3	Automatické prahovanie (podľa známeho rozloženia) . . . . .	14
1.4	Adaptívne prahovanie (podľa známeho rozloženia) . . . . .	15
1.5	Cannyho hranový detektor . . . . .	17
1.6	Detekcia horizontu pomocou Houghovej transformácie . . . . .	18
1.7	Postup aktualizovania dynamického modelu pozadia . . . . .	19
1.8	Odčítanie pozadia . . . . .	20
3.1	Príklad architektúry konvolučnej neurónovej siete . . . . .	22
5.1	návrh nastaviteľného uchytenia kamery . . . . .	27
5.2	bloková schéma zariadenia . . . . .	27
6.1	Metriky presnosti naučeného modelu v programe roboflow . . . . .	29
6.2	Grafické zobrazenie priebehu učenia modelu roboflow . . . . .	30
6.3	Exportovanie vytvoreného datasetu . . . . .	30
7.1	Záložka na výber poskytovateľa obrazu . . . . .	33
7.2	Záložka na výber metódy detektie objektov . . . . .	34
7.3	Záložka na výber metódy sledovania objektov . . . . .	34
7.4	Záložka na výber detekovaných tried . . . . .	35
B.1	Alenčino zrcadlo . . . . .	41

## **Zoznam tabuliek**

A.1 Přehled příkazů . . . . .	40
-------------------------------	----

# **Zoznam výpisov**

C.1	Ukázka sazby zkratek . . . . .	42
C.2	Příklad Schur-Cohnova testu stability v prostředí Matlab. . . . .	43
C.3	Příklad implementace první kanonické formy v jazyce C. . . . .	44

# Úvod

Úvod studentské práce, např. . .

Nečíslovaná kapitola Úvod obsahuje „seznámení“ čtenáře s problematikou práce. Typicky se zde uvádí: (a) do jaké tematické oblasti práce spadá, (b) co jsou hlavní cíle celé práce a (c) jakým způsobem jich bylo dosaženo. Úvod zpravidla nepřesahuje jednu stranu. Poslední odstavec Úvodu standardně představuje základní strukturu celého dokumentu.

Tato práce se věnuje oblasti **DSP!** (**DSP!**), zejména jevům, které nastanou při nedodržení Nyquistovy podmínky pro **symfvz!** (**symfvz!**).<sup>1</sup>

Šablona je nastavena na *dvostranný tisk*. Nebudte překvapeni, že ve vzniklé PDF jsou volné stránky. Je to proto, aby důležité stránky jako např. začátky kapitol začínaly po vytisknutí a svázání vždy na pravé straně. Pokud máte nějaký závažný důvod sázet (a zejména tisknout) jednostranně, nezapomeňte si přepnout volbu **twoside** na **oneside**!

---

<sup>1</sup>Tato věta je pouze ukázkou použití příkazů pro sazbu zkratek.

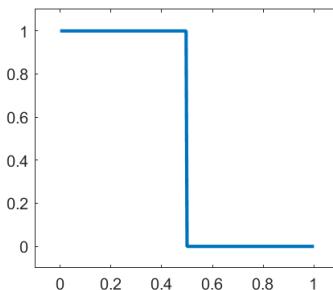
# 1 Detekcia objektov v obraze

Pre niektoré techniky klasifikácie objektov je nutné najprv v obraze tieto objekty nájsť. Existuje k tomu veľké množstvo metodík, pri čom každá z nich má svoje výhody a nevýhody. Je teda nutnosťou zistiť ktorá metóda najviac vyhovuje použitiu pre túto prácu.

## 1.1 Prahovanie

Prahovanie je jasová transformácia pri ktorej je šedotónový obraz rozdelený na dve a viac oblastí podľa jasovej úrovne pixelov. O tom, do ktorej z oblastí je daný pixel priradený rozhoduje prah, ktorého hodnotu je treba určiť. Výsledkom je obraz s len dvomi úrovňami jasu - binárny obraz.

Pri správnom predspracovaní obrazu prahovaním, je možné v ňom jednoducho oddeliť objekty od pozadia, v prípade ak sa jas objektu dostatočne líši od jasu pozadia. Ak sa však jasové hodnoty objektu a pozadia príliš podobajú, nemusí existovať dostatočne robustné určenie prahu oddelujúceho objekty od pozadia.



Obr. 1.1: Funkcia prahovania s prahom 0.5 pre svetlé pozadie

### 1.1.1 Globálne prahovanie

Globálne prahovanie je často používaná technika pri ktorej je použitý jediný prah na celý obraz, čo nemusí byť vhodné v prípade ak je jasová úroveň pozadia rôzna v rôznych miestach obrazu.

- **Jednoduché prahovanie** je technika pri ktorej je globálny prah určený dopredu ručne. V prípade že sa jasová úroveň pozadia v čase mení, je vhodnejšie použiť techniky s adaptívnym nastavením prahu.
- Prahovanie **podľa známeho rozloženia** predpokladá že poznáme relatívnu veľkosť oblasti ktorú zaberá pozadie v obraze. Ak vieme že objekt je svetlejší ako pozadie, môžeme určiť prah z kumulatívneho histogramu tak, aby relatívny

počet pixelov pod úrovňou prahu bol rovnaký ako oblasť ktorú ma zaberať pozadie.

- Algoritmus **K-Means** (K priemerov) pre zhlukovú analýzu, rozdeľuje dátu do skupín s cieľom minimalizovať vzdialenosť bodov v zhluku a maximalizovať vzdialenosť medzi zhlukmi. Jedná sa o algorytmus učenia bez učiteľa. Funguje na princípe iterovaného posúvania  $k$  stredov zhlukov (v prípade binárneho prahovania je  $k = 2$ ), smerom k priemeru hodnôt priradenému danému stredu v aktuálnom kroku.
- **Otsuova metóda** automatického určenia prahu je založená na maximalizovaní vzájomnej odchýlky medzi triedami. Využíva normalizovaný kumulatívny histogram z ktorého určuje vzájomnú rozptyl pre všetky možné hodnoty prahu a vyberá ten optimálny.



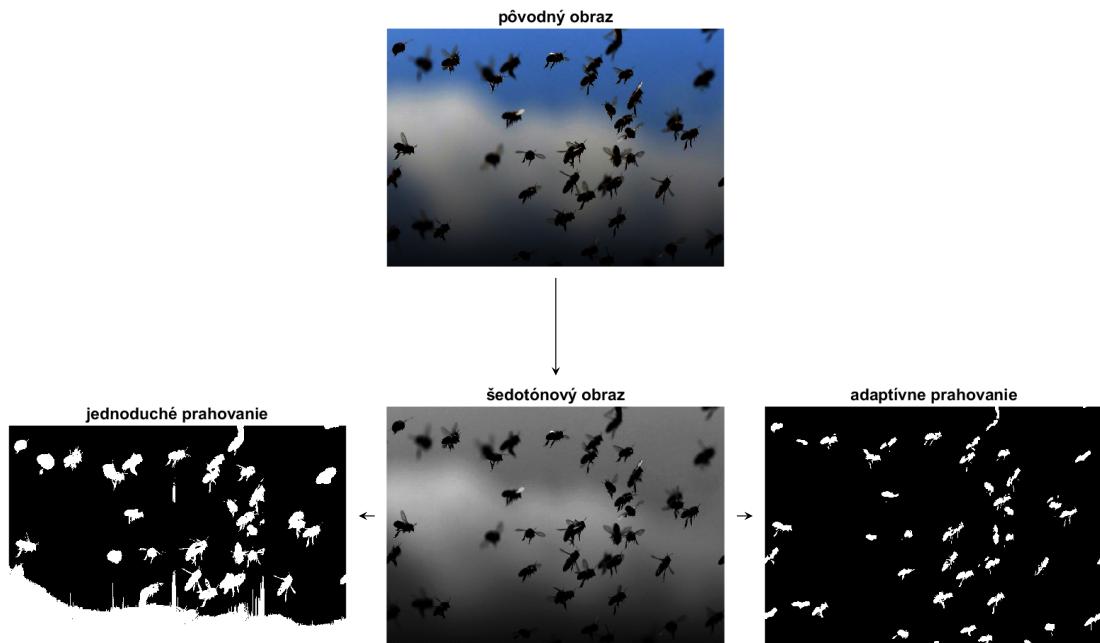
Obr. 1.2: Jednoduché prahovanie



Obr. 1.3: Automatické prahovanie (podľa známeho rozloženia)

### 1.1.2 Lokálne prahovanie

V prípade že je jas pozadia rôzny v rôznych častiač obrazu, je vhodné určiť hodnotu prahu z jasovej úrovne okolia každého prahovaného pixelu. Toto takzvané adaptívne prahovanie, má súčasť vyššiu výpočetnú náročnosť ako jednoduchšie globálne prahovanie, keďže je nutné určiť prah pre každý bod obrazu samostatne, je však robustnejšie voči nevhodnému pozadiu.



Obr. 1.4: Adaptívne prahovanie (podľa známeho rozloženia)

## 1.2 Detekcia hrán

Ako jednu z metód vyhľadávania objektov v obraze je možné využiť obraz hrán a vyhľadať v ňom kontúry objektov. Opäť existuje niekoľko spôsobov ako získať obraz hrán a ako ho využiť k segmentácii obrazu, teda k detekcii objektov.

Na vytvorenie obrazu hrán z nasnímaného obrazu sa používa konvolúcia obrazu s operátorom navrhnutým tak aby aproximovalo deriváciu určitého rádu. Medzi často používané operátory patria Prewittov a Sobelov operátor, ktoré approximujú prvú deriváciu v určitom smere (horizontálnom, vertikálnom, diagonálne). Druhú deriváciu approximuje Laplaceov operátor, používa sa tiež v kombinácii s Gaussovým filtrom (LoG operátor), čo má za následok zníženie citlivosti na šum.

Komplexnejšou metódou detekcie hrán je takzvaný *Cannyho hranový detektor*. Ide o proces vo viacerých krokoch, pre jeden vstupný obraz sa postupuje nasledovne:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Prewittov  
operátor

Sobelov  
operátor

Laplaceov  
operátor

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

LoG operátor

1. Vstupný obraz býva väčšinou filtrovaný z dôvodu zníženia citlivosti na šum v obraze. Vhodné je k tomu napríklad Gaussove rozmazanie.
2. Použije sa horizontálny a vertikálny Sobelov operátor ako detektor hrán, na získanie obrazu hrán v oboch smeroch  $I_x$  a  $I_y$ .

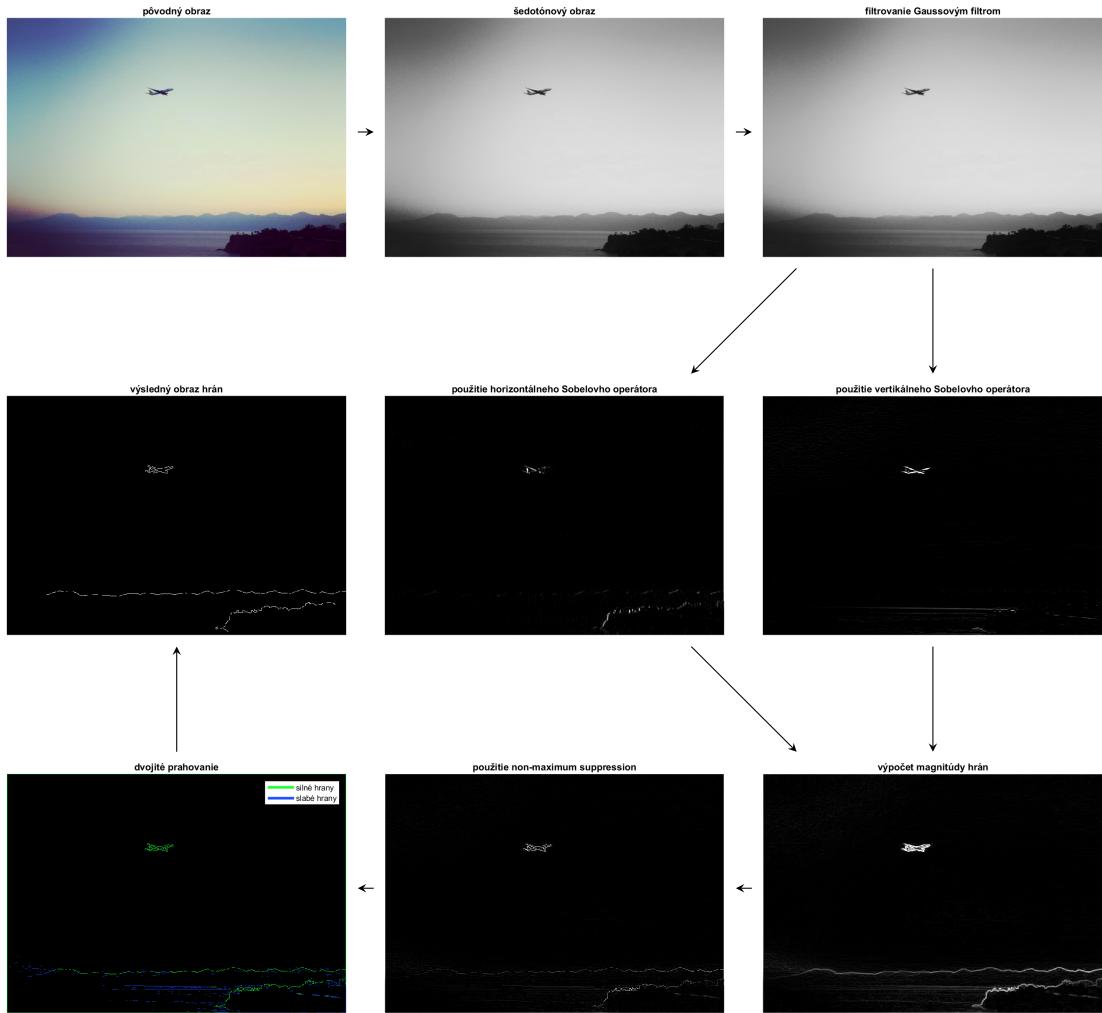
$$I_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \quad I_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

3. Z toho je možné určiť magnitúdu a smer gradientu:

$$G = \sqrt{I_x^2 + I_y^2} \quad \theta = \text{atan2}(I_x, I_y)$$

4. Ku zníženiu redundancie detekovaných hrán sa použije algorytmus *non maxima suppression* (potlačenie nemaximálnej hodnoty). Použitím tohto algoritmu dôjde k zúženiu hrán, pričom sa ponechajú len tie časti hrán ktorých magnitúda je väčšia ako v ich okolí.
5. Dvojitým prahovaním sa v obraze hrán nájdú tri úrovne:
  - **silné hrany** sú hrany vyššie ako obidva prahy. Do výsledného obrazu hrán sú určite pridané.
  - **slabé hrany** sú hrany medzi dvomi prahmi.
  - **nie hrany** sú tie hrany ktorých magnitúda je nižšia ako obidva prahy.
6. Iteratívne sú slabé hrany dotýkajúce sa silných hrán označované za silné, až kým sa žiadna slabá hrana silnej nedotýka. Vo výslednom obraze sú ponechané len silné hrany.

Obraz hrán je možné ďalej spracovať, napríklad morfológickými operáciami: otvorením odstrániť krátke nevýznamné hrany, zatvorením spojiť hrany blízko pri sebe.

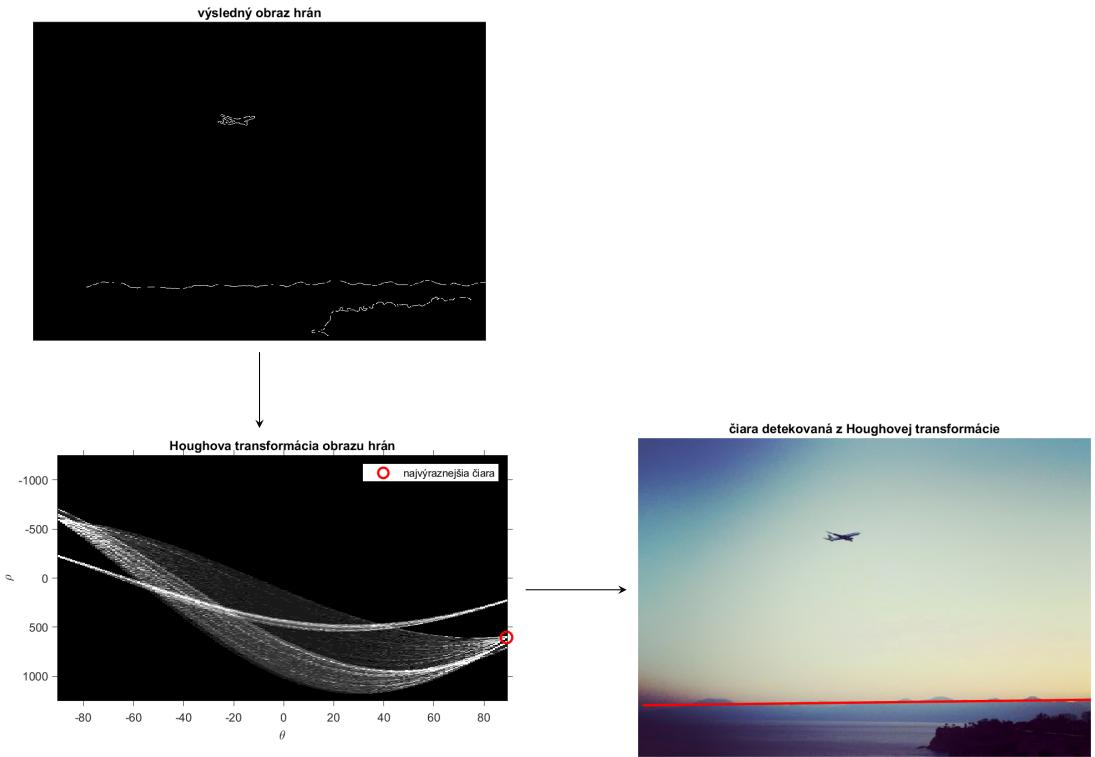


Obr. 1.5: Cannyho hranový detektor

Vyhľadávaním uzavertých kontúr v obraze hrán je možné identifikovať potenciálne objekty v obraze.

### 1.2.1 Detekcia horizontu pomocou Houghovej transformácie

Pokial časť obrazu obsahuje horizont, je vhodnejšie detektovať lietajúce objekty len na oblohe nad ním, aby sa predišlo falošným detekciám. Horizont je možné detektovať z obrazu hrán pomocou Houghovej transformácie, podľa práce. Vo výsledku Houghovej transformácie sa horizont prejaví ako najvýraznejšia čiara.



Obr. 1.6: Detekcia horizontu pomocou Houghovej transformácie

### 1.3 Detekcia pohybu

V prípade že je požadovaná detekcia pohybujúcich sa objektov, je možnosťou detektie práve detekovaním ich pohybu. Jednou z jednoduchých metód detektie pohybu sú rozdielové metódy, pri ktorých je vypočítavaný rozdielový snímok z viacerých snímkov zo sekvencie  $\{I_1, I_2, \dots, I_n\}$ . Rozdielový snímok môže byť:

- **jednostranný** - je najjednoduchší nenesie informáciu o smere pohybu, využíva sa len v miestach kde  $I_1(x, y) > I_2(x, y)$ .

$$D(x, y) = \begin{cases} 0 & I_1(x, y) - I_2(x, y) < \epsilon \\ 1 & I_1(x, y) - I_2(x, y) \geq \epsilon \end{cases}$$

- **obojstranný** - dostaneme ho upravením vzťahu pre jednostranný rozdielový snímok, použitím absolútnej hodnoty. Tým je dosiahnutá zameniteľnosť  $I_1(x, y)$  a  $I_2(x, y)$ , využíva sa teda na celom obrazu. Neodstraňuje nedostatok informácie o smere pohybu, tá však nemusí byť nutnosťou.

$$D(x, y) = \begin{cases} 0 & |I_1(x, y) - I_2(x, y)| < \epsilon \\ 1 & |I_1(x, y) - I_2(x, y)| \geq \epsilon \end{cases}$$

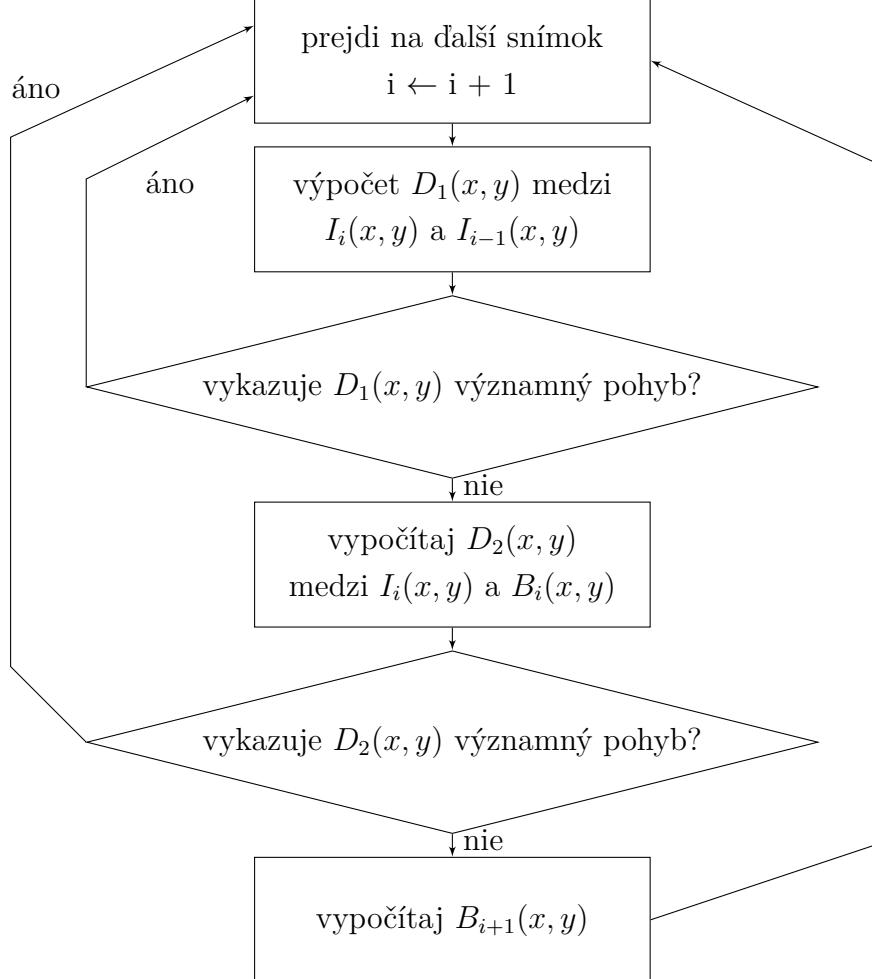
- **kumulovaný** - je vytvorený váženým súčtom jedno alebo obojstranných rozdielových snímkov. Je teda možné priemerovať pohyb za určitý čas určením každej váhy  $\omega = 1/(N - 1)$ , alebo určiť rôznym snímkom rôznu váhu a získať tak informáciu o smere pohybu.

$$D(x, y) = \sum_{i=1}^{N-1} \omega_i \cdot D_i(x, y)$$

Komplexnejšou metódou detektie pohybu je vytvorenie rozdielového snímku nie medzi nasledujúcimi snímkami, ale aktuálnym snímkom a vytvoreným modelom pozadia. Ten je možné zostaviť:

- ako jeden obraz pozadia bez objektov.
- ako priemerný snímok niekoľkých snímkov pozadia.
- ako dynamický model - iteratívnym aktualizovaním podľa aktuálneho snímku.

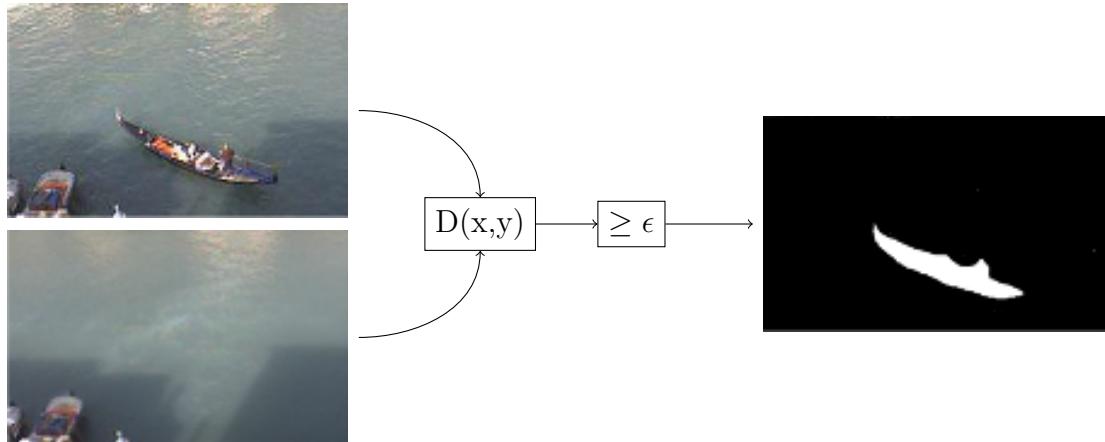
Tým sa model pozadia postupne prispôsobuje malým zmenám v prostredí.



Obr. 1.7: Postup aktualizovania dynamického modelu pozadia

Výpočet nového modelu pozadia  $B_{i+1}(x, y)$  môže prebiehať napríklad pomocou lineárneho zabúdania, teda  $B_{i+1}(x, y) = \alpha \cdot I_i(x, y) + (1 - \alpha) \cdot B_i(x, y)$ .

Obraz obsahujúci detekovaný pohyb je získaný rozdielom a prahovaním aktuálneho snímku a modelu pozadia.



Obr. 1.8: Odčítanie pozadia

## **2 Popis objektov**

### **2.1 príznaky**

### 3 Klasifikácia

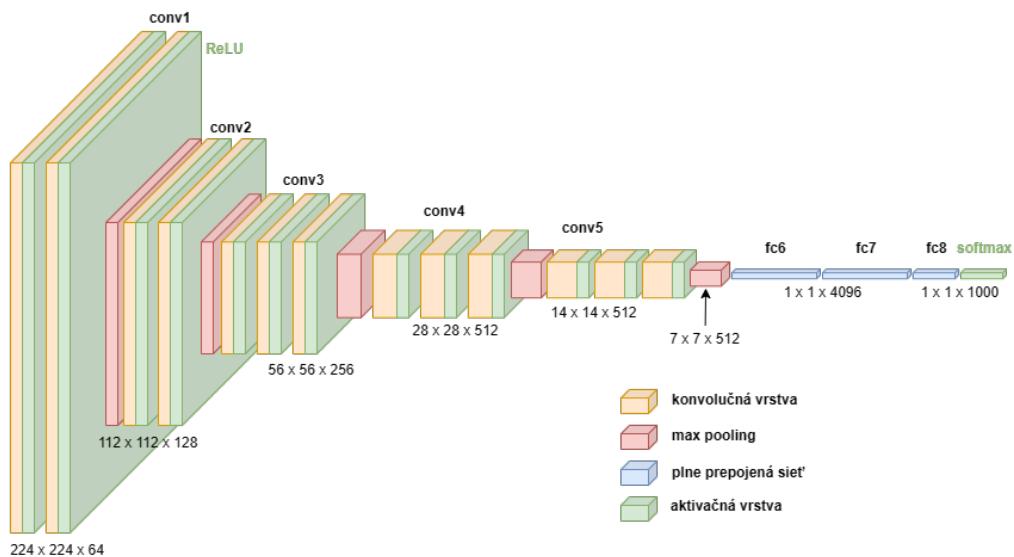
Ďalším krokom je určenie triedy detekovaných objektov, teda ich klasifikácia. Tá môže byť vykonaná na základe obrazu, alebo pomocou príznakov získaných z neho.

#### 3.1 Konvolučné neurónové siete

Na rozdiel od plne prepojenej neurónovej siete, sú *Konvolučné neurónové siete (Convolutional neural network)* (CNN) prispôsobené na prácu s obrazom ako so vstupným signálom. Ich architektúra je usporiadaná do vrstiev, štandardne po sebe nasledujú:

1. **Konvolučná vrstva** - používa konvolučné filtre na extrakciu príznakov z obrazu.
2. **Aktivačná vrstva** - aplikuje aktivačnú funkciu, často napríklad *Usmernená lineárna jednotka* (ReLU) na každý bod. Zavádzajú do siete nelinearitu a pomáha zachytávať zložitejšie vzory.
3. **Pooling vrstva** - redukuje priestorové rozlíšenie vstupných dát. Tým sa zníži výpočtová náročnosť nasledujúcich vrstiev. Najčastejšie sa používa *max pooling*, teda výber najvyššej hodnoty v okolí.

Väčšinou je použitá kombinácia niekoľko **konvolučných** vrstiev nasledovaných **aktivačnou** vrstvou, po ktorých **pooling** vrstva pripraví redukovaný vstup pre ďalšie vrstvy. Takýchto kombinácií nasleduje niekoľko, posledná je pripojená na plne prepojenú neurónovú sieť. Konvolučná časť slúži na vyhľadávanie príznakov, plne prepojená časť na klasifikáciu pomocou nich.



Obr. 3.1: Príklad architektúry konvolučnej neurónovej siete

Rozmery vrstiev závisia na tvare vstupného signálu. V prípade šedotónového obrazu je každý bod obrazu reprezentovaný jednou skalárhou hodnotou, v prípade farebného záleží počet hodnôt na pixel na farebnom modele. Klasifikáciou z farebného obrazu získavame väčšie množstvo informácií ale je vyžadovaný vyšší výkon, nakoľko sa konvolučné filtre aplikujú po jednom na každú zložku obrazu (teda v prípade RGB modelu na červenú, zelenú a modrú zložku samostatne).

Výstup z poslednej vrstvy plne prepojenej časti siete je výstupom z celej CNN. Výstupy klasifikačnej neurónovej siete s úlohov klasifikovať viacero tried, mávajú formáty:

- Jeden výstup, ktorého hodnota určuje triedu do ktorej sú vstupné dátá sietou priradené.
- Vektor výstupov výstupov s toľkými hodnotami, na koľko tried je naučená siet klasifikovať. Hodnoty výstupov môžu reprezentovať:
  - Ohodnotenie triedy (score-based classification). V tomto prípade čísla na výstupe nemajú samostatne význam, ale ako predpoveď je určená trieda s najvyšším skóre.
  - Pravdepodobnosť zaradenia do danej triedy. Tá je získaná použitím výstupnej aktivačnej vrstvy typu *softmax*, ktorá spôsobí že každý z výstupov má hodnotu od 0 do 1 a súčet všetkých výstupov je 1.

# 4 Detekcia a klasifikácia v jednom kroku

## 4.1 R-CNN

Konvolučné neurónové siete založené na regiónoch (Region-based CNN) (R-CNN) sú skupina modelov strojového učenia založená na CNN. Ich cieľom je nájdenie a klasifikovanie objektov v obraze. Ich výstupom je množina rámčekov ohraničujúcich objekty a im pridelené triedy.

R-CNN pracujú v niekoľkých etapách:

1. **Selektívne vyhľadávanie** - vstupný obraz je spracovaný a sú extrahované *Regióny záujmu* (*Regions of Interest*) (RoI), teda orámované oblasti obrazu, ktoré by mohli obsahovať objekty alebo ich časti. Počet takto navrhovaných oblastí môže dosahovať niekoľko tisíc.
2. **Extrakcia príznakov** - každý RoI je predložený ako vstup pre naučenú CNN, ktorá vyprodukuje príznakový vektor pre každý RoI.
3. **Klasifikácia** - pomocou skupiny modelov *Metódy podporných vektorov* (*Support Vector Machine*) (SVM) je podľa príznakov extrahovaných CNN, klasifikovaný každý RoI do jednej z určených tried alebo ako pozadie - teda nepatriaci do žiadnej triedy.
4. **Regresia ohraničení** - konečný krok, zvyšujúci presnosť orámovania objektov. Používa sa pri ňom naučený model nezávyslý na mierka nazývaný *bounding box regressor*. Jeho výstup je štvorozmerný, skladá sa z polohy a rozmerov ohraničujúceho obdĺžnika.

R-CNN sú efektívne, no náročné na výpočetný výkon. Z toho dôvodu boli vyvinuté rýchlejšie varianty *Fast R-CNN* a *Faster R-CNN*.

*Fast R-CNN* aplikuje CNN na celý vstupný obraz a extrahuje z neho mapu príznakov. Až na výslednú mapu aplikuje takzvaný RoI *pooling*, ktorý extrahuje príznaky pre každý RoI, pomocou okna pevnej veľkosti. Zvyšok modelu pracuje podobne ako R-CNN, s využitím plne prepojenej siete na klasifikáciu a generovanie orámovaní.

*Faster R-CNN* nadväzuje na *Fast R-CNN* nahradením selektívneho vyhľadávania modelom typu *Siet návrhu regiónov* (*Region Proposal Network*) (RPN). Vstupný obraz prechádza predučenou CNN, sú extrahované príznaky. RPN využije nájdené príznaky aby určila kde sa nachádzajú potenciálne objekty v obraze. V konečnom kroku sú klasifikované navrhnuté ohraničené oblasti podľa príznakov extrahovaných v predošлом kroku. *Faster R-CNN* je dostatočne rýchla na použitie v reálnom čase.

## 4.2 YOLO

*You only look once* (YOLO) je populárny algorytmus detekcie, ktorý na rozdiel od R-CNN rozdeľuje obraz do mriežky a postupne aplikuje klasifikátor. Postupuje nasledovne:

1. Vstuný obraz je rozdelený na mriežku  $S \times S$ , ktorej veľkosť závisí na verzii YOLO.
2. V každej bunke mriežky je predikovaný určitý počet ohraničených oblastí a im priradené skóre dôveryhodnosti. Tie značia úroveň istoty že oblasť obsahuje objekt a že dané ohraničenie je správne.
3. Je použitých viacero klasifikátorov. YOLO predikuje viacero oblastí za každú bunku. Počas trénovalia je jednému z prediktorov udelená zodpovednosť za predikovanie daného objekto podľa toho ktorý z nich dosahuje najväčší *Pomer prieniku voči zjednoteniu* (*Intersection over Union*) (IOU).
4. Každá bunka predikuje rozdelenie pravdepodobnosti všetkých tried.

Hlavnou výhodou YOLO je rýchlosť. Všeobecne pracuje rýchlejšie a s menším požadovaným výpočetným výkonom. Na druhú stranu má nevýhody spôsobené obmedzením veľkosti orámovaných oblastí. Presnosť YOLO býva nižšia pri primalých objektoch, pre použitia vyžadujúce vyžšiu presnosť sa môže javiť ako výhodnejšie použiť iné modely.

# 5 Návrh kombinácie a usporiadania hardvérových komponentov

Táto kapitola sa zameriava na návrh a implementáciu prenosného zariadenia pre detekciu, klasifikáciu a trasovanie lietajúcich objektov. Cieľom návrhu je vytvorenie zariadenia s možnosťou nasadenia v čo najrôznejších podmienkach. Jednou z podminok je teda jeho prenosnosť, možnosť napájania z akumulátoru a nezávyslosť na iných zariadeniach pri jeho nastavení a používaní.

## 5.1 Jednodoskový počítač

Jedným z najčastejšie používaných jednodoskových počítačov je rada *Raspberry Pi*. V čase návrhu bol najnovším model *Pi 4 model B*. Tento model disponuje:

- 1.5GHz ARM Cortex-A72 procesorom so 4 jadrami,
- do 8 Gb pamäte RAM,
- rozhraním HDMI s možnosťou pripojenia dvoch monitorov,
- 4 USB, z toho 2 USB 3.0,
- MIPI CSI konektor pre pripojenie kamery.

Ďalšou možnosťou je použitie jednodoskového počítača špeciálne navrhnutého na použitie v aplikáciách počítačového videnia, či všeobecne umelej inteligencie (Nvidia Jetson Xavier, Google Coral Dev Board, Rock Pi, Nvidia Jetson Nano, a podobné). *Raspberry Pi* má v porovnaní nižší odber, je kompaktnejšie, jednoduché na použitie a má dobrú kompatibilitu s operačnými systémami. Dokumentácia a knižnice pre *Raspberry Pi* sú jednoducho dostupné a zariadenia majú garantovanú dlhodobú podporu software aj hardware.

Z týchto dôvodov bolo *Raspberry Pi* zvolené na použitie v zariadení.

## 5.2 Kamera

*Raspberry Pi* dovoľuje pripojenie kamery cez MIPI konektor. Je teda najjednoduchšie použiť kamerový modul vyrobený konkrétnie pre *Raspberry Pi*. Najaktuálnejší oficiálny *Raspberry Pi* kamerový modul v čase návrhu je 12Mpx *Raspberry Pi Camera 3* so senzorom Sony IMX708 a automatickým ostrením. Parametre kamery a objektívu sú:

- **Ohnisková vzdialenosť:** 4,74 mm
- **Horizontálne zorné pole** 66 °
- **Vertikálne zorné pole** 41 °

## 5.3 Ovládacie prvky

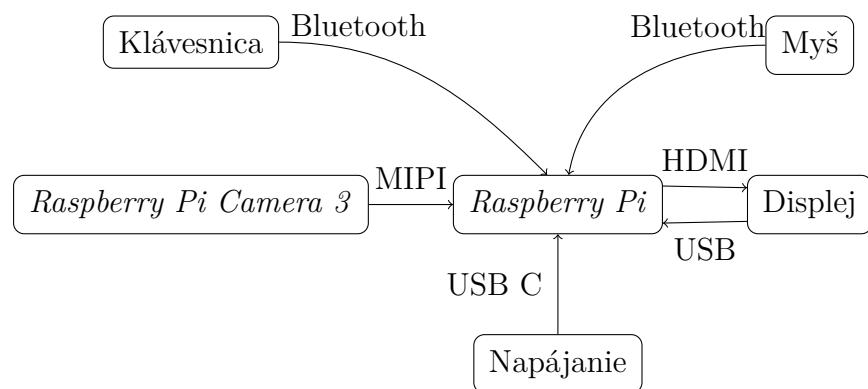
Ako najlepší ovládaci a zobrazovací prvok bol z hľadiska prenosnosti a kompaktnosti zvolený dotykový displej, konkrétnie 7 palcový IPS displej od firmy Waveshare. Táto veľkosť je postačujúca na zobrazovanie výsledkov detekcie aj ovládanie zariadenia. Rozhranie HDMI značne zjednodušuje prepojenie a inštaláciu displeja.

## 5.4 Uchytenie kamery

Praktickou výhodou prenosného zariadenia by bola možnosť jeho umiestnenia o státiv. Kedže je cieľom kamerou mieriť smerom nahor, na oblohu, bolo by užívateľské rozhranie pri nesprávnej vzájomnej orientácii kamery a displeja natočené neprakticky nadol. Namiesto pevnej konfigurácie uhla medzi displejom a kamerou bolo navrhnuté pohyblivé uchytenie kamery v kryte zariadenia, vyrobiteľné 3D tlačou.



Obr. 5.1: návrh nastaviteľného uchytenia kamery



Obr. 5.2: bloková schéma zariadenia

## 6 Tvorba datasetu

Správne naučenie modelov a ich schopnosť generalizovať z veľkej miery záleží na kvalite použitého datasetu (množiny dát). Tvorba datasetu je teda klúčový krok pri práci so systémami strojového videnia.

Cieľom pri tvorbe datasetu je zabezpečiť dostatočnú kvalitu a správnosť dát, dosiahnuť početne podobné zastúpenie každej triedy a správne reprezentovať rôznorodosť dát v rámci jednotlivých tried. Je vhodné aby boli vzorky získané pri rôznych podmienkach, podľa plánovaného použitia výsledného modelu.

### 6.1 Získavanie dát

Kedže táto práca nadväzuje na prácu [1], tvorí dataset v nej vytvorený základ pre ten použitý v tejto práci. Navyše bol rozšírený o niekoľko ďalších open source datasetov a vlastných anotovaných fotografií.

Na tvorbu datasetu bol použitý program *roboflow*, ktorý umožňuje načítavanie existujúcich datasetov, nahrávanie vlastných fotografií, anotáciu, spájanie dostupných datasetov, augmentácia datasetu a generovanie anotačných súborov rôznych formátov.

Po pridaní všetkých častí datasetu a kontrole správnosti anotácie, boli počty inštancií jednotlivých tried:

- **vták:** 9047
- **dron:** 1190
- **helikoptéra:** 509
- **hmyz:** 249
- **lietadlo:** 845

### 6.2 Úprava, rozdelenie a rozšírenie datasetu

Niektoré triedy sú reprezentované výrazne menším množstvom jednotlivých inštan- cii, čo je spôsobené nižšou dostupnosťou dát. Väčšina fotografií vtákov z vzdialenosťi pri ktorej by mali byť detekované pochopiteľne obsahuje niekolko desiatok jedincov, trieda vták je teda nadmerne zastúpená. Je možné tento problém čiastočne vyriešiť augmentáciou datasetu, čo program *roboflow* umožňuje priamo pri generovaní novej verzie. V budúcnosti je vhodnejším riešením ďalšie rozšírenie datasetu o snímky s objektami aktuálne nedostatočne reprezentovaných tried.

Aby bolo zabránené preučeniu modelu, musia byť dáta na ktorých je model učený rôzne od vylidačných a testovacích. Anotované obrazové dáta boli rozdelené do skupín s počtom:

- **trénovanie:** 2367
- **validácia:** 572
- **testovanie:** 290

Snímky boli predspracované upravením veľkosti na  $640 \times 640$  px a boli aplikované augmentácie, ktorými bol počet inštancií v trénovacej množine rozšírený na 7101:

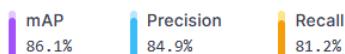
- Rotácia  $-30^\circ$  až  $+30^\circ$
- Zkosenie  $\pm 15^\circ$  vertikálne aj horizontálne
- Posun odtieňu  $-35^\circ$  až  $+35^\circ$
- Zmena saturácie  $-25\%$  až  $+25\%$
- Zmena expozície  $-20\%$  až  $+20\%$
- Rozmazanie do 0,8 px

## 6.3 Testovanie použiteľnosti datasetu

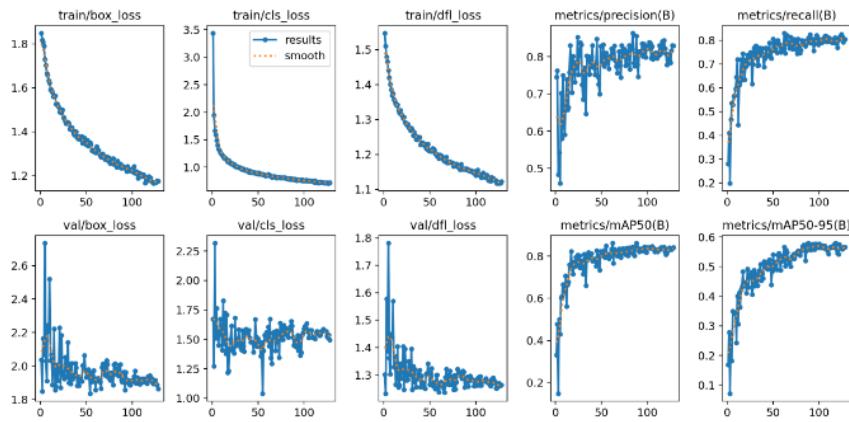
Použiteľnosť modelu bola testovaná naučením modelu typu *Roboflow 3.0 Object Detection* priamo v aplikácii *roboflow*, ktorý dosahoval úroveň presnosti 86.1 % pri zvolenej variante modelu *fast*. Pri dlhšom učení modelu určeného na reálne použitie sa dá predpokladať že dosiahnutá presnosť bude vyššia.

Obrázok 6.1 zobrazuje metriky presnosti naučeného modelu vypočítané v programe *roboflow*, tak ako sa v ňom zobrazujú. Vypočítané metriky sú:

- **Stredná priemerná presnosť (Mean Average Precision) (mAP)**  
priemer priemerných presností za všetky triedy
- **Presnosť (Precision)** =  $\frac{TP}{TP+FP}$ , značí s akou mieru správnej predikcie, v momente keď model nejakú hodnotu predikuje.
- **Senzitivita (Recall)** =  $\frac{TP}{TP+FN}$ , na druhú stranu predstavuje pomer počtu správnych predikcií, ku počtu skutočne správnych hodnôt danej triedy.

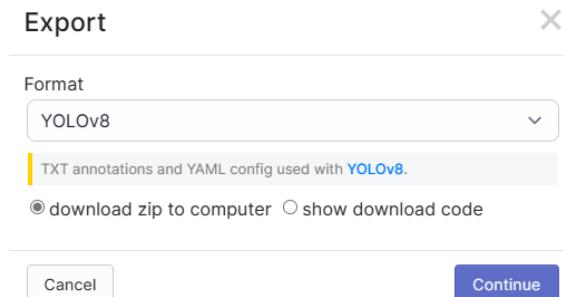


Obr. 6.1: Metriky presnosti naučeného modelu v programe roboflow



Obr. 6.2: Grafické zobrazenie priebehu učenia modelu roboflow

Výsledná vygenerovaná verzia datasetu je z programu exportovateľná v niekoľkých formátoch, napríklad pre učenie siete YOLO alebo pre použitie s knižnicou *tensorflow*. Pri exporte je stiahnutý komprimovaný priečinok obsahujúci obrazové dátá a anotáciu.



Obr. 6.3: Exportovanie vytvoreného datasetu

## 7 Návrh softvéru

Pri návrhu a tvorbe aplikácie systému detekcie lietajúcich objektov bol kladený dôraz na:

- Spustiteľnosť a použiteľnosť na počítači
- Ovládateľnosť dotykovým panelom
- Modulárnosť a rozšíritelnosť
- Spätná väzba a grafické zobrazenie výsledkov užívateľovi
- Možnosť používať, testovať a porovnávať rôzne metódy detekcie, klasifikácie a trasovania.

### 7.1 Použitý jazyk a knižnice

Použitím programovacieho jazyka *python*, bola získaná kompatibilita zo zariadeniami a operačnými systémami ktoré podporujú interpretér jazyka *python*, teda hlavne *Raspberry Pi* s nainštalovaným operačným systémom *Raspberry Pi OS* a osobné počítače, čo uľahčuje vývoj aplikácie a umožňuje jeho použitie na ľubovoľnom počítači.

Na tvorbu grafického rozhrania bola použitá knižnica *Rozhranie toolkitu Tcl/Tk (Tk interface)* (tkinter). Tá umožňuje vytvárať grafické používateľské rozhranie priamo z kódu v jazyku *python*. Obsahuje tiež objekty časovačov, vďaka ktorým je možné jednoducho a spoľahlivo ovládať kontinuálny beh aplikácie. Užívateľské rozhranie vytvorené knižnicou tkinter je kompatibilné s ovládaním klávesnicou a myšou, aj dotykovým displejom.

Knižnica *OpenCV* obsahuje implementáciu množstva algoritmov spracovania obrazu a počítačového videnia. Umožňuje základné operácie s videom aj jednotlivými snímkami ako načítanie, ukladanie a zobrazovanie, pričom podporuje rôzne formáty obrazových dát. Sofistikovanejšie úlohy ktoré knižnica rieši zahrnujú detekciu, klasifikáciu, sledovanie pohybu. Knižnica *OpenCV* je kompatibilná s viacerými programovacími jazykmi, vrátane jazyku *python*.

### 7.2 Štruktúra aplikácie

Aplikácia je zložená z modulov reprezentujúcich jednotlivé implementované metódy detekcie, klasifikácie či trasovania alebo riadiacich modulov ovládajúcich tok dát medzi modulmi. Základnými typmi modulov tvoriacich aplikáciu sú:

- **Hlavná aplikácia** - existuje len jedna varianta, je zodpovedná za ovládanie systému, užívateľské rozhranie a súštanie ostatných modulov. Pre každý z ďalších jednotlivých typov modulov je hlavnému modulu priradený zoznam

existujúcich modulov daného typu, z ktorých si užívateľ môže vybrať ten aktívny, ktorý sa aktuálne použije na daný účel.

- **Poskytovateľ obrazu** - slúži na získavanie obrazových dát. Každý cyklus začína vyžiadaním nového obrazu od aktuálne zvoleného poskytovateľa obrazu hlavným modulom. Existujú dve varianty:
  - **Video** - pri spustení aplikácia načíta všetky videá uložené v priečinku `./videos/` a pridá každému jeden objekt reprezentujúci dané video ako zdroj obrazu.
  - **Kamera** - ak je v systéme nainštalovaná *python* knižnica pre ovládanie kamery, je do zonamu poskytovateľov obrazu pridaný objekt reprezentujúci kameru.
- **Detektor objektov** - po získaní aktuálneho obrazu je aplikovaný zvolený algorytmus detekcie. Detektoru je predaný obraz, jeho výstupom je zoznam rámikov oblastí v ktorých sa potenciálne nachádza objekt. V prípade zvolenia metódy detekcie a klasifikácie v jednom kroku je preskočený výber klasifikátora a výstupom je aj klasifikácia.
- **Klasifikátor** - ak je zvolený algorytmus detekcie, sú oblasti s potenciálnym obsahom objektu predané klasifikátoru. Klasifikátor vráti hlavnému modulu zaradenie objektov do tried, po prípade označenie oblasti za pozadie. V hlavnej aplikácii je následne graficky zobrazené orámovanie a klasifikácia objektu.
- **Sledovač (Tracker)** - k ušetreniu výkonu je možné aplikovať detekciu a klasifikáciu len raz za určitý počet snímkov už a detekované a klasifikované objekty sledovať pomocou sledovačou z knižnice *OpenCV*.
- **Tester výsledkov** - v prípade že je zvolené video ako zdroj obrazu a v priečinku `./video_annotation/` je pridaný *Súbor s čiarkou rozdelenými hodnotami (Comma Separated Values)* (csv) s rovnakým menom ako video obsahujúci správne orámovanie, je spustený testovací modul ktorý ohodnocuje detekciu a klasifikáciu podľa určenej správnej hodnoty zo súboru.

Jednotlivé moduly sú vytvárané dedením od abstraktnej triedy predstavujúcej rozhranie modulu s aplikáciou. Pre každý z typov modulu je pripravená jedna abstraktná rodičovská trieda, obsahujúca niektoré zdieľané premenné, ako je názov modulu a funkcie ktoré musí modul implementovať. Povinnou funkciou je napríklad funkcia vyžiadania si ďalšieho snímku pre modul poskytovateľa obrazu.

## 7.3 Implementácia

V aplikácii sú ako modul detekcie a klasifikácie aktuálne implementovaný model YOLO, pričom aplikácia automaticky pridá detekované naučené modely. Tie musia

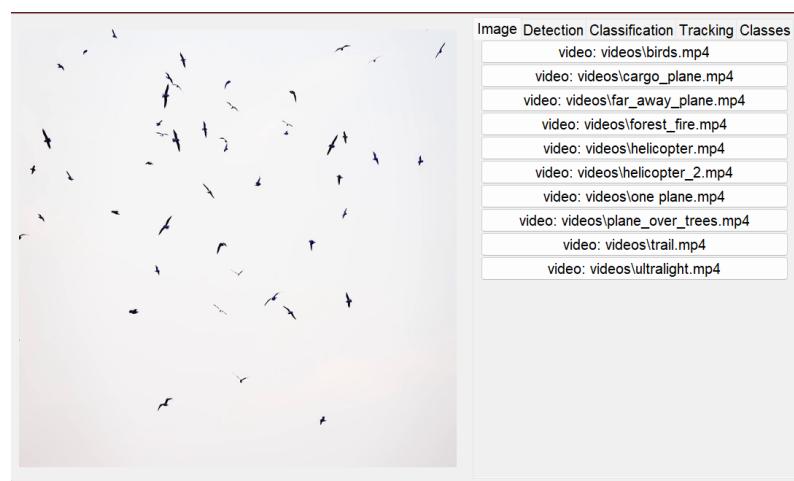
byť uložené v priečinku `./yolo/[model]/`, kde je `[model]` nahradené názvom modelu. V tomto priečinku je uložený model samotný v súbore `model.onnx` a csv súbor so zoznamom názvov a farebných označení tried vo formáte  
"*názov*", "*červená*", "*zelená*", "*modrá*"  
s názvom `classes.csv`. Aplikácia načíta všetky nájdené modely spolu s popisom tried a vloží ich do zoznamu detektorov.

## 7.4 Návrh grafického rozhrania

Grafické užívateľské rozhranie (*Graphical User Interface*) (GUI) umožňuje intuitívne ovládanie aplikácie a zobrazovanie výsledkov detekcie, klasifikácie a sledovania. Ovládanie jednotlivých modulov aplikácie je rozdelené do záložiek. Každá záložka umožňuje zvoliť ktorá z inštancii modulu každého z typov sa aktuálne používa.

V ľavej časti obrazovky je zobrazený aktuálny snímok, s výstupom detekcie a klasifikácie. Zobrazujú sa v ňom orámovania detekovaných objektov, ich priradenie do tried a hodnotenie dôveryhodnosti klasifikácie. Ak existuje súbor s anotáciou správneho orámovania a klasifikácie pre aktuálne prehrávané video, sú zobrazené aj tie.

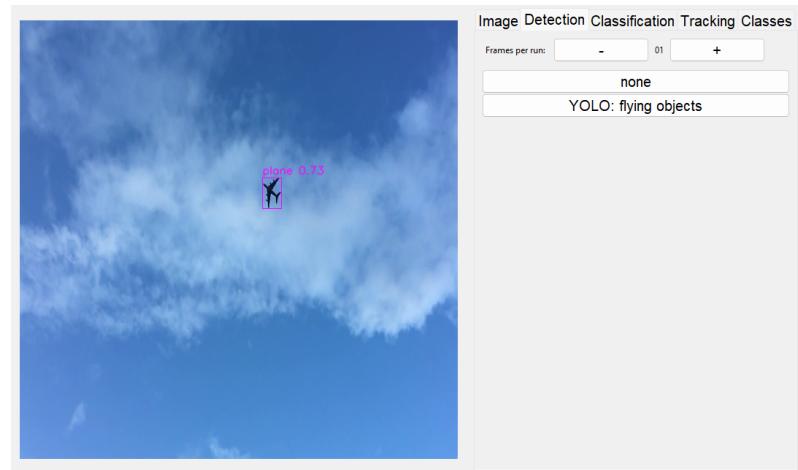
V prvej záložke je užívateľovi umožnené zvoliť zdroj obrazu. Ak je aplikácia spustená na *Raspberry Pi* a je nainštalovaná kižnica ovládania kamery, je do zoznamu zdrojov pridaná kamera. Pri spustení aplikácie sú načítané videá z priečinku `./videos` a sú zobrazené v zozname na záložke zdrojov spolu s kamerou. Kliknutím na názov zdroja je zdroj zvolený a užívateľovi sa z neho začne zobrazovať obraz.



Obr. 7.1: Záložka na výber poskytovateľa obrazu

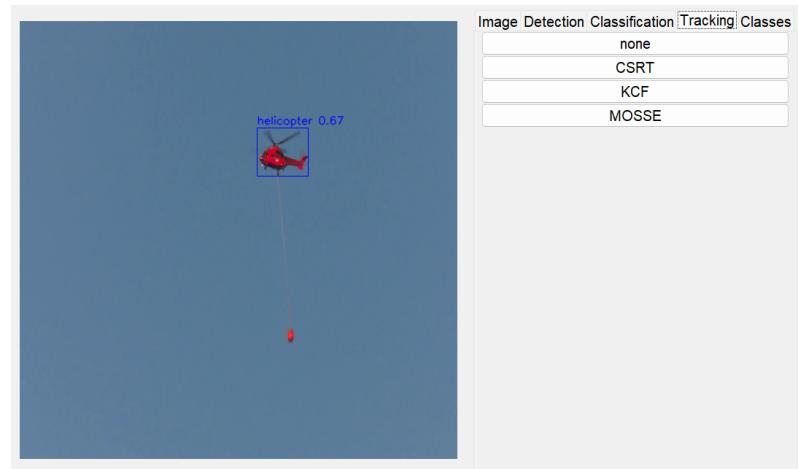
V druhej záložke výberu detektora si užívateľ vyberie metódu detekcie, alebo metódu súčasnej detekcie a klasifikácie, pri ktorej nie je použiteľný výber klasifikátoru

na ďalšej záložke. Zároveň je na tejto záložke nastaviteľná perióda spúšťania detektie, medzi snímkami na ktorých je detekcia spustená sa buď zobrazované ohraničenie nemení, alebo je akualizované sledovačom.



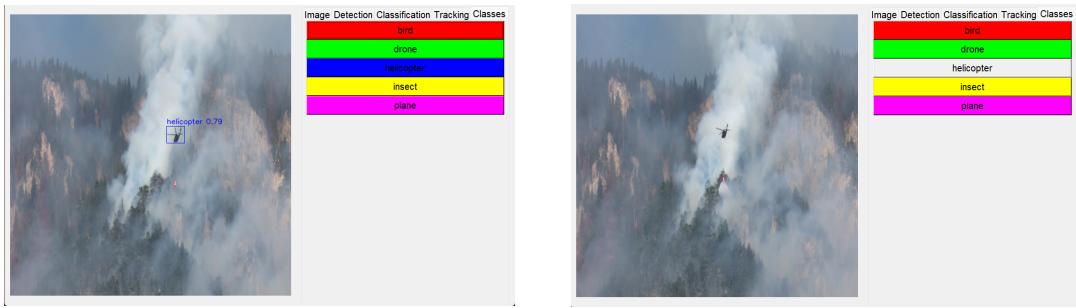
Obr. 7.2: Záložka na výber metódy detekcie objektov

V záložke sledovania objektov je umiestnený výber zo sledovačov implementovaných v knižnici *OpenCV*. Sledovač je použitý na snímkoch medzi spustením detektie.



Obr. 7.3: Záložka na výber metódy sledovania objektov

Posledná záložka obsahuje menu z možnosťou zapnúť a vypnúť zobrazenie detekcie konkrétnych tried objektov.



Obr. 7.4: Záložka na výber detekovaných tried

# **Závěr**

Shrnutí studentské práce.

## **Literatúra**

- [1] JUREČKA, Tomáš. Detekce a klasifikace létajících objektů. Brno, 2021. Diplomová práca.

# Zoznam symbolov a skratiek

<b>ReLU</b>	Usmernená lineárna jednotka
<b>CNN</b>	Konvolučné neurónové siete (Convolutional neural network)
<b>R-CNN</b>	Konvolučné neurónové siete založené na regiónoch (Region-based CNN)
<b>RoI</b>	Regióny záujmu (Regions of Interest)
<b>SVM</b>	Metódy podporných vektorov (Support Vector Machine)
<b>RPN</b>	Siet návrhu regiónov (Region Proposal Network)
<b>YOLO</b>	You only look once
<b>IOU</b>	Pomer prieniku voči zjednoteniu (Intersection over Union)
<b>tkinter</b>	Rozhranie toolkitu Tcl/Tk (Tk interface)
<b>csv</b>	Súbor s čiarkou rozdelenými hodnotami (Comma Separated Values)
<b>GUI</b>	Grafické užívateľské rozhranie (Graphical User Interface)
<b>mAP</b>	Stredná priemerná presnosť (Mean Average Precision)

# **Zoznam príloh**

<b>A Některé příkazy balíčku thesis</b>	<b>40</b>
A.1 Příkazy pro sazbu veličin a jednotek . . . . .	40
A.2 Příkazy pro sazbu symbolů . . . . .	40
<b>B Druhá příloha</b>	<b>41</b>
<b>C Příklad sazby zdrojových kódů</b>	<b>42</b>
C.1 Balíček listings . . . . .	42
<b>D Obsah elektronické přílohy</b>	<b>45</b>

# A Některé příkazy balíčku thesis

## A.1 Příkazy pro sazbu veličin a jednotek

Tab. A.1: Přehled příkazů pro matematické prostředí

Příkaz	Příklad	Zdroj příkladu	Význam
<code>\textind{...}</code>	$\beta_{\max}$	$\$\\beta\\_\\textind{max}\\$$	textový index
<code>\const{...}</code>	$U_{\text{in}}$	$\$\\const{U}\\_\\textind{in}\\$$	konstantní veličina
<code>\var{...}</code>	$u_{\text{in}}$	$\$\\var{u}\\_\\textind{in}\\$$	proměnná veličina
<code>\complex{...}</code>	$u_{\text{in}}$	$\$\\complex{u}\\_\\textind{in}\\$$	komplexní veličina
<code>\vect{...}</code>	$\mathbf{y}$	$\$\\vect{y}\\$$	vektor
<code>\mat{...}</code>	$\mathbf{Z}$	$\$\\mat{Z}\\$$	matice
<code>\unit{...}</code>	kV	$\$\\unit{kV}\\$$ či $\$\\unit{kV}\\$$	jednotka

## A.2 Příkazy pro sazbu symbolů

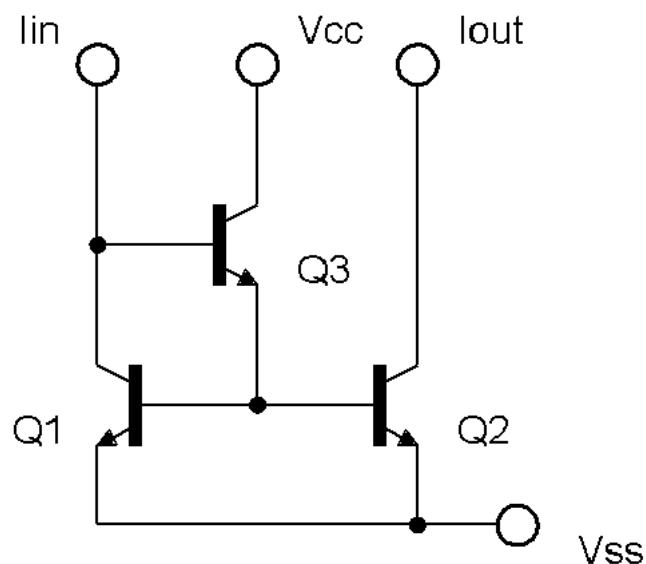
- `\E`, `\eul` – sazba Eulerova čísla: e,
- `\J`, `\jmag`, `\I`, `\imag` – sazba imaginární jednotky: j, i,
- `\dif` – sazba diferenciálu: d,
- `\sinc` – sazba funkce: sinc,
- `\mikro` – sazba symbolu mikro stojatým písmem<sup>1</sup>: μ,
- `\uppi` – sazba symbolu π (stojaté řecké pí, na rozdíl od `\pi`, což sází π).

Všechny symboly jsou určeny pro matematický mód, vyjma `\mikro`, jenž je použitelný rovněž v textovém módu.

---

<sup>1</sup>znak pochází z balíčku `textcomp`

## B Druhá příloha



Obr. B.1: Zlepšené Wilsonovo proudové zrcadlo.

Pro sazbu vektorových obrázků přímo v L<sup>A</sup>T<sub>E</sub>Xu je možné doporučit balíček TikZ. Příklady sazby je možné najít na TeXample. Pro vyzkoušení je možné použít programy QTikz nebo TikzEdt.

## C Příklad sazby zdrojových kódů

### C.1 Balíček *listings*

Pro vysázení zdrojových souborů je možné použít balíček *listings*. Balíček zavádí nové prostředí *lstlisting* pro sazbu zdrojových kódů, jako například:

```
\section{Balíček lstlisting}
Pro vysázení zdrojových souborů je možné použít
    balíček \href{https://www.ctan.org/pkg/listings}%
    {\texttt{listings}}.
Balíček zavádí nové prostředí \texttt{lstlisting} pro
    sazbu zdrojových kódů.
```

Podporuje množství programovacích jazyků. Kód k vysázení může být načítán přímo ze zdrojových souborů. Umožňuje vkládat čísla řádků nebo vypisovat jen vybrané úseky kódu. Např.:

Zkratky jsou sázeny v prostředí *acronym*:

6 \begin{acronym}[KolikMista]

Šířka textu volitelného parametru *KolikMista* udává šířku prvního sloupce se zkratkami. Proto by měla být zadávána nejdélší zkratka nebo symbol. Příklad definice zkratky **symfvz!** je na výpisu C.1.

Výpis C.1: Ukázka sazby zkratky

```
69  % \acro{symfvz}          % název
70  %   [\ensuremath{f_{\text{\texttt{uz}}}}] % symbol
71  %   {vzorkovací kmitočet}      % popis
72  %
```

Ukončení seznamu je provedeno ukončením prostředí:

26 \end{acronym} {Metody podporných vektorov (Support Vector Machine)}

#### Poznámka k výpisům s použitím volby jazyka *czech* nebo *slovak*:

Pokud Váš zdrojový kód obsahuje znak spojovníku `-`, pak překlad může skončit chybou. Ta je způsobena tím, že znak `-` je v českém nebo slovenském nastavení balíčku *babel* tzv. aktivním znakem. Přepněte znak `-` na neaktivní příkazem `\shorthandoff{-}` těsně před výpisem a hned za ním jej vratte na aktivní příkazem `\shorthandon{-}`. Podobně jako to je ukázáno ve zdrojovém kódu *šablony*.

Na výpisu C.2 naleznete příklad kódu pro Matlab, na výpisu C.3 zase pro jazyk C.

Výpis C.2: Příklad Schur-Cohnova testu stability v prostředí Matlab.

```
1 %% Priklad testovani stability filtrov  
2  
3 % koeficienty polynomu ve jmenovateli  
4 a = [ 5, 11.2, 5.44, -0.384, -2.3552, -1.2288];  
5 disp( 'Polynom:' ); disp(poly2str( a, 'z' ))  
6  
7 disp('Kontrola pomocí korenů polynomu:');  
8 zx = roots( a );  
9 if( all( abs( zx ) < 1 ))  
10    disp('System je stabilní')  
11 else  
12    disp('System je nestabilní nebo na mezi stability');  
13 end  
14  
15 disp(' '); disp('Kontrola pomocí Schur-Cohn:');  
16 ma = zeros( length(a)-1,length(a) );  
17 ma(1,:) = a/a(1);  
18 for( k = 1:length(a)-2)  
19    aa = ma(k,1:end-k+1);  
20    bb = fliplr( aa );  
21    ma(k+1,1:end-k+1) = (aa-aa(end)*bb)/(1-aa(end)^2);  
22 end  
23  
24 if( all( abs( diag( ma.' ) ) ) )  
25    disp('System je stabilní')  
26 else  
27    disp('System je nestabilní nebo na mezi stability');  
28 end
```

Výpis C.3: Příklad implementace první kanonické formy v jazyce C.

```
// první kanonická forma
short fxdf2t( short coef[][][5], short sample)
{
    static int v1[SECTIONS] = {0,0}, v2[SECTIONS] = {0,0};
    int x, y, accu;
    short k;

    x = sample;
    for( k = 0; k < SECTIONS; k++){
        accu = v1[k] >> 1;
        y = _sadd( accu, _smpy( coef[k][0], x));
        y = _ssh1(y, 1) >> 16;

        accu = v2[k] >> 1;
        accu = _sadd( accu, _smpy( coef[k][1], x));
        accu = _sadd( accu, _smpy( coef[k][2], y));
        v1[k] = _ssh1( accu, 1);

        accu = _smpy( coef[k][3], x);
        accu = _sadd( accu, _smpy( coef[k][4], y));
        v2[k] = _ssh1( accu, 1);

        x = y;
    }
    return( y);
}
```

## D Obsah elektronické přílohy

Elektronická příloha je často nedílnou součástí semestrální nebo závěrečné práce. Vkládá se do informačního systému VUT v Brně ve vhodném formátu (ZIP, PDF ...).

Nezapomeňte uvést, co čtenář v této příloze najde. Je vhodné komentovat obsah každého adresáře, specifikovat, který soubor obsahuje důležitá nastavení, který soubor je určen ke spuštění, uvést nastavení kompilátoru atd. Také je dobré napsat, v jaké verzi software byl kód testován (např. Matlab 2018b). Pokud bylo cílem práce vytvořit hardware zařízení, musí elektronická příloha obsahovat veškeré podklady pro výrobu (např. soubory s návrhem DPS v Eagle).

Pokud je souborů hodně a jsou organizovány ve více složkách, je možné pro výpis adresářové struktury použít balíček `dirtree`.

```
/..... kořenový adresář přiloženého archivu
  logo ..... loga školy a fakulty
    BUT_abbreviation_color_PANTONE_EN.pdf
    BUT_color_PANTONE_EN.pdf
    FEEC_abbreviation_color_PANTONE_EN.pdf
    FEKT_zkratka_barevne_PANTONE_CZ.pdf
    UTKO_color_PANTONE_CZ.pdf
    UTKO_color_PANTONE_EN.pdf
    VUT_barevne_PANTONE_CZ.pdf
    VUT_symbol_barevne_PANTONE_CZ.pdf
    VUT_zkratka_barevne_PANTONE_CZ.pdf
  obrazky ..... ostatní obrázky
    soucastky.png
    spoje.png
    ZlepseWilsonovoZrcadloNPN.png
    ZlepseWilsonovoZrcadloPNP.png
  pdf ..... pdf stránky generované informačním systémem
    student-desky.pdf
    student-titulka.pdf
    student-zadani.pdf
  text ..... zdrojové textové soubory
    literatura.tex
    prilohy.tex
    reseni.tex
    uvod.tex
    vysledky.tex
    zaver.tex
    zkratky.tex
  sablona-obhaj.tex ..... hlavní soubor pro sazbu prezentace k obhajobě
  sablona-prace.tex ..... hlavní soubor pro sazbu kvalifikační práce
  thesis.sty ..... balíček pro sazbu kvalifikačních prací
```