



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

Táto práca sa zaobrá využitím počítačového videnia v oblasti detektie, trasovania a klasifikácie lietajúcich objektov v reálnom prostredí. Cieľom je vytvoriť robustný systém, schopný efektívne pracovať v nepriaznivých podmienkach a presne identifikovať rôzne typy objektov na oblohe. Práca postupuje od teoretických základov, výberu metód, až po návrh a implementáciu systému počítačového videnia.

KĽÚČOVÉ SLOVÁ

počítačové videnie, detekcia objektov, trasovanie, klasifikácia, lietajúce objekty, mikro-počítače, dataset

ABSTRACT

This work deals with the use of computer vision in the field of detection, tracking and classification of flying objects in a real environment. The goal is to create a robust system, capable of working effectively in adverse conditions and accurately identifying different types of objects in the sky. The work progresses from theoretical foundations, choice of methods, to the design and implementation of a computer vision system.

KEYWORDS

computer vision, object detection, tracking, classification, flying objects, microcomputers, dataset

FRANKA, Jakub. *Systém pro sledování a klasifikaci objektů na obloze*. Diplomová 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: Diplomová 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

Chcel by som poďakovať pani Ing. Ilone Janákovej Ph.D. za jej cenné vedenie, trpežlivosť a podporu pri mojej technickej diplomovej práci.

Obsah

Úvod	13
1 Detekcia objektov v obraze	15
1.1 Prahovanie	15
1.1.1 Globálne prahovanie	16
1.1.2 Lokálne prahovanie	17
1.2 Detekcia hrán	18
1.2.1 Detekcia horizontu pomocou Houghovej transformácie	20
1.3 Detekcia pohybu	20
2 Popis objektov	23
2.1 Popis tvaru	23
2.2 Fotometrické príznaky	24
3 Klasifikácia	25
3.1 Klasifikátory	25
3.2 Konvolučné neurónové siete	25
4 Detekcia a klasifikácia v jednom kroku	27
4.1 R-CNN	27
4.2 YOLO	28
4.2.1 Non-Maximum Suppression	28
5 Ukladanie naučených modelov	31
5.1 ONNX	31
5.2 Keras	31
6 Vyhodnotenie kvality detekcie	32
6.1 Kvalita orámovania	32
6.2 Kvalita klasifikácie	33
6.2.1 Matica zámien	33
6.2.2 Metriky vychádzajúce z matice zámien	33
6.3 Výkon detekcie	35
6.3.1 Snímková frekvencia	35
6.3.2 Oneskorenie	35
7 Návrh usporiadania hardvérových komponentov	36
7.1 Jednodoskový počítač	36
7.2 Kamera	36

7.3	Ovládacie prvky	37
7.4	Uchytenie kamery	38
7.5	Kryt zariadenia	38
8	Tvorba datasetu	40
8.1	Získavanie dát	40
8.2	Úprava, rozdelenie a rozšírenie datasetu	43
8.3	Testovanie použiteľnosti datasetu	43
9	Návrh softvéru	45
9.1	Použitý jazyk a knižnice	45
9.2	Štruktúra aplikácie	45
9.2.1	Štruktúra modulu aplikácie	46
9.2.2	Implementované moduly	47
9.3	Automatické spúšťanie aplikácie	49
9.4	Návrh grafického rozhrania	49
10	Kontrola a porovnanie prístupov	52
10.1	Zber dát	52
10.2	Úprava dát	52
10.3	Vyhodnotenie	55
Závěr		59
Literatúra		60
Zoznam symbolov a skratiek		61
Zoznam príloh		62

Zoznam obrázkov

1.1	Funkcia prahovania s prahom 0.5 pre svetlé pozadie	15
1.2	Jednoduché prahovanie	16
1.3	Automatické prahovanie (podľa známeho rozloženia)	17
1.4	Adaptívne prahovanie (podľa známeho rozloženia)	17
1.5	Cannyho hranový detektor	19
1.6	Detekcia horizontu pomocou Houghovej transformácie	20
1.7	Postup aktualizovania dynamického modelu pozadia	22
1.8	Odcítanie pozadia	22
3.1	Príklad architektúry konvolučnej neurónovej siete	26
4.1	Vstup a výsledok <i>Potlačenie nemaximálnej hodnoty (Non-Maximum Suppression)</i> (NMS)	29
4.2	Algoritmus Non-Maximum Suppression pre orámovanie objektov	30
6.1	Orámovanie na anotovanom snímku	32
6.2	Nájdenie prvého bodu prieniku orámovaní	33
6.3	Rozdelenie prvkov matice	34
6.4	Snímková frekvencia a oneskorenie	35
7.1	Rozlíšenie kamery	37
7.2	Návrh nastaviteľného uchytenia kamery	38
7.3	Bloková schéma zariadenia	38
7.4	Pohľad na zariadenie spredu	39
7.5	Pohľad na zariadenie zo zadu, s odnímateľnými krytmi	39
8.1	Založenie roboflow projektu	40
8.2	Definícia tried roboflow projektu	41
8.3	Anotácia snímku v programe roboflow	41
8.4	Generované verzie datasetu v roboflow	42
8.5	Ukážka niekoľkých snímok z datasetu	42
8.6	Metriky presnosti naučeného modelu v programe roboflow	44
8.7	Grafické zobrazenie priebehu učenia modelu roboflow	44
8.8	Exportovanie vytvoreného datasetu	44
9.1	Príklad výstupu pri spustení aplikácie	47
9.2	Sekcia [Service] konfigurácie automatického spustenia aplikácie	49
9.3	Záložka na zobrazenie výstupov detekcie	50
9.4	Záložka na nastavenie modulov	51
10.1	Užívateľské rozhranie programu na anotáciu videa	53
10.2	Definícia tried v anotovanom videu	53
10.3	Výber upravovaného orámovania	54
10.4	Úprava vybraného orámovania	54

10.5 Odstránenie sekvencie snímkov	55
10.6 Uloženie a úprava videa	55
10.7 Načítanie videí na porovnanie modelov	56
10.8 Načítanie modelov na porovnanie	57
10.9 Aplikácia na porovnanie s načítanými videami a modelmi	57
10.10 Prebiehajúci výpočet porovnania	58
10.11 Zobrazenie výsledkov porovnania	58

Zoznam tabuliek

Zoznam výpisov

Úvod

V súčasnej dobe hrá počítačové videnie dôležitú rolu v mnohých aplikáciách. Táto práca sa zameriava na jeho využitie pri návrhu systému schopného detektovať, trasať a klasifikovať objekty lietajúce na oblohe. Cieľom je vytvoriť robustný systém schopný nasadenia v reálnych nepriaznivých podmienkach so schopnosťou detektie a klasifikácie rôznych typov živých aj umelo vytvorených lietajúcich objektov. Je požadovaná možnosť použitia rôznych prístupov spracovania vstupných dát a porovnanie ich výkonu a presnosti. Pri návrhu systému je nutné zvážiť obmedzený výkon výpočtovej jednotky systému a voliť jednoduchšie.

Systém bude získať obrazové dátá pomocou statickej kamery sledujúcej časť oblohy. Časť obrazu môže obsahovať aj horizont. Kamera v bude reálnom čase zachytávať sekvenču snímkov. Na nich systém musí rozpoznať pohybujúce sa objekty na pozadí, ktoré sa môže s postupom času tiež pohybovať či meniť, či už kvôli pohybujúcim oblakom alebo zmene svetelných podmienok.

Detektované objekty budú klasifikované do piatich tried: vták, hmyz, lietadlo, helikoptéra, dron. Kvôli rôznym farbám rôznych inštancii týchto objektov môžu mať objekty vyšší, ale aj nižší jas ako pozadie.

Táto práca bude postupovať od teoretických základov do konkrétneho návrhu a implementácie popísaného systému počítačového videnia.

Prvá kapitola sa venuje základným princípm detektie objektov v obraze. Sú tu preskúmané existujúce metódy s porovnaním ich výhod a nevýhod, čo povedie k výberu vhodných metód pre navrhovaný systém.

V druhej kapitole bude popísané získavanie charakteristík popisujúcich jednotlivé detektované objekty. Zohľadňujú sa tu aspekty ako tvar, farba a ďalšie príznaky, na ktorých záleží klasifikácia v ďalšom kroku.

Na základe popisu objektov bude v ďalšej kapitole popísaná problematika klasifikácie, teda rozpoznávania a zaradenia detektovaných objektov do definovaných tried podľa ich typu (lietadlo, helikoptéra, vták, hmyz, ...).

Štvrtá kapitola predstaví moderné metódy integrujúce detekciu a klasifikáciu do jedného procesu. Cieľom je optimalizácia efektivity a jednoduchosti architektúry systému.

Následne bude navrhnutý a odôvodnený výber hardvérových komponentov systému. Budú porovnané mikropočítače vhodné na aplikáciu v tejto práci a zvolené vhodné periférie na získavanie dát a užívateľské rozhranie.

Pre správne naučenie modelov použitých v systéme je nevyhnutné vytvoriť dostačne rozsiahlu a správne navrhnutú anotovanú množinu dát, v prípade algoritmov počítačového videnia ide o databázu snímkov s označenými a triedenými objektami. Šiesta kapitola sa zameriava na tvorbu tejto databázy (datasetu) a jej validáciu.

V poslednej kapitole bude zdokumentovaný návrh softvéru pre systém detektie, klasifikácie a trasovania objektov. V tejto kapitole bude priblížená štruktúra aplikácie, nástroje použité pri vývoji a návrh jej užívateľského rozhrania.

1 Detekcia objektov v obraze

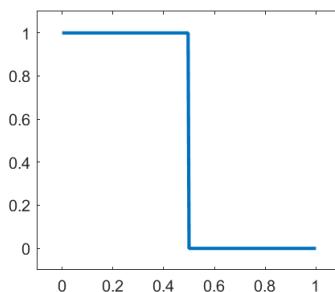
Pre niektoré techniky klasifikácie objektov je nutné najprv v obraze tieto objekty nájsť. Existuje k tomu veľké množstvo metodík, pri čom každá z nich má svoje výhody a nevýhody. Je teda nutnosťou zistiť, ktorá metóda najviac využíva použitie pre túto prácu. Je nutnosťou zvoliť jednoduchšie metódy vzhľadom na výpočtový výkon zariadenia.

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 oddeľujúceho objekty od pozadia. [3] [5]

Prevod farebného obrazu do šedotónového s cieľom získať pre každý pixel jednu hodnotu jasu je možný nie len výpočtom priemerného jasu farebných zložiek pixelu, ale aj ich lineárной kombináciou s váhami určenými podľa predpokladanej farby pozadia. V prípade detektie objektov na oblohe, ktorá tvorí pozadie prevažne modrej farby, dáva zmysel uvažovať modrú farbu s vyšším významom. Vďaka tomu môžu byť zvýraznené objekty, ktorých farba sa líši od modrej farby pozadia. Inou možnosťou je uvažovať len jednu farebnú zložku, teda modrú, čím by bol ušetrený výpočtový výkon pred tým využitý pri výpočte súčtov zložiek.



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

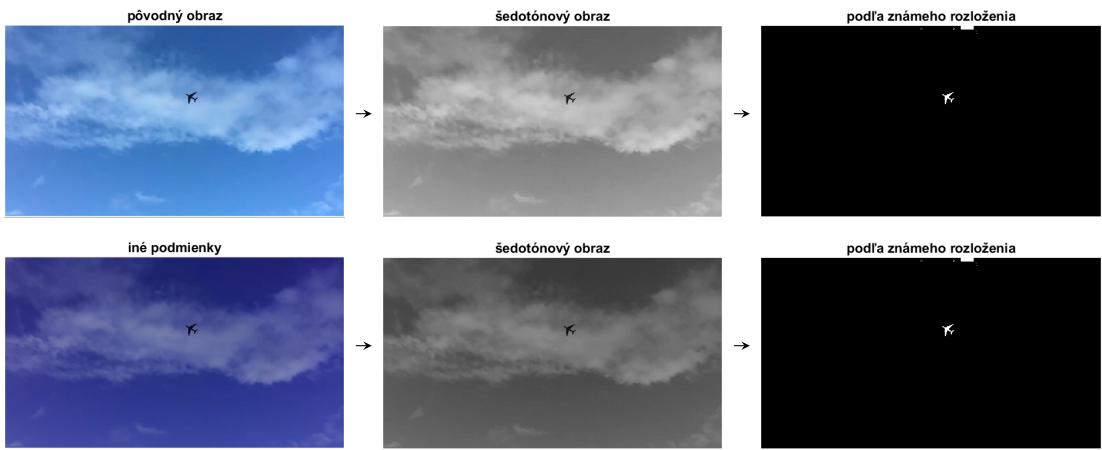
1.1.1 Globálne prahovanie

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. [3]

- **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 má 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 algoritmus 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 vzájomný rozptyl pre všetky možné hodnoty prahu a vyberá ten optimálny.



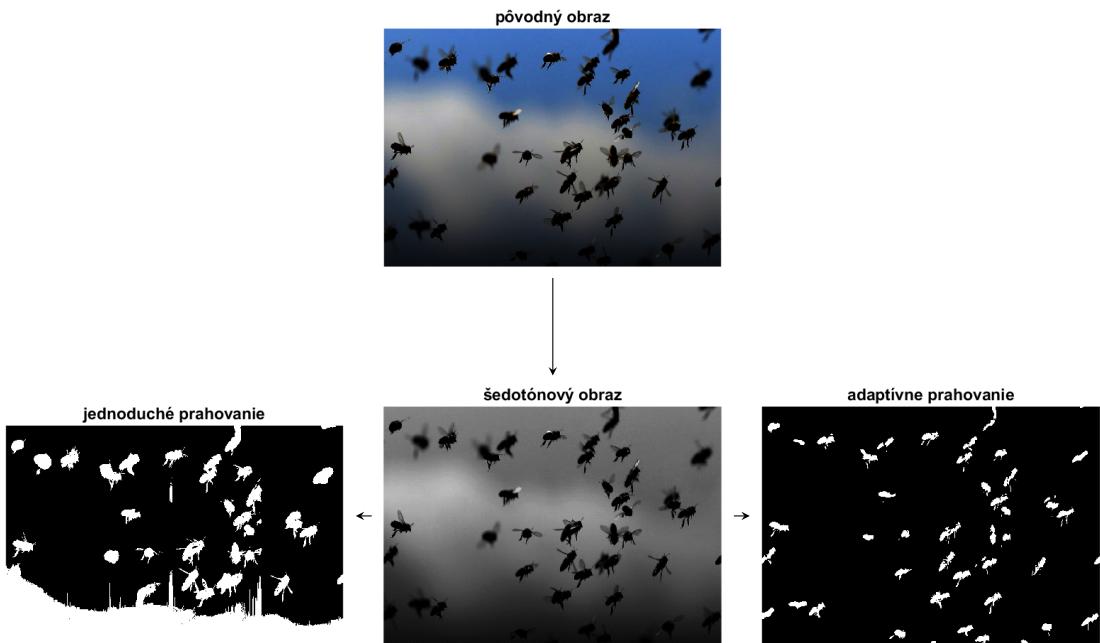
Obr. 1.2: Jednoduché prahovanie



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 výpočtovú náročnosť ako jednoduchšie globálne prahovanie, kedže je nutné určiť prah pre každý bod obrazu samostatne, je však robustnejšie voči nevhodnému pozadiu. [5]



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 aproximujú 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. [6]

$$\begin{array}{ccc} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ \text{Prewittov} & \text{Sobelov} & \text{Laplaceov} \\ \text{operátor} & \text{operátor} & \text{operátor} \end{array}$$

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

LoG operátor

Komplexnejšou metódou detekcie hrán je takzvaný *Cannyho hranový detektor*. Ide o proces vo viacerých krokoch [6]. Pre jeden vstupný obraz sa postupuje nasledovne:

1. Vstupný obraz býva väčšinou filtrovaný z dôvodu zníženia citlivosti na šum v obraze. Vhodné je k tomu napríklad Gaussove rozmažanie.
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:

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

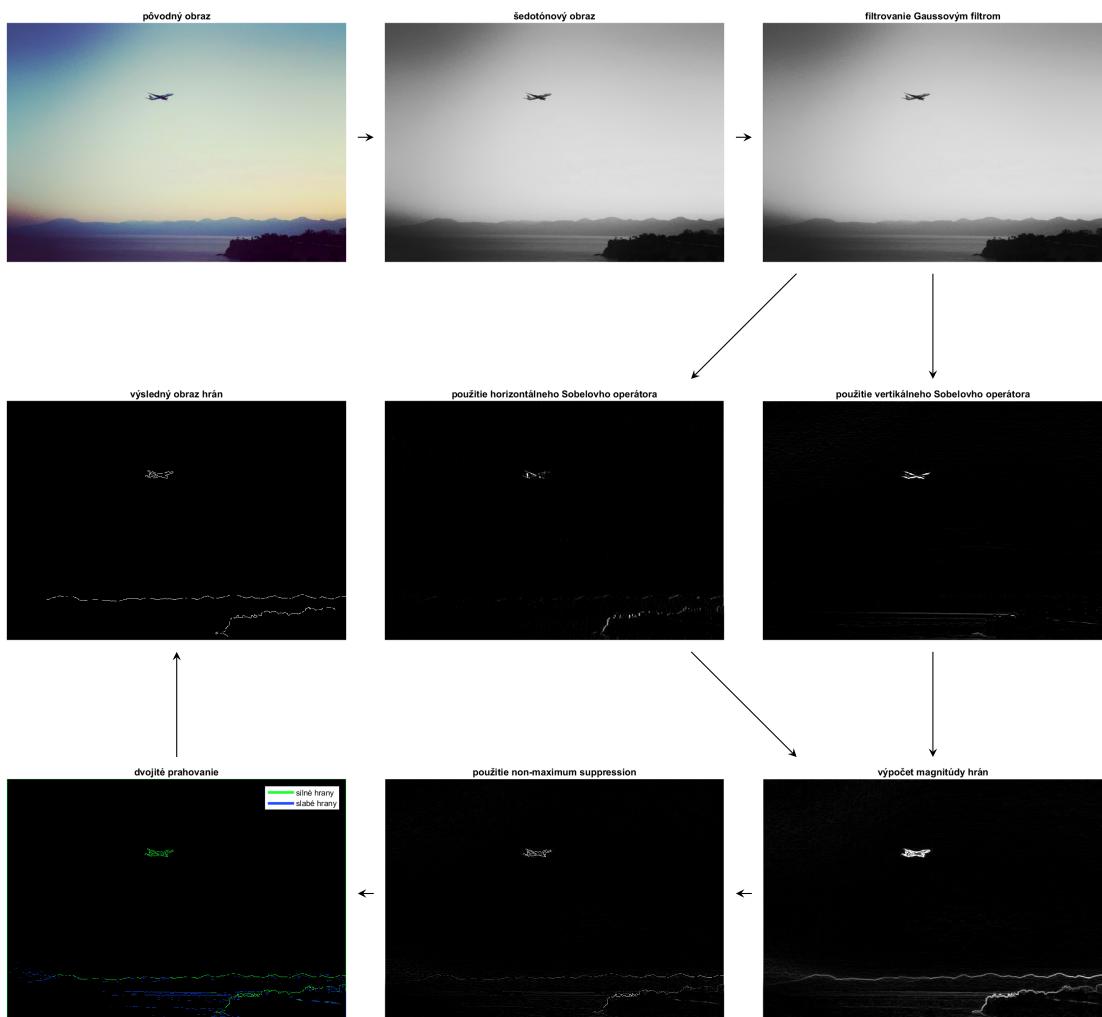
4. Ku zníženiu redundancie detekovaných hrán sa použije algoritmus NMS. Použitím tohto algoritmu dôjde k zúženiu hrán, pričom sa ponechajú len tie časti

hrán, ktorých magnitúda je väčšia ako v ich okolí.

5. Dvojitým prahovaním sa v obraze hrán nájdu 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.



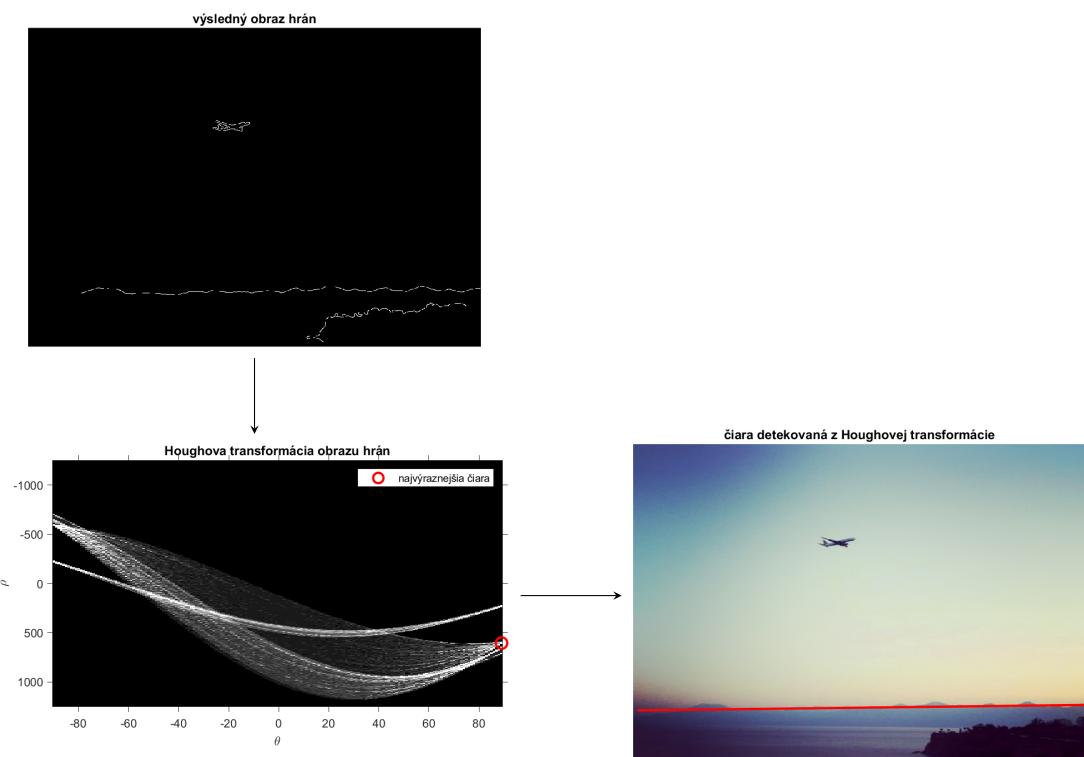
Obr. 1.5: Cannyho hranový detektor

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

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

1.2.1 Detekcia horizontu pomocou Houghovej transformácie

Pokiaľ 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 [2]. 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 detektovať ich práve podľa 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, vyhodnocuje sa len v miestach, kde $I_1(x, y) > I_2(x, y)$.

$$D(x, y) = \begin{cases} 0 & I_1(x, y) - I_2(x, y) < \epsilon \\ 1 & I_1(x, y) - I_2(x, y) \geq \epsilon \end{cases} \quad (1.1)$$

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

$$D(x, y) = \begin{cases} 0 & |I_1(x, y) - I_2(x, y)| < \epsilon \\ 1 & |I_1(x, y) - I_2(x, y)| \geq \epsilon \end{cases} \quad (1.2)$$

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

$$D(x, y) = \sum_{i=1}^{N-1} \omega_i \cdot D_i(x, y) \quad (1.3)$$

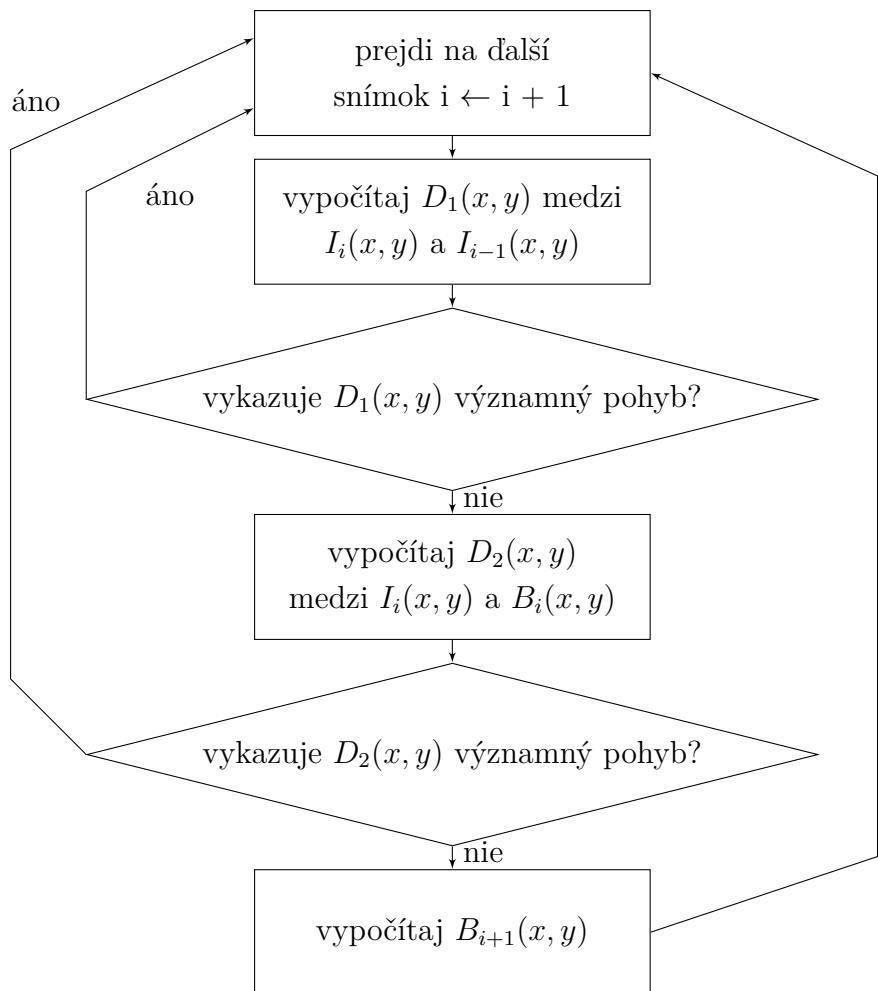
Komplexnejšou metódou detekcie pohybu je vytvorenie rozdielového snímku nie medzi nasledujúcimi snímkami, ale aktuálnym snímkom a vytvoreným modelom pozadia. Ten je možné zostaviť:

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

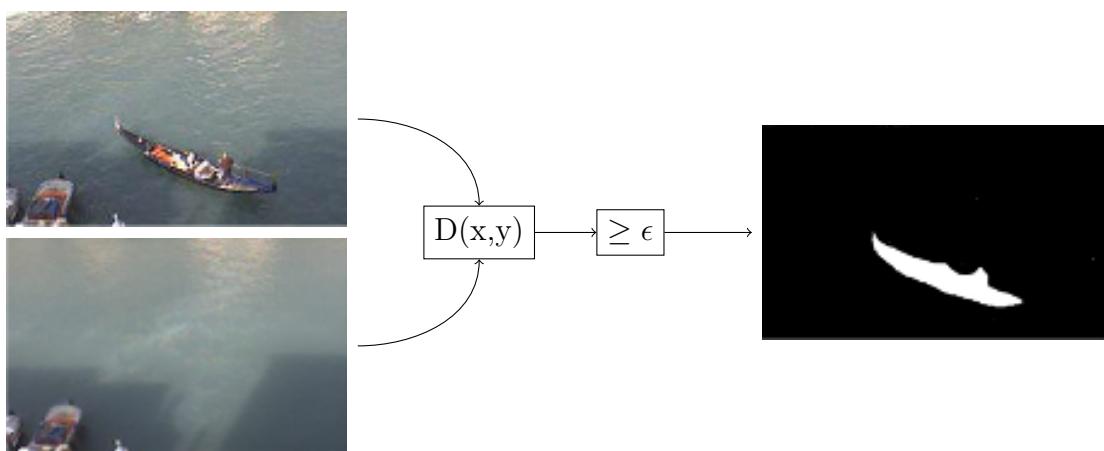
Tým sa model pozadia postupne prispôsobuje malým zmenám v prostredí.

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. [7]



Obr. 1.7: Postup aktualizovania dynamického modelu pozadia



Obr. 1.8: Odčítanie pozadia

2 Popis objektov

Po nájdení objektov v obraze nasleduje ich popis pomocou príznakového vektoru. Ten predstavuje matematickú reprezentáciu segmentu obrazu vektorom o ľubovoľnom počte rozmerov. Jeho prvky popisujú farbu, textúru alebo tvar objektu. Samotný objekt nemusí byť rekonštruovateľný z jeho popisu, cieľom je podľa neho rozlísiť od seba dva rôzne objekty.

Vypočítané hodnoty príznakového vektoru majú niekoľko využití:

- **Párovanie objektov** v dvoch snímkoch, teda určenie, ktoré body alebo objekty v jednom snímku korelujú s tými v druhom snímku. Výhodou je to pri vyhľadávaní rovnakého objektu v po sebe nasledujúcich snímkoch pri sledovaní pohybu, či pri hľadaní rovnakého objektu v obraze z dvoch kamier.
- **Klasifikácia** objektov v obraze. Objekty rôznych tried majú často rôzne hodnoty určitých príznakov, je však nutné určiť, ktoré príznaky sú pre klasifikáciu relevantné.

2.1 Popis tvaru

Jedny z najjednoduchších príznakov vychádzajú z tvaru objektu, používajú sa s binárnym obrazom získaným napríklad prahovaním. [4] Sú to napríklad:

- **Obsah** oblasti objektu, teda počet pixelov predstavujúcich objekt v binárnom obraze.
- **Obvod** je počet pixelov na hrane tvaru objektu. Diagonálne hrany môžu byť pri výpočte obvodu nadhodnotené v prípade, ak počítame s 4-okolím alebo podhodnotené, ak počítame s 8-okolím bodov na hrane. Je teda správnejšie počítať 1 pixel pre pravouhlé hrany a $\sqrt{2}$ pre diagonálne hrany.
- **Kompaktnosť** alebo kruhovosť je podobnosť tvaru ku kruhu. Vypočítava sa ako $\frac{\text{Obvod}^2}{\text{Obsah}}$ a je vždy vyššia alebo rovná kompaktnosti kruhu, čo je 4π .
- **Excentricita** môže mať niekoľko definícií, jednou z nich je pomer dĺžky najdlhšej úsečky v tvare objektu s dĺžkou najdlhšej úsečky na ňu kolmej. Taktiež je to excentricita elipsy s rovnakým momentom druhého rádu, teda pomer vzdialenosí ohnisiek a dĺžky hlavnej osy.
- **Pozdĺžnosť** je pomer výšky a šírky minimálneho opísaného obdĺžnika.
- **Pravouhlosť** alebo podobnosť s obdĺžnikom je pomer obsahu objektu a obsahu minimálneho opísaného obdĺžnika. Môže nadobúdať hodnoty do 1 a limitne môže klesať k 0.
- **Orientácia** alebo uhol hlavenj osi.
- **Eulerovo číslo** je topologický príznak značiaci rozdiel počtu spojených komponentov a dier v tvare objektu. Topológia popisuje vlastnosti tvaru, ktoré sa

nemenia úpravami okrem delenia a spájania častí tvaru.

- **Počet konkávností** nie je topologickým príznakom, napriek jeho podobnosti s nimi. Získa sa spočítaním spojitéh oblastí v rozdielneho obrazu objektu a jeho konvexného obalu.
- **Konvexnosť** je možné určiť z pomeru obsahu objektu ku obsahu jej konvexného obalu.

2.2 Fotometrické príznaky

Príznaky, ktorých hodnota závisí na jasových úrovniach popisovaného segmentu obrazu sa nazývajú fotometrické príznaky. Môže íst o priemernú ($B_{priemer}$), minimálnu (B_{min}), maximálnu (B_{max}) hodnotu jasu objektu, rozdiel extrémov v objekte ($B_{dif.}$), priemerov jasu objektu (Ω) a pozadia (Φ) ($B_{dif.pozadia}$), či príznaky vyplývajúce z histogramu (H) alebo geometrických momentov.

$$B_{priemer} = \frac{1}{N} \cdot \sum_{(x,y) \in \Omega} f(x,y) \quad B_{max} = \max(f(x,y)) \quad (2.1)$$

$$B_{dif.pozadia} = B_{priemer(\Omega)} - B_{priemer(\Phi)} \quad B_{min} = \min(f(x,y))$$

$$H_{priemer} = \frac{1}{N} \cdot \sum_{q=1}^N q \cdot h(q) \quad H_{kontrast} = \frac{1}{N} \cdot \sum_{q=1}^N (q \cdot h(q) - H_{priemer})^2$$

$$H_{energia} = \frac{1}{N} \cdot \sum_{q=1}^N h(q)^2 \quad H_{entropia} = \frac{1}{N} \cdot \sum_{q=1}^N h(q) \cdot \log_2 h(q)$$

Z geometrických momentov, ktorých hodnoty sú závislé na afinných transformáciách (translácia, rotácia a úprava mierky) je možné úpravou získať príznaky na transformáciu nezávislé. [9] Všeobecne je moment rádu (p,q) segmentu l(x,y) daný:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q l(x,y) dx dy \quad m_{pq} = \sum_X \sum_Y x^p y^q l(x,y) \quad (2.2)$$

(definícia)

(Pre konečný diskrétny obraz)

Centralizované momenty μ_{pq} , vztiahnuté k ťažisku objektu $(x_t, y_t) = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$, získavajú nezávislosť na transláciu:

$$\mu_{pq} = \sum_X \sum_Y (x - x_t)^p \cdot (y - y_t)^q \cdot l(x,y) \quad (2.3)$$

Z centralizovaných momentov je možné určiť momentové invarianty, základných sedem je nazývaných Huovými momentami.

3 Klasifikácia

Ďalším krokom je zaradenie detekovaných objektov do tried, teda ich klasifikácia. Klasifikácia objektov vychádza z ich popisu.

3.1 Klasifikátory

Klasifikátory sú algoritmy schopné automaticky kategorizovať dátu do dvoch a viac tried. Modely určené na klasifikáciu sú väčšinou učené s učiteľom, je potrebné pomerne veľké množstvo anotovaných dát. Výsledkom je schopnosť generalizovať klasifikáciu na nových neznámych dátach.

Existuje niekoľko typov klasifikátorov, napríklad **Lineárne klasifikátory** rozdeľujú príznakový priestor nadrovinami, **Bayesov klasifikátor** vypočítava pravdepodobnosť zaradenia do triedy. Ako klasifikátor môžu slúžiť aj plne prepojené **neurónové siete**.

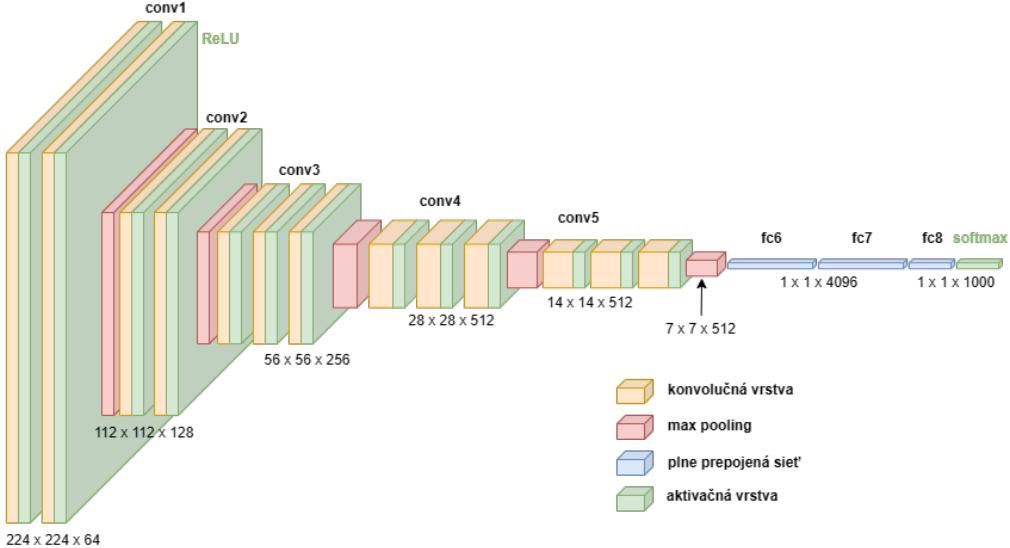
3.2 Konvolučné neurónové siete

Na rozdiel od plne prepojenej neurónovej siete, sú *Konvolučné neurónové siete (Convolutional neural network)* (CNN) prispôsobené na prácu s obrazom ako so vstupným signálom. Pracujú na princípe výpočtu príznakov pomocou konvolučnej časti siete a klasifikáciou plne prepojenou sieťou. 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 (Rectified Linear Unit)* (ReLU) na každý bod. Zavádzia do siete nelinearitu a pomáha zachytávať zložitejšie vzory.
3. **Pooling vrstva** - redukuje priestorové rozšírenie 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 siet. Konvolučná časť slúži na vyhľadávanie príznakov, plne prepojená časť na klasifikáciu pomocou nich.

Rozmery vrstiev závisia na tvare vstupného signálu. V prípade šedotónového obrazu je každý bod obrazu reprezentovaný jednou skalárhou hodnotou, v prípade farebného obrazu záleží počet hodnôt na pixel na farebnom modeli. Klasifikáciou



Obr. 3.1: Príklad architektúry konvolučnej neurónovej siete

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 úlohou klasifikovať viacero tried mávajú formáty:

- Jeden výstup, ktorého hodnota určuje triedu, do ktorej sú vstupné dátá sieťou priradené.
- Vektor výstupov s toľkými hodnotami, na koľko tried je naučená sieť klasifikovať. Hodnoty výstupov môžu reprezentovať:
 - Ohodnotenie triedy (score-based classification). V tomto prípade čísla na výstupe nemajú samostatne význam, ale ako predpoved 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

Existuje niekoľko metód schopných detektie aj klasifikácie pomocou jediného modelu. Vďaka tomu je možné vynechať krok detektie a získať priamo klasifikované orámovania, čím sa systém detektie zjednoduší. Je však nutné porovnať výkon a kvalitu detektie a klasifikácie v jednom kroku s prístupom so samostatnou detekciou a klasifikáciou.

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 obrazu. 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 nezávislý na mierke nazývaný *bounding box regressor*. Jeho výstup je štvorrozmerný, skladá sa z polohy a rozmerov ohraničujúceho obdĺžnika.

R-CNN sú efektívne, no náročné na výpočtový 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 algoritmus detektie, ktorý na rozdiel od R-CNN rozdeľuje obraz do mriežky a postupne aplikuje klasifikátor, eliminujúc krok návrhu regiónov záujmu. Zameriava sa teda na rýchlosť a jednoduchosť. Postupuje nasledovne:

1. Vstupný 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. Oblasti sú popísané štyrmi číslami, dve súradnice stredného bodu oblasti a dva rozmery ohraničujúceho obdĺžnika.
3. Je použitých viacero klasifikátorov. Počas trénovalia je jednému z prediktorov udelená zodpovednosť za predikovanie daného objektu podľa toho, ktorý z nich dosahuje najväčší *Pomer prieniku voči zjednoteniu* (*Intersection over Union*) (IoU). Výstupom klasifikácie v YOLO je vektor predikciej pravdepodobností, že objekt obsiahnutý v danom ohraničení patrí do každej triedy. Celkový výstup YOLOv8 je tenzor obsahujúci zoznam všetkých klasifikovaných oblastí.
4. Každá bunka predikuje rozdelenie pravdepodobnosti všetkých tried.

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

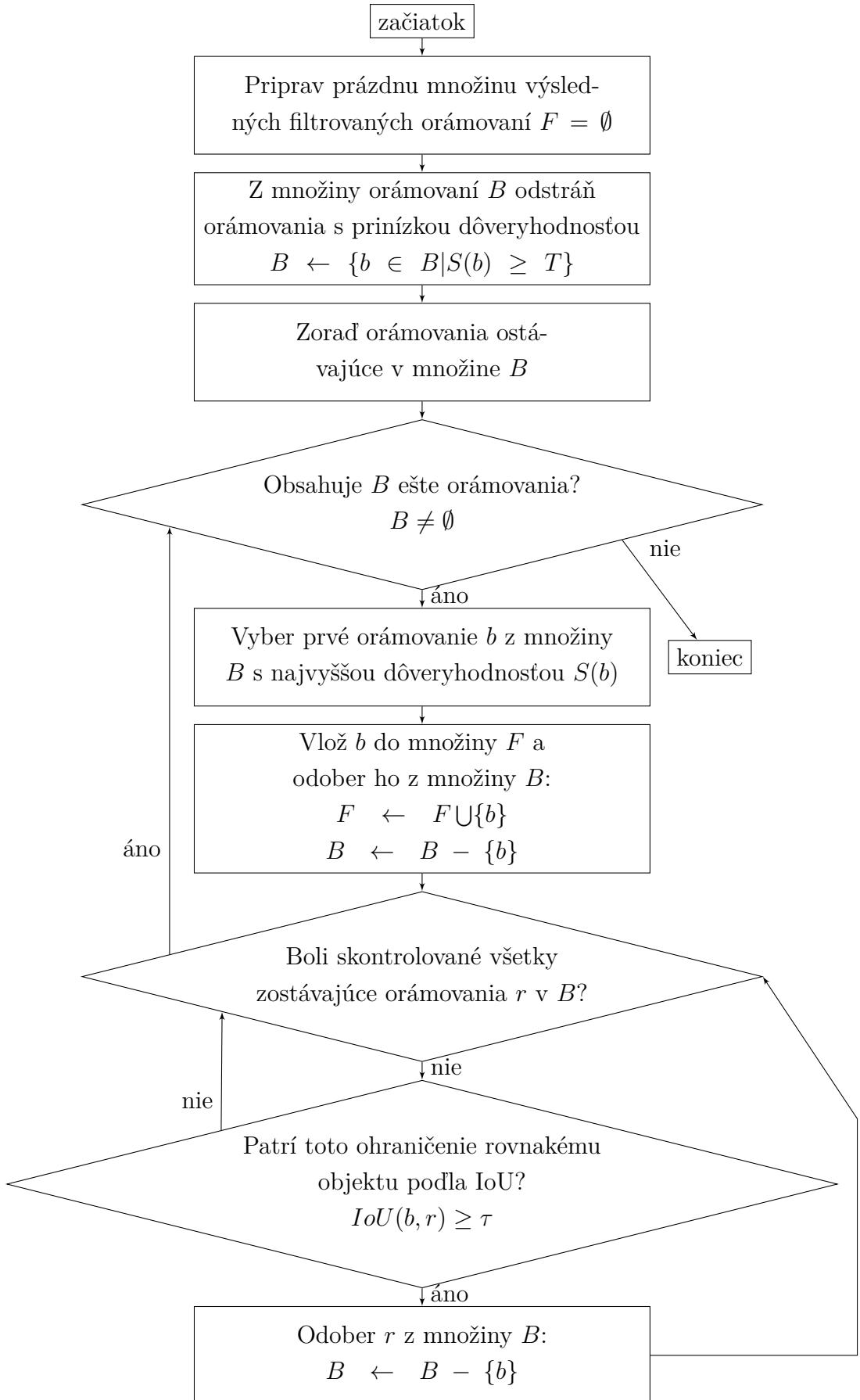
4.2.1 Non-Maximum Suppression

Výstup zo siete YOLOv8 je vo formáte zoznamu 8400 potenciálnych orámovaní, ktoré sú tvorené súradnicami stredového bodu, rozmermi orámovania a ohodnením dôveryhodnosti, že orámovevanie obsahuje objekt pre každú triedu. Typickým problémom detekčných modelov je že ich výstup obsahuje niekoľko prekrývajúcich sa orámovaní patriacich rovnakému objektu. Algoritmus NMS pre odstránenie redundantných orámovaní postupuje podobne ako pri jeho aplikovaní na obraz hrán. Výsledkom je nájdenie najlepšieho orámovania pre každý objekt v obraze. Pred jeho použitím je nutné určiť dva parametre. Prvým je prah IoU medzi orámovaniami τ ,

druhým prah dôveryhodnosti T . NMS začne prácu s množinou orámovaní B ohodených úrovňou dôveryhodnosti S .



Obr. 4.1: Vstup a výsledok NMS



Obr. 4.2: Algoritmus Non-Maximum Suppression pre orámovanie objektov

5 Ukladanie naučených modelov

Neurónové siete využívané v tejto práci je nutné po ich naučení uložiť tak, aby ich dokázala aplikácia systému v zariadení načítať a použiť. K tomu je dostupných niekoľko rôznych formátov.

5.1 ONNX

ONNX je otvorený formát vybudovaný na reprezentáciu rôznych modelov strojového učenia. Definuje stavebné bloky modelov strojového a hlbokého učenia so spoločným formátom súborov dovoľiacim vývojárom používať uložený model s rôznymi nástrojmi a programami. Hlavnou výhodou je interoperabilita umožňujúca vytváranie modelov, ich testovanie a používanie v nezávislých aplikáciach. Ak je pre aplikáciu známy formát vstupu a výstupu, sú modely uložené v ONNX rovnocenné z hľadiska ich ovládania, vďaka čomu je implementácia využitia modelov rovnakého druhu zjednodušená.

5.2 Keras

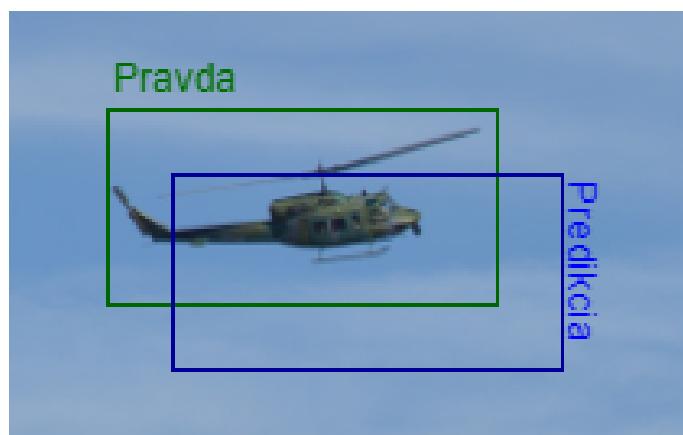
Knižnica *Keras* patriaca pod knižnicu *Tensorflow* umožňujúca prácu so strojovým učením je jednou s často používaných knižníc pre *Python* s týmto zameraním. Je v nej implementovaný vlastný formát súborov na ukladanie modelov, z ktorých je možné jednoducho načítať model postavený z komponentov z knižnice *Keras*. Vďaka existencii špecializovaných verzii knižníc *Tensorflow* na využitie s procesorom aj GPU je možné nainštalovaním správnej verzie knižnice rozhodnúť o využití výpočtového výkonu počítača.

6 Vyhodnotenie kvality detekcie

K porovnaniu a vyhodnoteniu kvality výsledkov rôznych prístupov je nutné zhodnotiť anotovaný dataset so snímkami zachytenými za reálnych podmienok kamerou priamo na zariadení. Použitím porovnávaných prístupov na snímky v tomto datasete a ich následným porovnaním s anotáciou je možné vypočítať metriky popisujúce ich kvalitu detektie a klasifikácie.

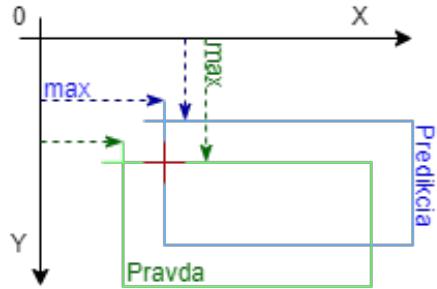
6.1 Kvalita orámovania

Často používanou metrikou presnosti orámovania je takzvané IoU. To je získané porovnaním orámovania určeného pri anotácii a orámovania predikovaného detektorm.

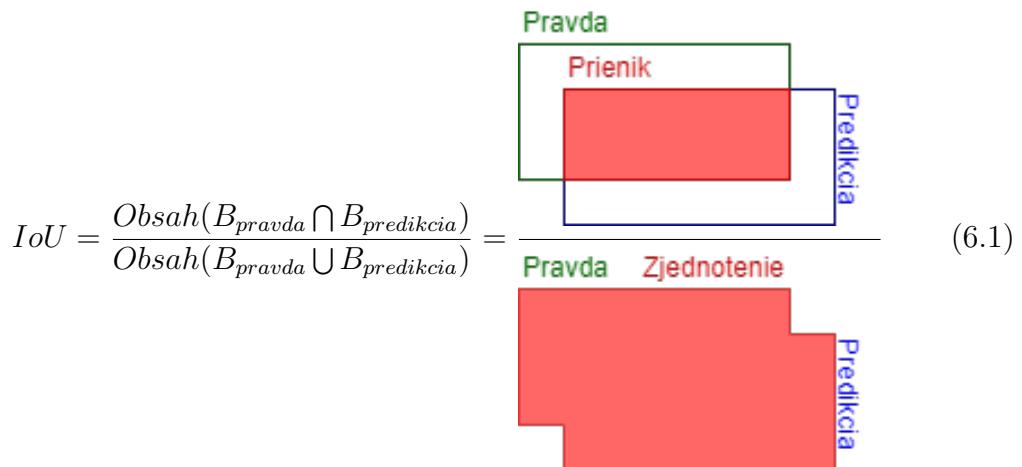


Obr. 6.1: Orámovanie na anotovanom snímku

Pred výpočtom je nutné určiť obsah prieniku a následne zjednotenia obdĺžnikov anotovaného a predikovaného orámovania. Zvolíme súradnicový systém s počiatkom v ľavom hornom rohu so súradnicami stúpajúcimi smerom dolu a doprava. Prienik dvoch obdĺžnikov je možné určiť nájdením ľavého horného rohu, zvolením najvyšších súradníc ľavých horných rohov obidvoch obdĺžnikov. Pravý dolný roh je naopak nájdený ako najnižšie súradnice pravých dolných rohov. Obsah prieniku je potom súčtom obsahov obdĺžnikov po odčítaní obsahu ich prieniku.



Obr. 6.2: Nájdenie prvého bodu prieniku orámovaní



Hodnota IoU sa pohybuje medzi 0 a 1, pričom vyššia hodnota značí vyššiu presnosť. Hodnota 1 znamená dokonalé prekrytie.

6.2 Kvalita klasifikácie

6.2.1 Matica zámien

Matica zámien (confusion matrix) je prehľadné zobrazenie výsledkov klasifikácie určitej množiny objektov. Vykresluje sa ako tabuľka rozdeľujúca počet vzoriek z každej skutočnej triedy podľa ich predikovanej triedy. Ak sú skutočné a predikované triedy v tabuľke v rovnakom poradí, na diagonále sa nachádzajú počty správne zaradených, mimo diagonály zas nesprávne zaradených inštancií.

6.2.2 Metriky vychádzajúce z matice zámien

Elementy matice zámien je možné roztriediť podľa správnosti ich klasifikácie do konkrétnej triedy. Hodnoty metrík klasifikácie sú vypočítavané vzhľadom na konkrétnu triedu alebo pre celkovú klasifikáciu.

Riadky matice 6.3 reprezentujú skutočnosť, stĺpce predikciu. P (pozitívne) zaradenie reprezentuje prvky zaradené do triedy, pre ktorú chceme počítať metriky klasifikácie. N (negatívne) značí ostatné triedy.

	N	P	N
N	TN	FP	TN
P	FN	TP	FN
N	TN	FP	TN

Obr. 6.3: Rozdelenie prvkov matice

- **TP** (true positive) sú inštancie zaradené správne do triedy, ktorej klasifikáciu chceme ohodnotiť.
- **TN** (true negative) sú zaradené správne do iných tried.
- **FP** (false positive) sú inštancie nesprávne označené za sledovanú triedu.
- **FN** (false negative) sú naopak označené ako sledovaná trieda, no správne patria do jednej z ostatných tried.

- **Presnosť (precision)** vyjadruje ako často, keď model predpovedá určitú triedu, je jeho predpoveď správna.

$$Precision = \frac{TP}{TP + FP} \quad (6.2)$$

- **Úplnosť (recall)** značí ako často model správne vyhodnotí objekt ako patriaci do danej triedy v pomere ku skutočnému počtu objektov z tej triedy. Nízke skóre úplnosti značí že model primálo často predpovedá danú triedu.

$$Recall = \frac{TP}{TP + FN} \quad (6.3)$$

- **F1 skóre** je kombináciou presnosti a úplnosti. Využíva sa v prípade hľadania rovnováhy medzi presnosťou a úplnosťou.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (6.4)$$

- **Správnosť (accuracy)** je pomer spočtu právne klasifikovaných inštancií ku počtu všetkých inštancií.

$$Accuracy = \frac{TP + TN}{P + N} \quad (6.5)$$

- **Citlivosť (sensitivity)** znamená, ako často model správne vyhodnocuje objekty danej triedy ako patriace do tej triedy.

$$Sensitivity = \frac{TP}{P} \quad (6.6)$$

- **Špecificka (specificity)** na druhú stranu vyjadruje, ako často sú správne vyhodnotené prvky nepatriace do danej triedy.

$$Specificity = \frac{TN}{N} \quad (6.7)$$

6.3 Výkon detekcie

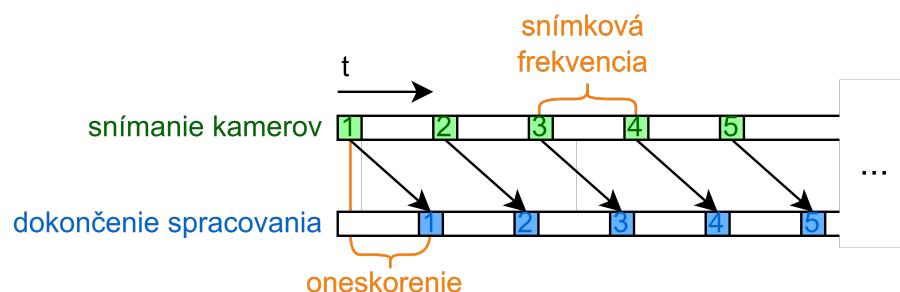
6.3.1 Snímková frekvencia

Dôležitým parametrom systému detekcie je frekvencia, s akou je schopný získavať a spracovať snímky. V prípade, že je snímková frekvencia príliš nízka, je možné, že príliš rýchlo sa pohybujúce objekty nestrávia v zábere kamery dostatočný čas na to, aby boli zachytené na aspoň jednom snímku. Jedny z hlavných faktorov obmedzujúcich snímkovú frekvenciu sú:

- Maximálna snímkovacia frekvencia kamery pri danom rozlíšení.
- Výpočtový výkon zariadenia, na ktorom je systém spustený.
- Zložitosť metód použitých v systéme.

6.3.2 Oneskorenie

V rôznych oblastiach využitia systému detekcie môže byť požadované spracovanie vstupného obrazu v určitom časovom úseku. Ak je potrebné dosiahnuť čo najnižšie oneskorenie, musia byť zvolené dostačne jednoduché metódy s prihliadnutím na výkon zariadenia.



Obr. 6.4: Snímková frekvencia a oneskorenie

7 Návrh 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 podmienok je teda jeho prenosnosť, možnosť napájania z akumulátora a nezávislosť na iných zariadeniach pri jeho nastavení a používaní.

7.1 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 softvéru aj hardvéru. Z týchto dôvodov bolo *Raspberry Pi* zvolené na použitie v zariadení.

7.2 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 °
- **Najväčšie rozlíšenie** 4608 x 2592 px

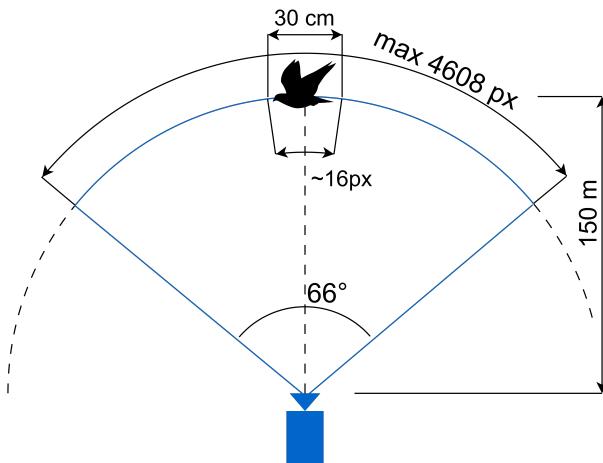
Za bežných podmienok sa vtáky pohybujú vo výške približne 150 metrov nad zemou [10]. Ak predpovedáme, že vták má za letu približne 30 cm na šírku, zobrazí sa vo výslednom snímku na najviac približne 16 pixeloch v horizontálnej osi.

$$W_u = 2 \times \arctan\left(\frac{d}{2D}\right) = 2 \times \arctan\left(\frac{0,3}{2 \times 150}\right) \approx 0.002[\text{rad}] \quad (7.1)$$

kde W_u je uhlová šírka objektu, d je šírka objektu a D je jeho vzdialenosť od kamery.

$$W_{px} = R_{px} \times \frac{W_u}{R_u} \quad (7.2)$$

kde W_{px} je šírka objektu v obraze v pixeloch, R_{px} je horizontálne rozlíšenie v pixeloch a R_u je uhol zorného poľa kamery.



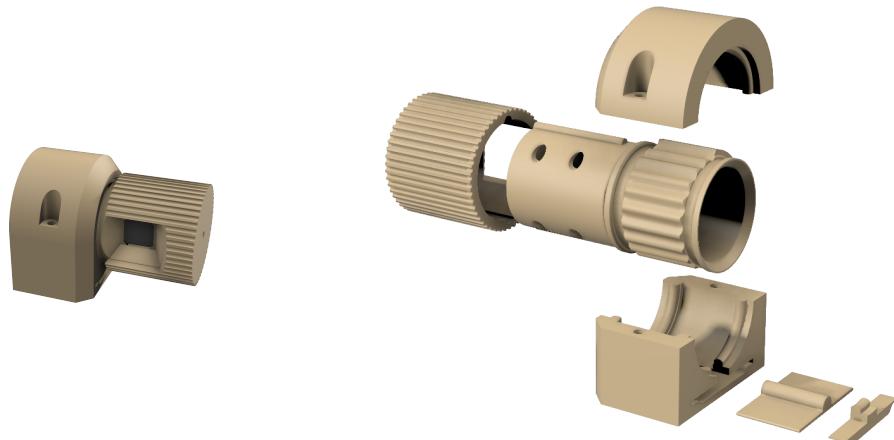
Obr. 7.1: Rozlíšenie kamery

7.3 Ovládacie prvky

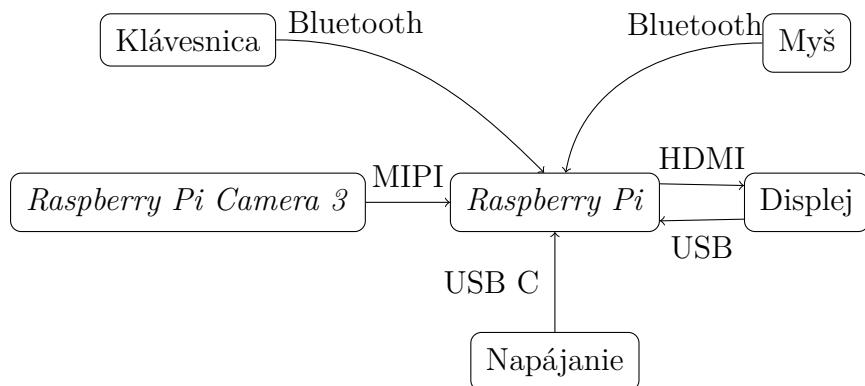
Ako najlepší ovládaci a zobrazovací prvok bol z hľadiska prenosnosti a kompaktnosti zvolený dotykový displej, konkrétnie 7 palcový IPS displej od firmy Waveshare. Táto veľkosť je postačujúca na zobrazovanie výsledkov detekcie aj ovládanie zariadenia. Rozhranie HDMI značne zjednodušuje prepojenie a inštalačiu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

7.4 Uchytenie kamery

Praktickou výhodou prenosného zariadenia je možnosť jeho umiestnenia na statív. Kedže cieľom je kamerou mierit smerom nahor na oblohu, bolo by užívateľské rozhranie pri nesprávne navrhnutej 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. 7.2: Návrh nastaviteľného uchytenia kamery



Obr. 7.3: Blokov\u00e1 sch\u00e9ma zariadenia

7.5 Kryt zariadenia

Kryt zariadenia sa skladá z troch hlavných častí. Jednotlivé komponenty zariadenia sú upevnené ku spoločnej stredovej doske. Tá má zaistiť pevné spojenie komponentov. Z prednej strany dosky je pripojený displej. Ten je prepojený ku *Raspberry Pi*

na druhej strane stredovej dosky, na ktorej je na jednej strane vytvorený výrez na HDMI a USB káble. Na druhej strane je zároveň umiestnená batéria napájajúca *Raspberry Pi*. Displej je napájaný cez USB pripojené k *Raspberry Pi*, pomocou ktorého zároveň komunikuje jeho dotyková plocha.



Obr. 7.4: Pohľad na zariadenie spredu



Obr. 7.5: Pohľad na zariadenie zo zadu, s odnímateľnými krytmi

Hlavná doska je vložená medzi prednú a zadnú časť vonkajšieho krytu. Vonkajší kryt obklopuje celé zariadenie, okrem vnútorné nevyužitých USB výstupov *Raspberry Pi* a batérie. Tie sú prístupné za odnímateľnými krytmi.

8 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.

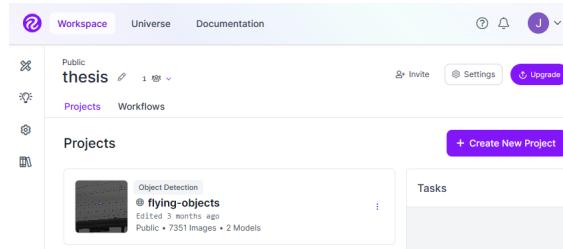
8.1 Získavanie dát

Kedže táto práca nadväzuje na prácu [1], tvorí dataset v nej vytvorený základ pre dataset 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, augmentáciu datasetu a generovanie anotačných súborov rôznych formátov.

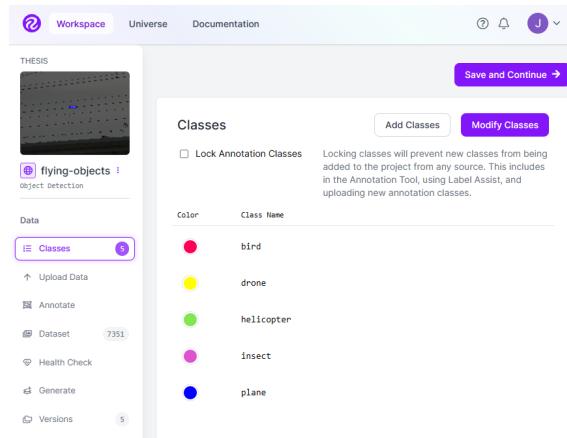
Práca s programom *roboflow* postupuje v krokoch:

1. Vytvorenie projektu. Projekt programu *roboflow* zahŕňa snímky, ich anotáciu a vygenerovaný výstup spolu s augmentáciou.



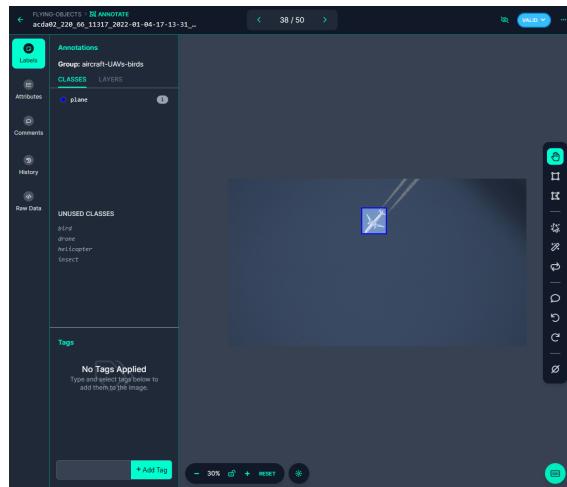
Obr. 8.1: Založenie roboflow projektu

2. Definícia tried anotovaných objektov. V prípade vloženia cudzích datasetov je možné spojiť triedy s rôznym názvom, ktoré majú reprezentovať rovnakú triedu objektu. Triedam sú priradené farby pre ľahšie rozlíšenie objektov pri ručnej anotácii.



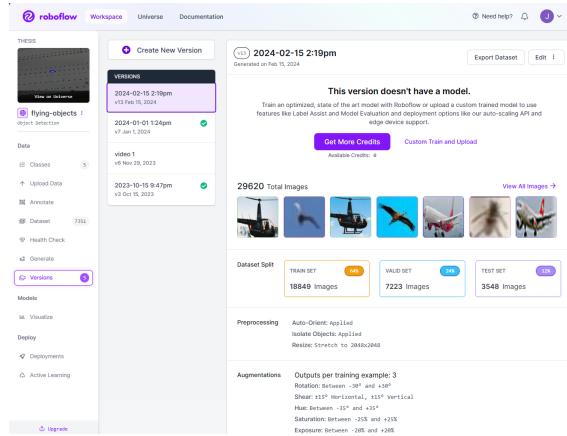
Obr. 8.2: Definícia tried roboflow projektu

3. Nahratie vlastných fotografií alebo naklonovanie existujúceho datasetu. Tako importovať je možné snímky spolu aj s anotáciou v prípade, že je už vytvorená v kompatibilnom formáte.
4. Ručná anotácia. Pre použitie v tejto práci bola vytvorená anotácia obdĺžnikovoým orámovaním objektov so zaradením do tried.



Obr. 8.3: Anotácia snímku v programe roboflow

5. Kontrola, nastavenie a vygenerovanie novej inštancie datasetu. Pred stiahnutím archív s anotovaným datasetom je potrebné nastaviť parametre ako je požadované rozlíšenie a spôsob zarovnania tvaru snímok (čierne pozadie, výrez, roztahnutie, zrkadlenie). Taktiež je možné aplikovať nastavené randomizované augmentácie na náhodne vybrané snímky z trénovacieho datasetu. Po nastavení počtu generovaných snímok je možné vygenerovať a stiahnuť dataset.



Obr. 8.4: Generované verzie datasetu v roboflow

Po pridaní všetkých častí datasetu a kontrole správnosti anotácie boli počty inštancií jednotlivých tried v trénovacom datasete:

- **vták:** 1198
- **dron:** 1898
- **helikoptéra:** 1281
- **hmyz:** 934
- **lietadlo:** 972



Obr. 8.5: Ukážka niekoľkých snímok z datasetu

8.2 Úprava, rozdelenie a rozšírenie datasetu

Niektoré triedy sú reprezentované výrazne menším množstvom jednotlivých inštancií, čo je spôsobené nižšou dostupnosťou dát. Väčšina fotografií vtákov zo vzdialenosťi, pri ktorej by mali byť detekované, pochopiteľne obsahuje niekoľko 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 dátá, na ktorých je model učený, rôzne od validačných a testovacích. Anotované obrazové dátá boli rozdelené do skupín s počtom:

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

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

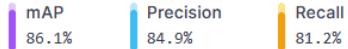
- Rotácia -30° až $+30^\circ$
- Zkosenie $\pm 15^\circ$ vertikálne aj horizontálne
- Posun odtieňu -35° až $+35^\circ$
- Zmena saturácie -25% až $+25\%$
- Zmena expozície -20% až $+20\%$
- Rozmazanie do 0,8 px

8.3 Testovanie použiteľnosti datasetu

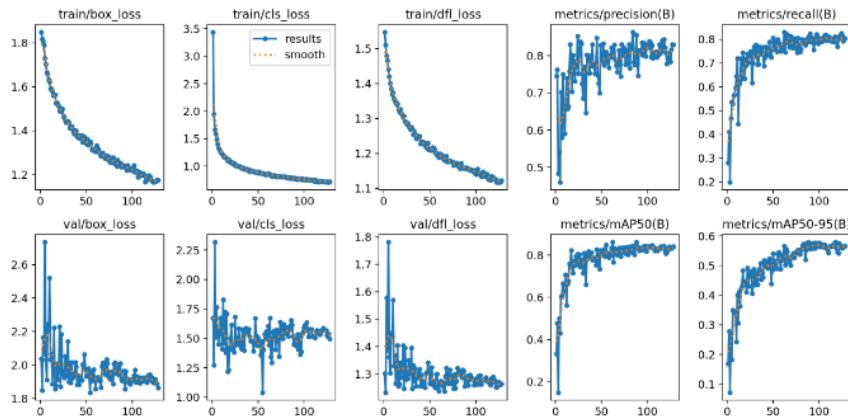
Použiteľnosť modelu bola testovaná naučením modelu typu *Roboflow 3.0 Object Detection* priamo v aplikácii *roboflow*, ktorý dosahoval úroveň presnosti 86.1 % pri zvolenom variante modelu *fast*. Pri dlhšom učení modelu určeného na reálne použitie sa dá predpokladať, že dosiahnutá presnosť bude vyššia.

Obrázok 8.6 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).**
- **Senzitivita (Recall).**

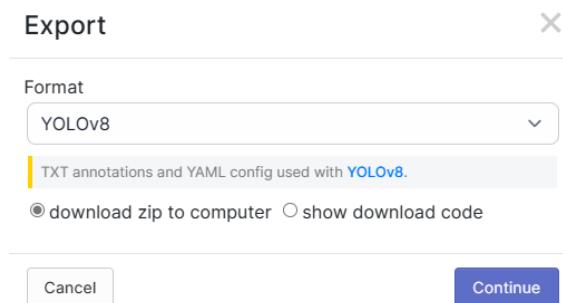


Obr. 8.6: Metriky presnosti naučeného modelu v programe roboflow



Obr. 8.7: 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. 8.8: Exportovanie vytvoreného datasetu

9 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 rozšíriteľnosť,
- spätná väzba a grafické zobrazenie výsledkov užívateľovi,
- možnosť používať, testovať a porovnávať rôzne metódy detekcie, klasifikácie a trasovania.

9.1 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 Tk/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 spoloahlivo 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. Sofistikanejšie úlohy, ktoré knižnica rieši, zahrňujú detekciu, klasifikáciu, sledovanie pohybu. Knižnica *OpenCV* je kompatibilná s viacerými programovacími jazykmi, vrátane jazyku *python*.

9.2 Štruktúra aplikácie

Aplikácia systému je zložená z automaticky načítavaných modulov. Každý z modulov predstavuje časť retazca spracovania obrazu a detekcie objektov v ňom. Program patriaci každému modulu je vykonávaný v samostatnom vlákne, vďaka čomu nemusia moduly čakať na vykonanie nasledujúceho modulu a namiesto toho môžu spracovávať ďalší snímok v poradí.

Jedným z modulov, ktoré sú vždy aktívne, je modul reprezentujúci zoznam aktívnych modulov. Ten je zodpovedný za načítavanie a spúšťanie modulov, distribúciu dát medzi modulmi a ukladanie a načítavanie konfigurácie modulov.

9.2.1 Štruktúra modulu aplikácie

Každý z automaticky načítavaných modulov dedí od abstraktnej triedy `modules.ModuleBase`. Tá predpisuje spoločnú časť správania modulov, menovite:

- **Prvky užívateľského rozhrania týkajúce sa modulu:**
 - záznam v zozname typov modulov,
 - užívateľské rozhranie s nastaveniami modulu,
 - záznam v zozname aktívnych inštancií modulov.
- **Zoznam konfiguračných metód.** Tie sú využívané pri získavaní konfigurácie z modulu a jej vloženie do modulu pri jeho spustení. Priamo v kóde modulu sú konfiguračné metódy definované ako bežné metódy, ktoré prijímajú ako voliteľný vstup novú hodnotu v prípade, že je požadovaná zmena a vracajú aktuálnu hodnotu určitého nastavenia modulu. Automatické spúšťanie ukladania konfigurácie pri zmene nastavenia a opäťovné načítavanie konfigurácie je dosiahnuté implementáciou dekorátora, ktorým sú konfiguračné metódy označené. Pri definícii triedy sú automaticky vložené jej označené konfiguračné metódy do zoznamu spoločného pre všetky moduly, kde sú priradené danému modulu. Pri volaní konfiguračnej metódy sa spúšťa wrapper metóda definovaná v dekorátore, úlohou ktorej je zmena uloženej hodnoty v konfiguračnom súbore.
- **Unikátnu identifikáciu modulu** pridelenú automaticky pri vytvorení novej inštancie modulu.
- **Určenie typu vstupných aj výstupných dát.** To umožňuje niektorým modulom vedieť, aké dátá môžu očakávať a užívateľskému rozhraniu aké dátá je možné vizualizovať.
- **Vstupnú frontu**, do ktorej sú vkladané objekty obsahujúce kolekciu dát priradených k jednotlivým typom dát, ktoré modul definuje ako svoje vstupy a výstupy. Objekt s dátami je pri jeho vzniku vložený ako referencia do fronty každého z modulov, ktoré potom čakajú na dostupnosť ich požadovaných dát. Po získaní vstupu je vykonaná činnosť modulu a do objektu z fronty sú pridané výstupné dátá. Kedže sú objekty do vstupných front predávané referenciou, dátá sú po ich pridaní dostupné aj v ostatných moduloch. Aby bolo čakanie efektívne, je pri pridaní dát nastavený *threading.Event*, ktorý zobudí čakajúce moduly. Modul má dostupné funkcie na nahliadnutie do fronty, získanie nasledujúceho objektu z fronty a vloženie nového objektu do front všetkých

aktívnych modulov, ktoré očakávajú vstup.

- **Konzolový výstup.** Každému modulu je predaný objekt reprezentujúci konzolový výstup. Výpis do konzoly je vždy uvedený spolu s názvom a ID modulu, a časovou značkou. Konzolový výstup je vykonávaný v samostatnom vlákne, moduly teda nečakajú na dokončenie výpisu a súčasný výpis z viacerých modulov nespôsobí nesprávny formát výstupu. Výstup podporuje konzolové farby pomocou escape sekvencií a farebne rozlišuje informatívny, varovný, chybový výstup, a výstup informujúci o nastavení modulov. Názov modulu a časová značka je tiež farebne rozlíšená.

```
module list : 46abbe6c - 16:08:17 - creating module
module list : 46abbe6c - 16:08:17 - finished creating
module list : 46abbe6c - 16:08:17 - looking for modules
in background_model.py
module list : 46abbe6c - 16:08:17 - importing module
object BackgroundModel
```

Obr. 9.1: Príklad výstupu pri spustení aplikácie

- **Inicializačné vlákno** spustené pri inicializácii modulov, ktoré je zodpovedné za prvotné nastavenie modulu.
- **Aktívne vlákno** je spustené pri aktivácii modulov, ktoré v slučke získava, spracováva vstupné dátá a ukladá výstupné dátá. Toto vlákno počas života objektu neskončí, ak nedôjde k nezachytenej výnimke.
- **Timeout dát.** Každému objektu vo vstupnej fronte je priradený čas jeho vzniku. Keď sa ich modul pokúša z fronty čítať, sú preskočené všetky objekty staršie ako určený čas.

Aplikácia automaticky načíta všetky moduly definované v súboroch uložených v priečinku *autoload_modules* použitím python knižnice `importlib`. Výberom modulu zo zoznamu dostupných modulov je vytvorená nová inštancia modulu. Na nej je možné zmeniť konfiguráciu pred vložením inštancie do zoznamu aktívnych modulov.

9.2.2 Implementované moduly

Do aplikácie boli implementované nasledujúce moduly:

- **Zdroj obrazu:** Tento typ modulu získava nové obrazové dátá. Následne môže vytvárať šedotónový obraz, pričom je možné určiť váhy farebných zložiek. Nakoniec je možné vykonávať prahovanie, a to buď s nastaveným alebo s automaticky určeným prahom. Existujú dva varianty:

- **Video:** V nastaveniach modulu videa ako zdroja obrazu je možné okrem spoločných nastavení pre všetky zdroje určiť aj cestu k súboru videa. Po pridaní do zoznamu aktívnych modulov a spustení, tento modul začne pridať obrazové dáta do fronty so snímkovacou frekvenciou, akú má zdrojové video.
 - **Kamera:** Ak je v prostredí, v ktorom aplikácia beží dostupná python knižnica PyCamera2, ktorá slúži na ovládanie kamery a získavanie snímok z nej, je možné pridať modul kamery. Ten má možnosť nastaviť rozlíšenie kamery.
- **Dynamický model pozadia:** Generuje postupne sa aktualizujúci model pozadia, s ktorým potom vypočíta absolútnu hodnotu rozdielu s aktuálnou snímkou. Ten je potom prahovaný nastavenou hodnotou, čím vzniká binárny obraz, v ktorom je možné detegovať pohybujúce sa objekty. Nastaviť je v tomto module možný prah, ktorý ked prekročí úroveň pohybu v obraze, prestane sa aktualizovať model pozadia. Nastaviteľné prahy sú dva, prah dvoch po sebe nasledujúcich snímkov a prah medzi snímkou a modelom pozadia. Ich úroveň je určená relatívne k rozlíšeniu obrazu a normalizovaná na maximálnu úroveň jasu jedného pixelu, čím je získaná nezávislosť na rozlíšení a číselnej reprezentácii obrazu. Ďalšie nastavenia zahŕňajú časovú konštantu zmeny pozadia, spolu s maximálnou zmenou jasu každého pixelu za jednu iteráciu.
- **Generátor orámovaní z binárneho obrazu:** Vyhľadáva súvislé oblasti v binárnom obraze a vytvára obdlžnikové orámovania. Pred vyhľadávaním najprv vykoná morfológické zatvorenie s cieľom spojiť blízke oblasti, ktoré pravdepodobne patria jednému objektu. Tvar jadra zatvorenia je možné nastaviť. Oblasti s menším počtom pixelov ako nastavený prah sú ignorované.
- **Tensorflow klasifikácia:** Načíta uložený naučený klasifikačný model vo formáte keras. Vďaka tomu je možné použiť na výpočty GPU na zariadeniach, kde je dostupná, nainštalovaním GPU verzie knižnice tensorflow. Tomuto modulu je potrebné nastaviť cestu k súboru s uloženým modelom. Načítaný model je potom aplikovaný na vystrihnuté oblasti podľa orámovaní z pôvodného farebného obrazu. Očakávaný je výstup typu one-hot, teda každý výstup určuje predpokladanú pravdepodobnosť zaradenia objektu v orámovaní do triedy asociovanej s daným výstupom. Výsledná klasifikácia je zvolená ako trieda s najvyššou pravdepodobnosťou, ak presahuje určený prah. Modely pre tento modul boli vytvorené v prostredí matlab deep network designer, naučené pomocou transfer learning a exportované do formátu pre tensorflow keras. Boli použité predučené konvolučné siete Resnet50, SqueezeNet a Googlenet.
- **YOLO detektor:** Reprezentuje detekčnú sieť YOLOv8 schopnú detekcie objektov aj ich klasifikácie v jedinom kroku. Pri použití tohto modulu teda nie je

nutné pridávať iné moduly detekcie. Výstup modelov YOLO je taktiež one-hot, výsledná klasifikácia funguje teda rovnako ako vo vyššie popísanom module tensorflow klasifikátorov. Spolu s klasifikáciou vracia tento modul aj samotné orámovanie detegovaných objektov.

- **Záznam videa:** Získané snímky je možné týmto modulom ukladať do súboru videa. Spolu s videom sú ukladané orámovania detegovaných potenciálnych objektov. Ak je dostupný modul generujúci orámovania, sú ukladané len snímky s potenciálnym výskytom objektu.

9.3 Automatické spúšťanie aplikácie

Zjednodušením používania zariadenia je zaistenie automatického spustenia aplikácie systému detekcie pri zapnutí zariadenia. Toho bolo dosiahnuté nastavením aplikácie ako servisu operačného systému *Raspberry Pi OS*, ktorý je nainštalovaný na *Raspberry Pi* na zariadení.

Servisy sú v systéme *Raspberry Pi OS* spravované servisovým manažérom *Systemd*. Definované sú v konfiguračných súboroch, takzvaných *systemd unit files*, uložených v priečinku */etc/systemd/system/*. Konfigurácia je v týchto súboroch rozdelená do sekcií. Najdôležitejšou sekciou je sekcia [Service], obsahujúca informácie o spustení aplikácie. Tu je nutné definovať pracovný priečinok aplikácie, príkaz spustenia a spôsob reštartovania v prípade nečakaného ukončenia aplikácie. Aby mala aplikácia prístup ku potrebným knižniciam, je nutné použiť virtuálne prostredie *python* s nainštalovanými knižnicami. V konfigurácii spustenia aplikácie je nutné použiť *python* interpretér nainštalovaný v tomto virtuálnom prostredí.

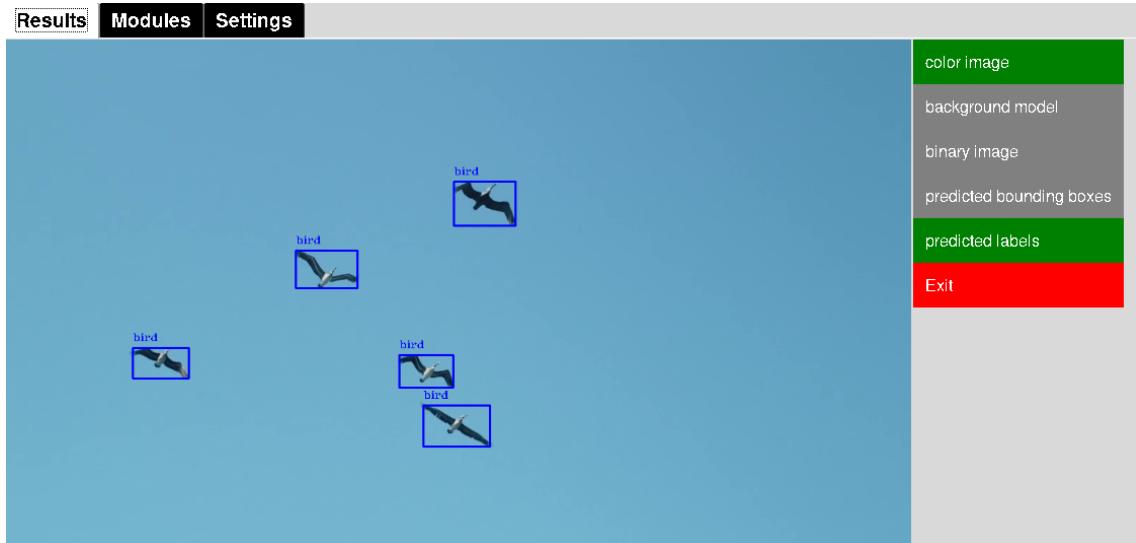
```
[Service]
Type=simple
WorkingDirectory=<cesta ku pracovnému priečinku>
ExecStart=<python interpreter> <skript aplikácie>
Restart=always
User=<užívateľ>
```

Obr. 9.2: Sekcia [Service] konfigurácie automatického spustenia aplikácie

9.4 Návrh grafického rozhrania

Grafické užívateľské rozhranie (Graphical User Interface) (GUI) umožňuje intuitívne ovládanie aplikácie a zobrazovanie výsledkov detekcie a klasifikácie. Ovládanie apli-

kácie je rozdelené do záložiek. V ľavej časti obrazovky je zobrazený aktuálny snímok s výstupom detekcie a klasifikácie. Zobrazujú sa v ňom orámovania detegovaných objektov a ich priradenie do tried. Prvá záložka (*Results*) zobrazuje aktuálny snímok spolu s vybranou vizualizáciou.



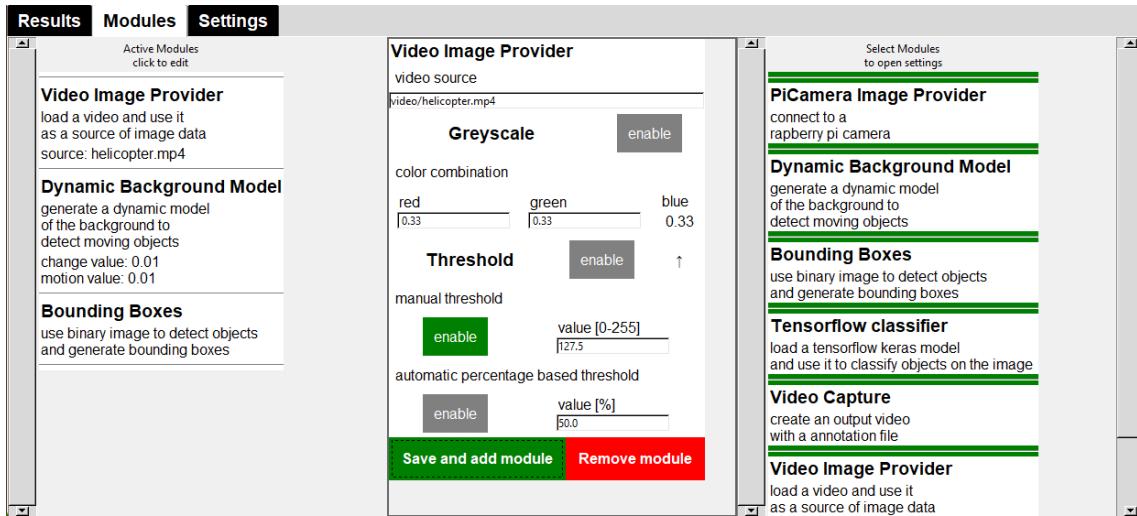
Obr. 9.3: Záložka na zobrazenie výstupov detekcie

Zobraziteľné dátá zahrňujú:

- **Pôvodný farebný obraz** získaný z videa alebo kamery.
- **Šedotónový obraz** vytvorený lineárnom kombináciou farebných zložiek pôvodného obrazu. Je využívaný pri detekcii prahovaním.
- **Aktuálny model pozadia** slúžiaci na detekciu objektov pohybom.
- **Binárny obraz**, v ktorom sú označené oblasti potenciálneho výskytu objektu.
- **Orámovanie oblastí** s potenciálnym výskytom objektu.
- **Klasifikácia oblastí** s farebným zvýraznením orámovaní a zobrazením názvu tried.

Zoznam na pravej strane obrazovky obsahuje prepínače rozhodujúce o tom, ktoré z možných zobraziteľných dát sú vľavo vykreslované. V zozname sú zobrazené len prepínače pre tie typy dát, ktoré sú produkované aktuálnymi aktívnymi modulmi.

O tom, ktoré moduly sú aktuálne aktívne, rozhoduje užívateľ v druhej záložke (*Modules*).



Obr. 9.4: Záložka na nastavenie modulov

Rozhranie tejto záložky je rozdelené do troch stĺpcov posúvateľných vertikálne, keďže množstvo obsahu každého nie je obmedzené. Ovládacie prvky v každom z týchto stĺpcov sú definované v súboroch spolu s automaticky načítavanými modulmi.

Stĺpec na pravej strane obsahuje zoznam dostupných typov modulov. Po kliknutí na typ modulu je buď vytvorená nová inštancia modulu zvoleného typu, alebo aktívna inštancia, ak už pre tento typ jedna existuje. Takto získaná inštancia je potom predaná strednému stĺpcu.

Stredný stĺpec obsahuje konfiguračné ovládacie prvky modulu. Zobrazuje sa vždy nastavenie zvolenej inštancie, pričom zmena nastavení spustí uloženie novej konfigurácie do yaml súboru. Kliknutím na tlačidlo *Save and add module* je modul s nastavenou konfiguráciou vložený do zoznamu aktívnych modulov, ak ide o novovytvorenú inštanciu.

V ľavom stĺpci sa nachádza zobrazenie aktívnych inštancii modulov. Kliknutím na záznam reprezentujúci aktívnu inštanciu modulu otvorí nastavenia tejto inštancie v strednom stĺpci, rovnako ako pri kliknutí na typ už aktívneho modulu v pravom stĺpci. Záznamy v ľavom stĺpci môžu zobrazovať dodatočné informácie, ako napríklad modul dynamického modelu pozadia, ktorý zobrazuje úroveň detektovaného pohybu, čo uľahčuje nastavenie prahov detekcie pohybu.

10 Kontrola a porovnanie prístupov

Aby bolo možné porovnať rôzne prístupy k detekcii a klasifikácii, je nutné zozbierať reálne zábery z použitej kamery. Tie sú následne po ručnom anotovaní predané jednotlivým typom detektorov. Detekcie je potom možné porovnať s anotáciou a získať tak rôzne metriky presnosti.

10.1 Zber dát

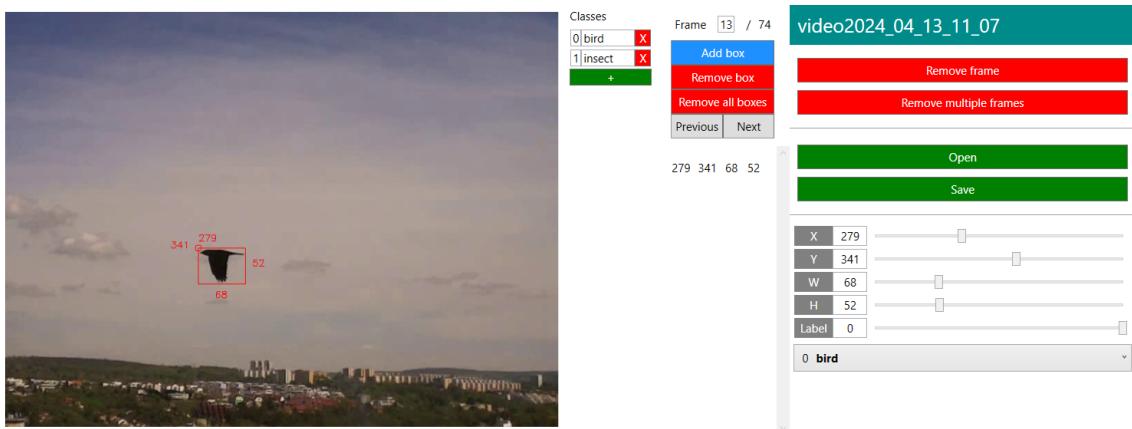
Reálne snímky z kamery boli zbierané použitím modulu aplikácie nazванého *video capture*, teda zber videa. Tento modul má za úlohu ukladať snímky získané z kamery, aj keď funguje rovnako aj so snímkami z uloženého videa. Výstup je ukladaný do video súboru vo formáte mp4. V prípade, že je medzi aktívnymi modulmi aj modul vytvárajúci orámovania, sú ukladané len tie snímky, ktoré obsahujú aspoň jeden detegovaný potenciálny objekt, teda aspoň jedno orámove. Jednotlivé orámovania sú ukladané do yaml súboru spolu s indexom snímky, ku ktorému patria.

Na zber dát bol použitý modul pre vytváranie dynamického modelu pozadia a modul detekcie objektov z binárneho obrazu, ktorý vzniká prahovaním rozdielového obrazu s modelom pozadia. Detegované objekty sú orámove a predané modulu na zber videa.

10.2 Úprava dát

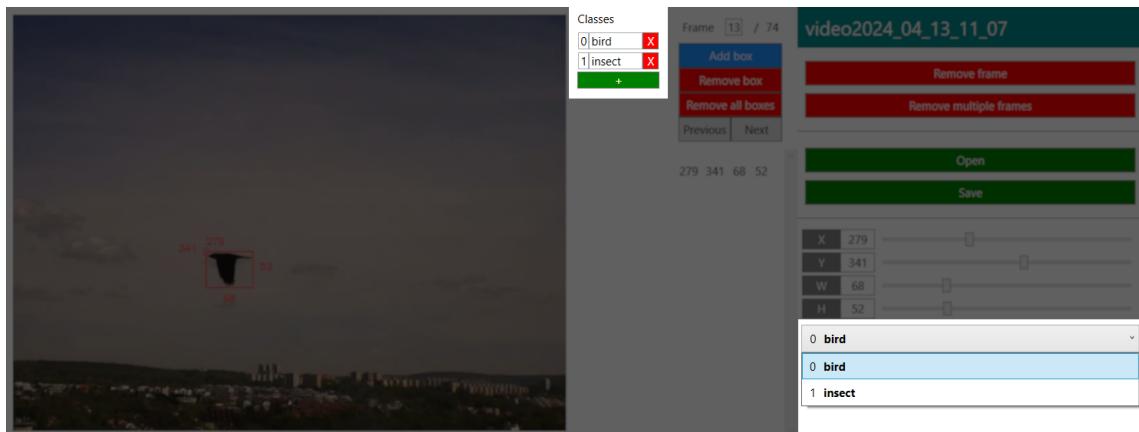
Pred použitím zozbieraných snímkov na vyhodnotenie a porovnanie prístupov k detekcii je nutné ich najprv anotovať, prípadne upraviť nesprávne orámove objekty, či falošne označené oblasti, ktoré objekt neobsahujú. Na tento účel bola vytvorená aplikácia na úpravu zbieraných videí.

Táto aplikácia, na rozdiel od hlavnej aplikácie detekcie a klasifikácie, bola napísaná v jazyku *C#*. Bol v nej použitý framework *WPF (Windows Presentation Foundation)* pre GUI a knižnica *OpenCV*, ktorá bola použitá aj v hlavnej aplikácii.



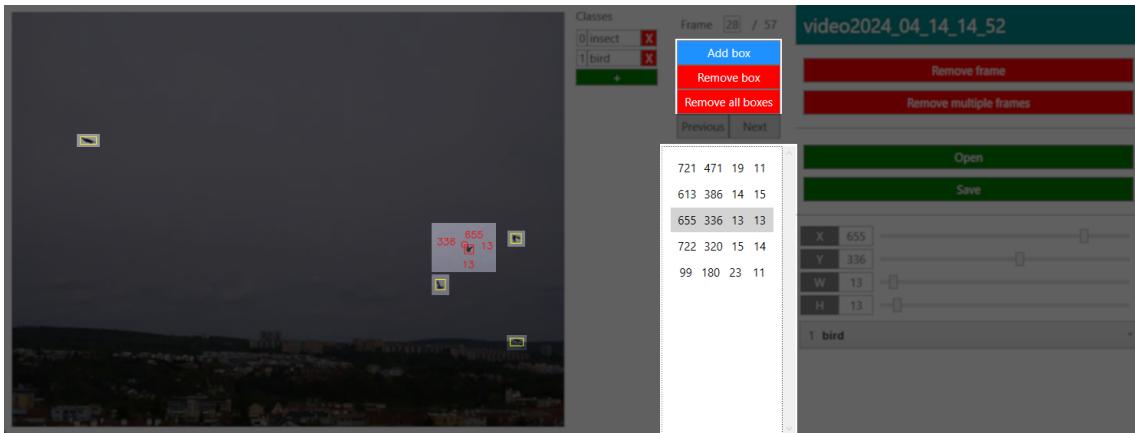
Obr. 10.1: Užívateľské rozhranie programu na anotáciu videa

Úprava videa začína načítaním súboru videa a súboru so zoznamom orámovaní. Užívateľ potom definuje triedy objektov nachádzajúcich sa v otvorenom videu. Novú definíciu triedy do zoznamu je možné pridať kliknutím na zelené zvýraznené tlačidlo. Triede je potom možné priradiť poradové číslo a názov. Červené tlačidlá vpravo od každej definícii slúžia na odstránenie definícii. Po vytvorení sa definícia zobrazí aj vo výberovom zozname v pravom dolnom rohu, z ktorého užívateľ volí zaradenie objektov vo videu do definovaných tried.



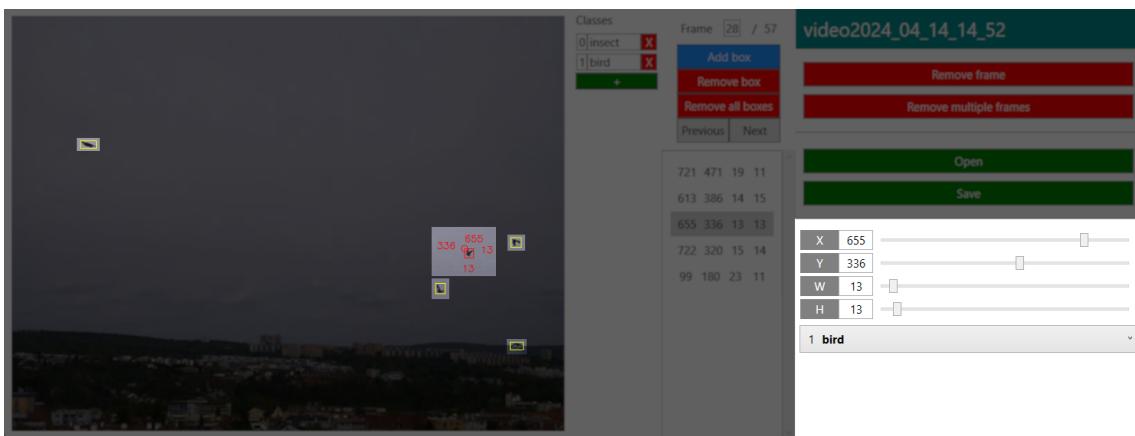
Obr. 10.2: Definícia tried v anotovanom videu

Orámovanie objektu, ktoré je práve upravované je najprv zvolené zo zoznamu. Je zvýraznené v ukážke aktuálneho snímku a sú pri ňom zobrazené jeho pozícia a veľkosť. Nové orámovania je možné pridať zvýrazneným modrým tlačidlom a odobrať červenými.



Obr. 10.3: Výber upravovaného orámovania

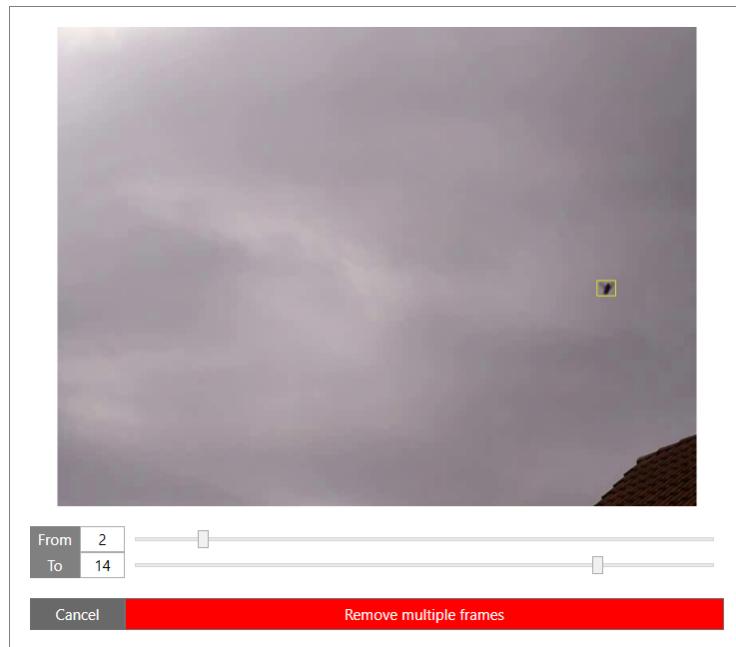
Zvolené orámovanie je možné presunúť, zmeniť jeho rozmery a nastaviť triedu objektu v ňom skupinou ovládacích prvkov vpravo dole. Hodnoty je možné zadávať priamo do textových políčok alebo posuvníkmi.



Obr. 10.4: Úprava vybraného orámovania

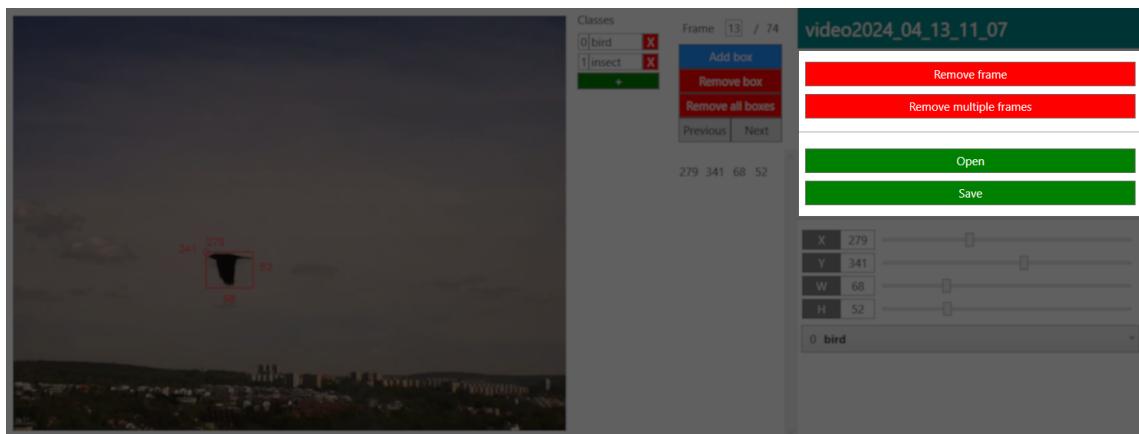
Ak sa na snímke nenachádza žiadna pravdivá detekcia, je možné ho z výsledného anotovaného videa vyniechať kliknutím na možnosť *Remove frame*.

V prípade, že bol zachytený väčší počet po sebe nasledujúcich snímok s falošnými detekciami, je možné ich z výsledného videa vyniechať zvolením možnosti *Remove multiple frames*. Po kliknutí na toto tlačidlo sa zobrazí modálne okno s možnosťou výberu rozsahu vynechaných snímok. Snímok zobrazený v tomto okne je bud prvý alebo posledný snímok vynechávanej sekvencie, podľa toho, ktorý užívateľ upravil ako posledný.



Obr. 10.5: Odstránenie sekvencie snímkov

Upravené a anotované video je nakoniec uložené opäť do dvoch súborov. Prvým súborom je znova video vo formáte mp4 a druhým yaml súbor obsahujúci orámovania so zaradením do tried pre každý snímok.



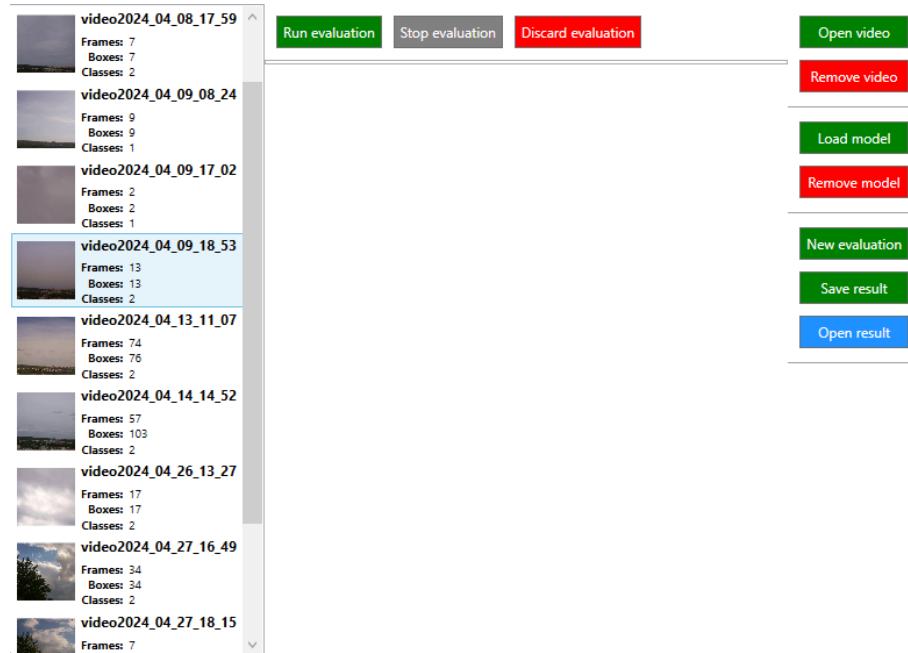
Obr. 10.6: Uloženie a úprava videa

10.3 Vyhodnotenie

Po získaní anotovaných záberov z kamery je možné porovnať kvalitu detekcie a klasiifikácie. Použitím natrénovaných modelov na tieto zábery a porovnaním ich výsledkov s anotáciou sú získané hodnoty potrebné pre výpočet metrík presnosti.

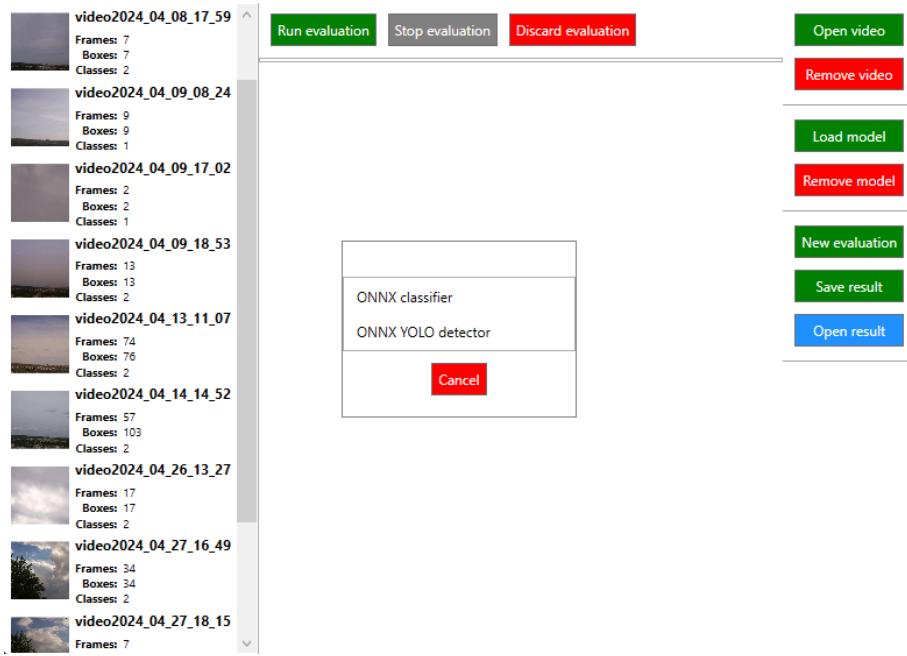
Na porovnanie a výpočet metrík bola vytvorená aplikácia s možnosťou načítať natrénované modely so zoznamom definícií tried a porovnať ich detekciu či klasifikáciu. K tomu načíta táto aplikácia aj videá anotované v predošлом kroku.

Táto aplikácia bola taktiež napísaná v jazyku *C#* s použitím GUI frameworku *WPF*. Na načítanie a používanie naučených modelov bola použitá knižnica *Microsoft.ML*. Modely sú načítavané zo súborov formátu *ONNX*.



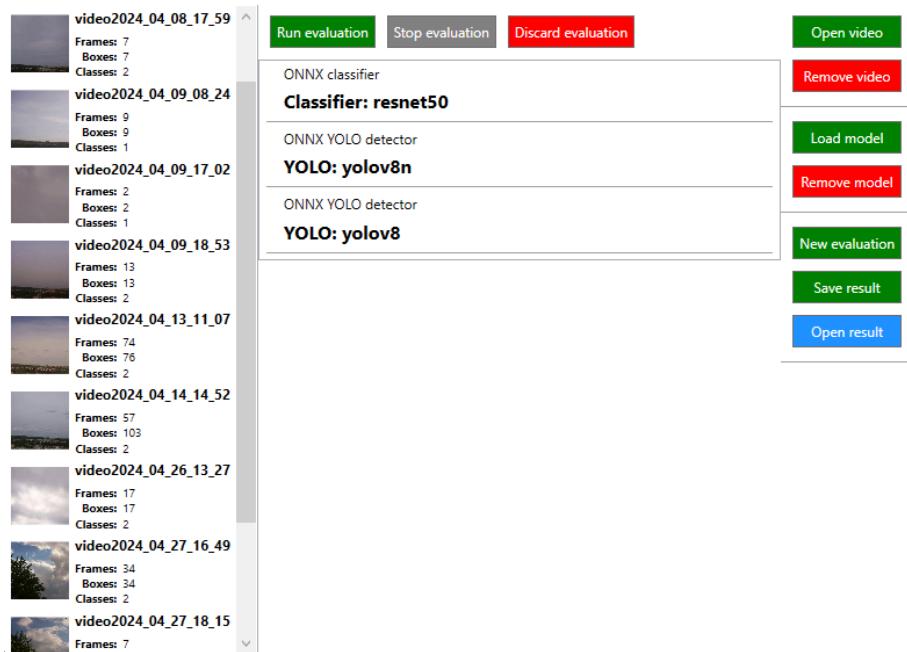
Obr. 10.7: Načítanie videí na porovnanie modelov

Po kliknutí na tlačidlo *Open video* sú užívateľovi po sebe zobrazené dva dialógové okná na otvorenia súboru s videom a súboru s anotáciou. Do zoznamu videí je možné pridať ľubovoľný počet videí, pričom pri vyhodnotení sú postupne zaradom použité ako vstup modelov. V zozname sa zobrazí názov videa, ukážka prvej snímky vo videu a počet snímok, orámovaní a tried vo videu.



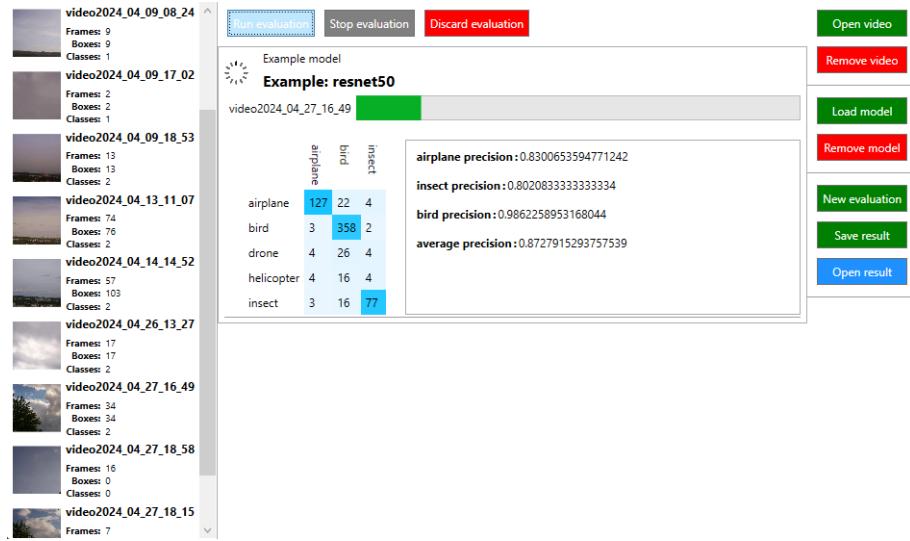
Obr. 10.8: Načítanie modelov na porovnanie

Modely je možné do aplikácie pridať pomocou tlačidla *Load model*. Po jeho stlačení je zobrazené modálne okno s možnosťou výberu typu modelu. Po zvolení typu sú užívateľovi opäť zobrazené dialógové okná na výber súboru s modelom a definíciou názvov tried podľa ktorých je porovávaná klasifikácia modelu a videa, nezáleží teda na poradí definovaných tried vo videu a pre výstup modelu.

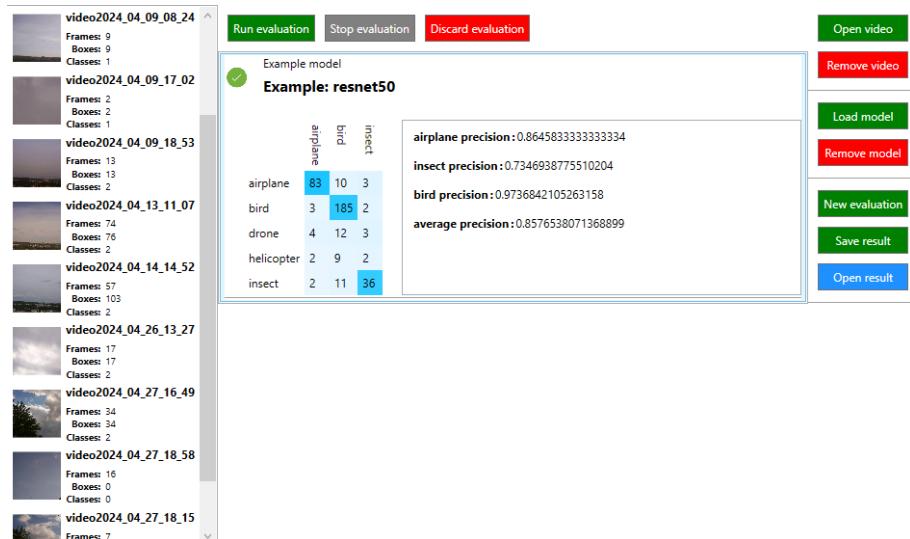


Obr. 10.9: Aplikácia na porovnanie s načítanými videami a modelmi

Načítať je možné niekoľko modelov a vyhodnocovať ich súčasne. Výsledky sú zobrazované priebežne počas ich výpočtu a pravidelne aktualizované. Počas vyhodnocovania je zobrazovaný jeho priebeh pre snímky práve spracovávaného videa. Zobrazovaná je matica zámien a hodnoty implementovaných metrík. V matici zámien sú v riadkoch zobrazené predikované, v stĺpcach pravdivé triedy podľa triedy. Zobrazujú sa len tie triedy ktoré sú definované pre daný model a všetky načítané videá.



Obr. 10.10: Prebiehajúci výpočet porovnania



Obr. 10.11: Zobrazenie výsledkov porovnani

Záver

V tejto práci bol popísaný návrh systému detekcie, trasovania a klasifikácie lietajúcich objektov. V kapitolách teoretickej časti práce boli vysvetlené teoretické základy postupov a metód detekcie objektov v obraze, ich popisu a následného zaradenia do tried. Dôraz bol kladený na rozdiely medzi metódami, ich výhody a nevýhody.

Nasledujúce kapitoly sa venovali samotnému návrhu systému. Bol navrhnutý a odôvodnený výber hardvérových komponentov a ich prepojenie. Pre jednoduchosť, spoľahlivosť a dostatočný výkon bola zvolená platforma *Raspberry Pi* s dotykovým panelom ako užívateľským rozhraním a kamerou od spoločnosti *Raspberry*.

Ďalej bol zdokumentovaný návrh databázy anotovaných snímkov určených na trénovanie klasifikátorov. Pri vytváraní bol použitý program *roboflow*. Výsledná databáza vznikla spojením datasetu z práce [1], niekoľkých voľne dostupných datasetov spolu s osobne získanými a anotovanými fotografiemi. Správnosť takto získanej databázy bola overená naučením modelu v systéme *roboflow* v nastavení na odskúšanie (fast). Na testovacej množine tento model dosahoval úroveň mAP 86,1 %.

Nakoniec bol popísaný softvér pre navrhovaný systém, jeho štruktúra a implementácia jeho častí. Spomenuté boli nástroje a knižnice použité pri jeho vývoji. Softvér zahrňuje moduly pre výber zdroja dát, detekciu objektov, ich klasifikáciu, sledovanie a kontrolu presnosti porovnaním s anotáciou. Systém umožňuje výber aktuálne používanej metódy a porovnanie medzi metódami. Kedže bol použitý programovací jazyk *python* a knižnice dostupné na väčšine zariadení a operačných systémoch, je možné program spustiť aj na bežnom počítači.

Ciele pokračovania tejto práce zahŕňajú ďalšie rozširovanie datasetu, implementáciu viacerých metód detekcie a klasifikácie a vyhodnotenie a porovnanie prístupov.

Literatúra

- [1] JUREČKA, Tomáš. Detekce a klasifikace létajících objektů. Brno, 2021. Diplomová práca.
- [2] JANOUŠEK, Jiří, Jan NOVOTNÝ, Petr MARCOŇ a Anna SIRUCKOVA. Algorithms for Flying Object Detection. In: 2018 Progress In Electromagnetics Research Symposium (PIERS-Toyama) [online]. Toyama, Japonsko, 2018, s. 782-783 [cit. 2024-01-03]. ISBN 978-4-8855-2316-8. ISSN 1559-9450. Dostupné z: doi:10.23919/PIERS.2018.8598196
- [3] MORSE, Bryan. Lecture 4: Thresholding [online]. Brigham, 1998 [cit. 2024-01-03]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf. Brigham Young University.
- [4] MORSE, Bryan. Lecture 9: Shape Description (Regions) [online]. Brigham, 1998 [cit. 2024-01-03]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/region-props-and-moments.pdf. Brigham Young University.
- [5] BUHL, Nikolaj. Image Thresholding in Image Processing. Encord [online]. 2023 [cit. 2024-01-03]. Dostupné z: <https://encord.com/blog/image-thresholding-image-processing/>
- [6] SAMINA. EDUCATIVE, INC. What is Canny edge detection? EDUCATIVE, INC. Educative [online]. Educative, 2023 [cit. 2024-01-03]. Dostupné z: <https://www.educative.io/answers/what-is-canny-edge-detection>
- [7] OPEN SOURCE COMPUTER VISION. How to Use Background Subtraction Methods [online]. 2023 [cit. 2024-01-03]. Dostupné z: https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html
- [8] LEUNG, Kenneth. How to Easily Draw Neural Network Architecture Diagrams [online]. In: . [cit. 2024-01-03]. Dostupné z: <https://towardsdatascience.com/how-to-easily-draw-neural-network-architecture-diagrams-a6b6138ed875>
- [9] LI, Erbo, Yazhou HUANG, Dong XU a Hua LI. Shape DNA: Basic Generating Functions for Geometric Moment Invariants [online]. Cornell University, 2017, 3-8 [cit. 2024-01-03]. Dostupné z: doi:10.48550/arXiv.1703.02242
- [10] LI, Erbo, Yazhou HUANG, Dong XU a Hua LI. Shape DNA: Basic Generating Functions for Geometric Moment Invariants [online]. Cornell University, 2017, 3-8 [cit. 2024-01-03]. Dostupné z: doi:10.48550/arXiv.1703.02242

Zoznam symbolov a skratiek

ReLU	Usmernená lineárna jednotka (Rectified Linear Unit)
CNN	Konvolučné neurónové siete (Convolutional neural network)
R-CNN	Konvolučné neurónové siete založené na regiónoch (Region-based CNN)
RoI	Regióny záujmu (Regions of Interest)
SVM	Metódy podporných vektorov (Support Vector Machine)
RPN	Siet návrhu regiónov (Region Proposal Network)
YOLO	You only look once
IoU	Pomer prieniku voči zjednoteniu (Intersection over Union)
tkinter	Rozhranie toolkitu Tcl/Tk (Tk interface)
csv	Súbor s čiarkou rozdelenými hodnotami (Comma Separated Values)
GUI	Grafické užívateľské rozhranie (Graphical User Interface)
mAP	Stredná priemerná presnosť (Mean Average Precision)
KNN	K najbližších susedov (K Nearest Neighbours)
NMS	Potlačenie nemaximálnej hodnoty (Non-Maximum Suppression)

Zoznam príloh