VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

PROTIDRONOVÁ OCHRANA PERIMETRU

ANTI-DRONE PERIMETER PROTECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ROMAN JANÍK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. MARTIN DRAHANSKÝ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Janík Roman**
Program: Informační technologie
Název: **Protidronová ochrana perimetru**
Anti-Drone Perimeter Protection
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte současné metody a nástroje pro detekci objektů ve videu. Vytvořte si přehled o současných přístupech k problematice detekce rychle se pohybujících a obtížně rozeznatelných objektů.
2. Navrhněte systém pro detekci a lokalizaci dronu či hejna dronů s ohledem na ochranu perimetru.
3. Vámi navržený systém z předchozího bodu implementujte.
4. Otestujte systém se skutečnými drony v terénu.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.

Literatura:

- Popovic V. et al. *Near-infrared high-resolution real-time omnidirectional imaging platform for drone detection*. In: Target and Background Signatures II, International Society for Optics and Photonics, 2016.
- Bandyopadhyay S. *Object of interest detection in video sequence using co-segmentation: A new era in video surveillance*. In: 4th International Conference on Recent Advances in Information Technology (RAIT), IEEE, 2018. s. 6.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. července 2019
Datum odevzdání: 31. července 2019
Datum schválení: 7. července 2019

Abstrakt

Rozvoj technologie dronů s sebou přináší příležitosti pro mnoho oblastí lidské činnosti, ale zároveň i bezpečnostní hrozby. Vzniká potřeba těmto hrozbám efektivně čelit. V této práci je popsána problematika a současné metody pro detekci objektů ve videu zachyceném pohybující se kamerou. Byl navržen systém pro detekci a lokalizaci dronu či hejna dronů. Algoritmus pro detekci je založen na konvoluční neuronové síti, konkrétně na algoritmu SSD. Konvoluční neuronovou síť byla natrénována na vlastním datasetu. Systém byl implementován pomocí knihovny OpenCV s možnou akcelerací algoritmu na GPU pomocí OpenCL. Vytvořené řešení bylo otestováno na videu.

Abstract

Developement of drone technology brings opportunities for many fields of human activity, but simultaneously brings security threats. A need to effectively face these threats arises. In this work is described the problematics and state-of-the-art methods for object detection in a video captured by moving camera. A system for detecting and locating a drone or a flock of drones has been proposed. Algorithm for detection is based on convolutional neural network, specifically on SSD algorithm. The convolutional neural network was trained on self-made dataset. The system was implemented using OpenCV library with possible algorithm acceleration on GPU using OpenCL. Created solution was tested on video.

Klíčová slova

detekce objektů, konvoluční neuronové sítě, Single Shot MultiBox Detector, TensorFlow, umělá inteligence, dron, bezpilotní letoun, video dohled, OpenCV, bezpečnostní hrozby, pohybující se kamera

Keywords

object detection, convolutional neural networks, Single Shot MultiBox Detector, TensorFlow, artificial intelligence, drone, UAV, video surveillance, OpenCV, security threats, moving camera

Citace

JANÍK, Roman. *Protidronová ochrana perimetru*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Martin Drahanský, Ph.D.

Protidronová ochrana perimetru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing., Dipl.-Ing., Martina Drahanského, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Janík
31. července 2019

Poděkování

Děkuji vedoucímu práce prof. Ing., Dipl.-Ing., Martinu Drahanskému, Ph.D. za možnost vypracovat zajímavou bakalářskou práci a vedení. Dále děkuji Ing. Michalu Hradišovi, Ph.D. a Ing. Tomáši Goldmannovi za jejich rady poskytnuté na konzultacích.

Obsah

Kapitola 1

Úvod

První snahy o umožnění počítačovým systémům vidět, tedy dát jim schopnost zpracovat a interpretovat informace z obrazových dat, byly vyvinuty již v 60. letech 20. století. Od té doby byl zaznamenán značný pokrok v počítačovém vidění, jak se oblast zabývající se těmito problémy nazývá. Navzdory stáří je počítačové vidění velmi aktuální a rozvíjející se oblast počítačové vědy. V současnosti prostupuje oblast počítačového vidění do mnoha technických oborů jako kontrola dopravy, medicína, kontrola kvality výrobků, vojenství, robotika a inteligentní dohledové systémy. Poslední tři příklady se týkají této práce.

Spolu s počítačovým viděním se v posledních letech i rapidně rozvíjely drony neboli bezpilotní letouny. V této práci je jako dron považován pouze bezpilotní letoun, i když je možné se setkat s použitím označení „dron“ pro jiné bezpilotní robotické zařízení jako loď, či ponorka. Dron je řízen na dálku člověkem, nebo létá autonomně řízen palubním počítačem. Součástí vybavení dronu jsou různé senzory, např. pro měření vzdálenosti, gyroskopy, akcelerometry, kamery atd. To umožňuje využití v mnoha oblastech lidské činnosti, například ve vojenství, zemědělství, fotografování ze vzduchu, kontrole dopravy a dohledových systémech. Drony dnes přispívají k rozvoji těchto oblastí a se zvyšující se dostupností těchto systémů běžné veřejnosti také roste jejich počet ve vzdušném prostoru.

Díky možnosti volně létat, umožňují drony nebývale mnoho způsobů zneužití této technologie. Výčet těchto bezpečnostních hrozeb zahrnuje narušení soukromí, narušení letového provozu, špionáž, pašování kontrabandu, zbraní a drog, zakládání požárů, teroristické útoky a atentáty. Drony mohou nést nebezpečné výbušniny, chemikálie, radioaktivní nebo biologické zbraně. Většina z výše uvedených hrozeb se již stala skutečností. Další negativní vlivy dronů jsou možné nehody se smrtícím potenciálem (pokud na člověka spadne několikakilogramový dron, následky mohou být fatální), střetnutí s jiným dronem, letadlem či ptákem a plašení zvěře, kdy ptáci na drony útočí. Technologii dronů lze považovat za ukázkový případ předběhnutí legislativy technikou a v současnosti státy zavádí nové zákony a regulace pro provoz těchto strojů.

Vzniká potřeba se proti výše uvedeným bezpečnostním hrozbám a rizikům aktivně bránit a předcházet jim. Protože je technologie dronů relativně nová a ještě před několika lety nebyly drony tak rozšířeny jako je tomu dnes, je i obrana proti dronům ve svých počátcích. K obraně proti dronům je nutná v prvé řadě jejich detekce a lokalizace, lokalizace jejich pilotů, dále pak rušení řídícího signálu, oslepení pomocí kouře, sestřelení, zachycení do sítě, spuštění poplachu, evidence událostí a jejich hlášení. K těmto se účelům používají radary, detektory řídících rádiových signálů (Wi-Fi), mikrofony, kamery a infrakamery. Za nejlépe rozvinutou technologii lze označit detekci radiových signálů, kde již existují komerční

řešení. Nevýhodou radaru je obtížná detekce malých dronů, kvůli malé odrazové ploše. Detekci mikrofonem negativně ovlivňuje malý dosah a šum. Infrakamery mají v současnosti malé rozlišení a drony jsou obtížně rozeznatelné díky malému kontrastu oproti okolnímu vzduchu. Na druhou stranu představují řešení pro detekci v noci. Zbývající alternativou je nahrávání obrazu ve viditelném spektru a detekce dronů pomocí algoritmů počítačového vidění. V průběhu vyhledávání zdrojů jsem narazil na jedno komplexní komerční řešení uvedeného problému. Zahrnuje detekci dronů pomocí rádiových signálů, zvuku, infrazáření a také obrazu zpracovaného klasifikační konvoluční neuronovou sítí. Tyto vstupy pak umožňuje propojit s rozhodovací logikou, aktivním hlášením událostí (např. SMS, email), informačním systémem a se stávajícím bezpečnostním systémem.

Cílem této práce je analýza současných metod pro detekci pohybujících se objektů ve videu nahraném pohybující se kamerou. Tento úkol je velmi obtížný, protože je nutné vyřešit problémy týkající se dvou rozdílných pohybů ve videu. Dalšími cíli jsou návrh, implementace, testování a vyhodnocení systému pro detekci dronu či hejna dronů. Implementovaná metoda bude pracovat co nejrychleji, zdrojem bude video nebo výstup z kamery. Poté bude metoda otestována jak na offline videu tak na video výstupu z kamery přímo v terénu se skutečnými drony. Závěrem bude vyhodnocena úspěšnost detekce.

Tato práce je součástí projektu řešeného ve spolupráci s Univerzitou obrany v Brně. Jejím přínosem bude možnost začlenění do projektu a pro seznámení čtenáře se současnými metodami pro detekci pohybujících se objektů.

Práce je dále členěna do pěti kapitol: druhá kapitola obsahuje úvod do problematiky detekce a sledování objektů, problémy s tím spojené a metriky, které se používají pro vyhodnocení detekčních algoritmů. Dále popisuje současné metody pro detekci pohybujících se objektů jako konvoluční neuronové sítě a kosegmentace. Návrh systému je popsán ve třetí kapitole. Poslední dvě kapitoly popisují implementaci navrženého systému a jeho vyhodnocení. V rámci závěru jsou diskutovány možnosti budoucího vývoje.

Kapitola 2

Přehled současného stavu

V této kapitole jsou popsány základní pojmy zpracování obrazu, které je zapotřebí znát pro porozumění následujícím kapitolám a principům detekce objektů ve videu. V kapitole jsou vysvětleny pojmy reprezentace digitálního obrazu, předzpracování obrazu, detekce objektů a sledování objektů, dále pak výzvy a problémy, týkající se detekce pohybujících se objektů a metriky, které se běžně používají pro vyhodnocení kvality detekčních algoritmů.

2.1 Předzpracování obrazu

Před samotnou aplikací algoritmů počítačového vidění je často nutné zdrojový obraz upravit tak, aby co nejvíce vyhovoval následujícím algoritmům v posloupnosti zpracování. Tyto úpravy se liší podle použitých algoritmů.

Standardní úpravou je změna velikosti obrazu, tedy zvětšení/zmenšení počtu řádků/sloupců v obraze. Zmenšením vstupního obrazu se obecně dosahuje zrychlení algoritmů, protože klesají časová a paměťová náročnost algoritmů, ovšem na úkor přesnosti. Algoritmy počítačového vidění jsou obvykle náročnější na výpočet, standardní časové složitosti jsou linearitmická $O(n \log n)$, kvadratická $O(n^2)$ a kubická $O(n^3)$.

Mnoho algoritmů nepotřebuje zpracovávat barevnou informaci, proto pracují se šedotónovým obrazem, kde jas je popsán odstíny šedi. Příklady algoritmů vyžadujících šedotónový obraz jsou detektory hran, které pracují s náhlou změnou jasu, nebo optický tok, kde se pohyb v obraze určuje podle jasu jednotlivých pixelů. Výhodou jsou také menší výpočetní nároky než u barevného obrazu. Šedotónový obraz můžeme získat z šedotónové (jednobarevné – mono-color) kamery nebo převodem z barevného modelu. Pro převod z modelu RGB se používá vzorec

$$I = 0,299R + 0,587G + 0,114B \tag{2.1}$$

kde I je výsledný odstín šedi (jas) a R, G, B jsou barevné složky modelu RGB [39]. Jednotlivé barvy jsou v součtu váhovány, protože lidské oko je na barvy různě citlivé.

Podobnou úpravou je převod šedotónového obrazu na binární (černobílý). Jednou z metod pro tuto úpravu je prahování, kde je hodnota jasu pixelu porovnávána s určeným prahem (threshold). Pokud je hodnota jasu menší než práh, pixel bude černý, jinak bílý. Tato jednoduchá metoda je často součástí mnoha algoritmů, např. u odčítání pozadí a dá se použít pro odstranění odlehlých hodnot (outliers). Obecný vzorec pro prahování je:

$$f(x) = \begin{cases} A & \text{pro } x < T \\ B & \text{pro } x \geq T \end{cases} \tag{2.2}$$

kde x je vstupní hodnota, $f(x)$ je výsledná hodnota, T je práh, A je nová hodnota pro x pod prahem a B je nová hodnota pro x nad prahem.

Pro vyhlazení obrazu se velmi často využívá Gaussova vyhlazování. Tato metoda redukuje šum a rozmazává hrany. Jedná se o konvoluční filtr, kde konvoluční jádro jsou váhy dané Gaussovou funkcí. Vzorec pro Gaussovo vyhlazování je:

$$h(x, y) = f(x, y) \ * \ G(x, y) \tag{2.3}$$

kde $f(x, y)$ je vstupní pixel, $h(x, y)$ je výsledný pixel a $G(x, y)$ je Gaussova funkce:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

2.2 Detekce objektů

V následujících řádcích bude definován termín detekce objektů ve videu. Video je posloupnost obrazů, kde každý z těchto obrazů se nazývá rámec. Obsah dvou po sobě jdoucích rámců spolu obvykle souvisí. Detekce objektu(ů) je zjištění přítomnosti a polohy objektu ve videu [23]. Tuto definici lze rozšířit na nalezení nejmenšího obdélníku ohraničující objekt [8].

Obrázek 2.1: Příklad detekce objektu, objekt je ohraničený obdélníkem (bounding boxem) [24].

Metody pro detekci pohybujících se objektů lze rozdělit na tři hlavní kategorie. Metody první kategorie spoléhají na vzhled objektů v jednotlivých snímcích, druhé kategorie spoléhají na informaci o pohybu mezi rámci a metody třetí kategorie kombinují obě předchozí metody [45].

2.2.1 Metody založené na vzhledu

Tyto metody pracují v prostorové doméně (*spatial domain*). Zahrnují informace obsažené pouze v jednom obraze, lze je tedy využít i pro detekci objektů z fotografie. Každý objekt má své charakteristické rysy (*features*), které ho popisují. Těmito rysy můžou být rohy, hrany, barvy, regiony, tvary nebo textury. Deskriptor popisuje jeden nebo více rysů. Příklady deskriptorů jsou průměr (např. průměrná barva) nebo histogram (histogram orientovaných gra dientů). Jako příklady algoritmů extrahující deskriptory lze uvést Speeded Up Robust Features (SURF) [10] a Histogram of Oriented Gradients (HOG) [15].

Následně je na deskriptorech natrénován klasifikátor, který zařadí objekt do příslušné třídy. Často používané klasifikační algoritmy jsou AdaBoost [19] a Support Vector Machines (SVM) [14]. Poté se obvykle použije metoda Sliding window, kde se vytvoří okno nad obrazem, které se postupně posouvá. Pokud klasifikátor vyhodnotí, že se v okně nachází rozpoznávaný objekt, vrátí se okno jako bounding box (ohraničující obdélník) [8].

Dalším přístupem založeným na strojovém učení jsou konvoluční neuronové sítě (CNN – Convolutional Neural Networks). Od předchozích dvou algoritmů se liší tím, že deskriptory jsou získány přímo pomocí CNN [8].

Nevýhodou algoritmů založených na strojovém učení je nutnost mít rozsáhlou trénovací množinu dat. Pro běžné aplikace jako detekce obličeje, chodců nebo automobilů existují trénovací množiny volně dostupné na Internetu, ovšem pro detekci dronů téměř neexistují, protože se jedná o relativně nový problém. Pánové Aker a Kalkan [8] tento problém vyřešili vytvořením vlastní umělé trénovací množiny. Další nevýhodou je, že tyto algoritmy pracují nejlépe, když jsou hledané objekty dostatečně velké a jasně viditelné. Je obtížné detekovat malý objekt, což je žádoucí u detekce dronů [45].

Alternativou k metodám strojového učení je kosegmentace [44] [37]. Tato metoda segmentuje současně společnou část více obrazů [9]. Kosegmentace je speciální případ segmentace, kdy na základě vyhledávacího obrazu (*query image*) je v jiném obrazu nebo rámci videa nalezena společná část obrazů. Obrovskou výhodou této metody je to, že kromě vyhledávacího obrazu nepotřebuje žádnou předchozí informaci ani lidský dohled [9]. V porovnání s metodami strojového učení odpadá trénování klasifikátorů na tisících příkladů.

2.2.2 Metody založené na pohybu

Metody založené na pohybu pracují v časové doméně (*temporal domain*). Pohybující se objekt lze popsat jako množinu pixelů v rámci videa mající souvislý pohyb v čase a sémantickou podobnost v prostoru obrazu [56]. Pokud detekce zahrnuje pouze informaci o pohybu, detekují se všechny pohybující se objekty ve videu. To je nežádoucí, protože spolu s cílovým objektem (např. auto) se detekují i ostatní objekty jako kymácející se stromy, plovoucí oblaka apod.

Metoda odčítání pozadí pracuje tak, že se od jednotlivých pixelů aktuálního rámce odečítají pixely pozadí. Výsledkem metody je maska změn v obraze, ze které je možné vysegmentovat objekt. Pro zlepšení výsledku se používá prahování. Tato metoda je velmi rychlá, je však poměrně nepřesná, protože je založená na hypotéze statického pozadí. Proto není vhodná pro video zachycené pohybující se kamerou.

Jinou široce používanou metodou je optický tok. Optický tok je rozdělení zjevných rychlostí pohybu jasových vzorů v obraze [26]. Lze si jej představit jako 2D pole vektorů, kde vektory vyjadřují směr a velikost pohybu jednotlivých pixelů mezi dvěma obrazy. Tato metoda je vhodná pro video zachycené pohybující se kamerou, je ovšem poměrně výpočetně

náročná. Proto se optický tok běžně nepočítá pro všechny body v obraze, ale pouze pro podmnožinu charakteristických bodů jako jsou rohy [40].

2.2.3 Hybridní metody

Hybridní metody kombinují informace o vzhledu a pohybu objektu, pracují v prostorovo-časové doméně (*spatio-temporal domain*). Teoreticky jsou hybridní metody nejlepším přístupem pro detekci objektů, protože pracují s více informacemi než metody předchozích dvou kategorií. Prací implementujících tyto metody je výrazně méně než těch pracujících v prostorové nebo časové doméně. Příkladem je [45], kde je použit koncept st-cubes (spatio-temporal cubes – prostorovo-časové kostky), které jsou složeny z několika rámců videa. Dále je kompenzován pohyb objektů mezi snímky a použit klasifikátor AdaBoost. Jako další příklad je možné uvést [53], kde autoři kombinují optický tok, SVM algoritmus a konvoluční neuronové sítě.

Některé z výše uvedených algoritmů budou podrobně vysvětleny v následující kapitole. Většinou je detektor objektů tvořen více než jedním algoritmem a algoritmy jsou kombinovány a různě modifikovány tak, aby co nejlépe plnily účel výsledné aplikace (např. detekce obličeje). Obecně se dá říci, že čím více informací je z videa extrahováno, tím přesnější je detekce objektu. S tím je ovšem spojena vyšší výpočetní náročnost. Výpočetní výkon počítačů se ovšem stále zvyšuje, algoritmy je možné akcelerovat výpočtem na grafické kartě (GPU) nebo pomocí obvodu FPGA (Field Programmable Gate Array). Navzdory desetiletím výzkumů neexistuje dokonalý algoritmus pro detekci objektů, a proto je detekce objektů stále otevřená oblast aktivního výzkumu.

2.3 Sledování objektů

Termín sledování objektů (*object tracking*) často splývá s termínem detekce objektů, ale tyto termíny se vzájemně liší, ačkoli mají mnoho společného. Sledování objektů je lokalizace pohybujícího se objektu v po sobě jdoucích rámcích videa [56]. Metody sledování objektů lze rozdělit do dvou hlavních kategorií: klasické sledovací metody a sledování detekcí.

Obecný koncept klasických sledovacích metod se skládá ze čtyř kroků. Prvním je výběr sledovaného objektu a to buď uživatelem, nebo automaticky detekčním algoritmem. Druhým krokem je extrakce rysů objektu. Třetím krokem je nalezení nejlepšího shody v následujícím rámci pomocí extrahovaných rysů. Posledním krokem je aktualizace rysů podle nalezené shody. Dále se opakují kroky 3 a 4 [56].

Detekce objektů a extrakce rysů byla popsána v předchozí podkapitole. K nalezení nejlepší shody se používají algoritmy založené na pravděpodobnosti. Jako příklady lze uvést metody založené na kernelu (Mean-Shift [13]), kde se počítá pravděpodobnost výskytu barvy pixelu objektu v předchozím rámci. Poté se prohledává okolí předchozí pozice objektu, výsledná pozice je dána největší pravděpodobností. Jinou metodou je částicový filtr (*particle filter*), částice (vzorky) jsou obdélníky reprezentující hypotetické umístění sledovaného objektu, které jsou rozmísťovány náhodně v okolí předchozí pozice objektu. V obdélnících je poté vypočtena podobnost pomocí histogramu barvy. Částice s největší podobností určují polohu objektu. Tato metoda byla podrobně popsána a použita v [38]. Pro sledování objektů se také používají metody založené na Kalmanově filtru nebo konvolučních neuronových sítích.

Sledování detekcí (*tracking-by-detection*) spočívá v detekování objektu v každém rámci videa. Tento přístup byl zvolen v [9].

2.4 Problémy spojené s detekcí pohybujících se objektů

Detekce pohybujících se objektů ve video sekvencích zachycených pohybující se kamerou je spojená s mnoha problémy. V práci *New Trends on Moving Object Detection in Video Images Captured by a moving Camera: A Survey* [56] shrnuli autoři hlavní problémy. V této podkapitole budou tyto problémy stručně popsány a uvedeny do kontextu s detekcí dronů.

- **Variance osvětlení** – osvětlení scény a sledovaného objektu se může měnit v čase. Důvody jsou pohyb zdrojů světla, odraz světla, pokles intenzity zdroje světla, ve venkovních scénách počasí, zakrytí zdroje světla atd. Důsledkem může být selhání metod založených na vzhledu.

  Detekce dronů bude probíhat výhradně ve venkovních scénách. Velkou roli zde hraje počasí, hlavně oblačnost má největší vliv na světelné podmínky. Přechod mraků způsobuje prudkou změnu osvětlení. Dále může dojít k odrazu světla od střech, vodních ploch, oken nebo skel automobilů. Ještě většími problémy jsou meteorologické jevy, které výrazně ovlivňují šíření světla ve scéně jako déšť, sníh, mlha nebo písečná bouře. Nedostatek světla v noci, při těžké oblačnosti apod. výrazně snižuje, až znemožňuje pravděpodobnost detekce.

Obrázek 2.2: Příklad variance osvětlení. Na snímcích jsou patrné změny vlivem denní doby a počasí [3].

- **Změna vzhledu pohybujícího se objektu** – video zachycuje pouze 2D projekci 3D scény, takže jakákoli rotace podél třetí osy může změnit vzhled objektu. Jiné změny vzhledu objektů mohou být: změna výrazu obličeje, roztáhnutí křídel, nafouknutí balónu, změna oblečení atd.

  Tento problém je u detekce dronů minimální, naprostá většina dronů svůj vzhled za letu nijak nemění.

- **Prudký pohyb** – prudké změny rychlosti a směru pohybu objektu nebo kamery můžou způsobit ztrátu objektu sledovací metodou (trackerem) nebo velkou chybu v odhadu polohy objektu.

  Drony jsou schopny prudkého pohybu, většinou ale ne tak zásadního, aby způsobil ztracení trackerem. To je ale možné při poryvu silného větru.

- **Okluze** – sledovaný objekt může být částečně nebo zcela zakryt jiným objektem, např. stromy, vozidly, budovami nebo chodci.

Obrázek 2.3: Příklad změny vzhledu objektu - skládací dron [18].

Obrázek 2.4: Příklad prudkého pohybu - srážka dvou aut [5].

Obrázek 2.5: Příklad okluze objektu. Auto je částečně zakryto pilířem mostu.

Drony můžou být zakryty zejména vegetací, terénem a v obydleném území navíc budovami a případně dopravními prostředky.

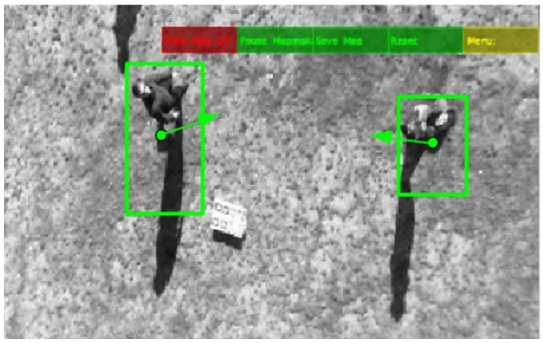
- **Komplexní pozadí** – ve scénách s komplexním pozadím může mít i lidské vidění problém s detekcí objektu. Platí to hlavně pro venkovní scény. V pozadí se také může vyskytovat pohyb, jako kymácející se stromy, plovoucí oblaka, vodní vlny atd. Díky tomu některé algoritmy nemusí objekt detekovat.

  Pro detekci dronů může být tento problém kritický. Zvláště může být obtížné detekovat dron ve velké vzdálenosti, kdy je ve videu pouze na několika desítkách pixelů. Dron může splývat s pozadím - např. šedý dron na šedém mraku nebo dron na pozadí s hustou vegetací.

Obrázek 2.6: Příklad komplexního pozadí [45].

- **Stín** – mnoho algoritmů nedokáže odlišit objekt od jeho stínu a detekuje jej jako součást objektu, bounding box tak obsahuje jak objekt, tak jeho stín. Tento problém se vyskytuje zejména u video sekvencí zachycených z výšky.

  Pro detekci dronů stín nepředstavuje žádný problém, protože stín dronu je velmi malý a většinou mimo záběr kamery.



Obrázek 2.7: Příklad přítomnosti stínu v obraze [40]. Bounding box osoby vlevo zahrnuje velkou část stínu.

- **Problémy spojené s kamerou** – kamera je důležitou součástí systému pro detekci objektů. Kamery mají různé senzory, čočky, rozlišení, rychlost snímání a vnitřní obvody pro zpracování obrazu. Nekvalitní video sekvence může znemožnit detekci objektů. Dalšími problémy jsou rozmazání vznikající vibracemi kamery a blokové artefakty vznikající při kompresi videa. Kvalitu video sekvence také výrazně zhoršuje přítomnost šumu.

- **Pohyb kamery** – Detekce objektů ve videu zachyceném pohybující se kamerou je složitější než ve videu zachyceném stabilní kamerou. Je nutné rozpoznat pohyb kamery od pohybu objektů a kompenzovat pohyb kamery. Tento úkol ovšem není jednoduchý, kamera se může pohybovat ve směru všech tří os prostoru. Pohyblivé bezpečnostní kamery umožňují jednoduchý pohyb PTZ (Pann – Tilt – Zoom). Složitější je řešení u kamer připevněných na létající prostředky (dron, letadlo), kde se kamera může pohybovat libovolně v 3D prostoru.

  V této práci bude použita pozemní kamera s možností PTZ pohybů.

Obrázek 2.8: Příklad nekvalitního obrazu [6].

Obrázek 2.9: Příklad obrazu zachyceného pohybující se kamerou, zde připevněnou ke dronu [24].

- **Deformace nepevných objektů** – nepevný pohybující se objekt je takový objekt, jehož části mohou mít různý pohyb. Příkladem je chodec, který při chůzi pohybuje končetinami, nebo pták mávající křídly. Mnoho algoritmů pak detekuje jednotlivé části objektu jako rozdílné objekty.

  Mezi nepevné objekty se řadí i drony, kde vrtule dronu mají rozdílný pohyb od dronu jako celku.

Obrázek 2.10: Příklad deformace nepevných objektů.

2.5 Metriky pro vyhodnocení detektorů

Tato podkapitola byla částečně převzata z práce pánů Yazdiho a Bouwmanse [56]. Pro vyhodnocování a porovnávání algoritmů pro detekci pohybujících se objektů je zapotřebí stanovit kvantitativní metriky, které vyčíslují kvalitu algoritmu. Pro použití metrik je nutné mít tzv. ground-truth obrazy, zjednodušeně binární masky objektů v obraze. Objekt je v ground-truth obraze vyznačen bílými pixely, zatímco pozadí černými, viz obr 2.11. Problémem je získání takovýchto masek, obvykle se musí vytvářet ručně, což je pro velké množiny dat zdlouhavé. Ground-truth obraz umožňuje použití obvyklých metrik pro binární klasifikaci dle následujících parametrů:

- **True Positive (TP)** – počet pixelů detekovaného objektu odpovídajících detekovaným pixelům v ground-truth.
- **False Positive (FP)** – počet pixelů detekovaného objektu odpovídajících nedetekovaným pixelům v ground-truth.
- **True Negative (TN)** – počet pixelů nedetekovaného objektu (pozadí) odpovídajících nedetekovaným pixelům v ground-truth.
- **False Negative (FN)** – počet pixelů nedetekovaného objektu (pozadí) odpovídajících detekovaným pixelům v ground-truth.

Z výše uvedených metrik se skládají nejpoužívanější metriky:

- **Precision** – měří procento všech detekovaných pixelů náležícím objektu.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

- **Recall** – měří procento všech pixelů náležícím objektu, které jsou správně detekovány.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Jinou používanou metrikou je **Accuracy**, která měří procento všech pixelů, které jsou správně detekovány a odmítnuty:

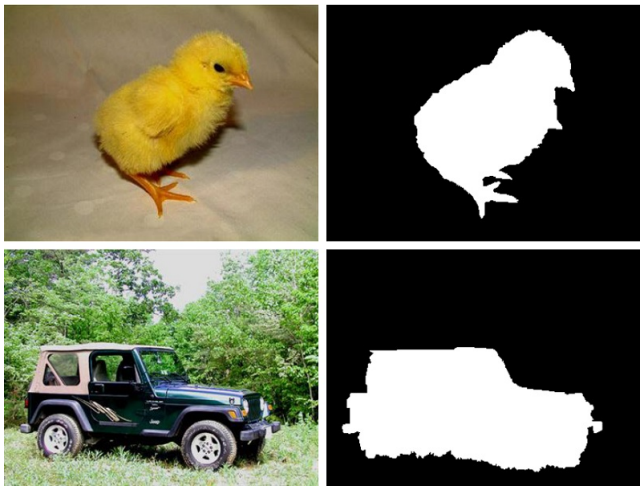
$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.7}$$

Pro metody sledování objektů, které reprezentují objekt bounding boxem se používá **Intersection over union (IoU)**, což je poměr přesahující oblasti předpokládaného bounding boxu a ground-truth bounding boxu k sjednocení obou oblastí:

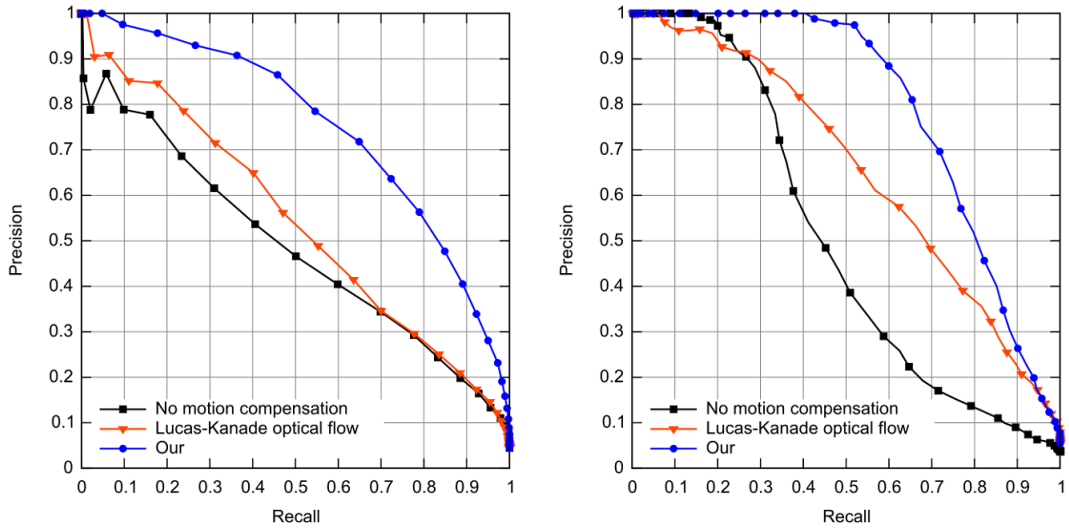
$$Intersection\ over\ union = \frac{A \cap GT}{A \cup GT} \tag{2.8}$$

kde A je oblast předpokládaného bounding boxu a GT oblast ground-truth bounding boxu. Metriku lze také použít pro vyhodnocení segmentace/kosegmentace. V některých pracích je tato metrika nazvána *overlap ratio*.

Často je autory používána reprezentace výsledků pomocí křivky. Nejpoužívanějším křivkou je **Precision-Recall (PR)**. Precision-Recall křivka je vytvořena vykreslením Precision



Obrázek 2.11: Příklady ground-truth obrazů: vlevo originální obrazy, vpravo jejich ground-truth obrazy [12].



Obrázek 2.12: Příklady Precision-Recall (PR) křivek [45].

proti Recall, příklady na obr 2.12. *Čím více je křivka blíže k pravému hornímu rohu, tím lepší je výkon metody* [8].

Spojením Precision a Recall vzniká **Average Precision (AP nebo AveP)** podle vzorce:

$$Average\ Precision = \int_0^1 p(r)\,\mathrm{d}r \tag{2.9}$$

kde p je Precision a r je Recall, jedná se tedy o integrál plochy pod Precision-Recall křivkou. Výsledná hodnota je vždy v intervalu $< 0, 1 >$.

Jednotliví autoři používají různé vlastní metriky, často vytvořené úpravou výše popsaných metrik. Bandyopadhyay [9] použil metriku *Error-rate*, která je vypočítána vzorcem:

$$Error - rate = 1 - \frac{A \cap GT}{A \cup GT} \tag{2.10}$$

jedná se tedy o inverzní hodnotu Intersection over union.

V některých pracích autoři upravují definice základních metrik, např. Aker a Kalkan [8] považují předpokládaný bounding box jako True Positive, když je Intersection over union větší než 50 %. Při vyhodnocování a porovnávání s jinými pracemi autoři často uvádějí použité množiny dat (datasety), počet obrazů, videí a rámců videí, jejich rozlišení a případně i velikost objektů v obrazech. Výsledky metod pro detekci objektů závisí na vstupních datech, zvolených kritérií pro vyhodnocení a při vyhodnocení potřebného času na použitém hardwaru. Metoda A dávající lepší výsledky než metoda B na určité množině dat může na jiné množině dat dávat horší výsledky. Důležité je při porovnávání používat stejnou datovou množinu a stejný formát dat pro všechny porovnávané metody. Otázkou je, jestli je možné mít vždy lepší výsledky, pokud se zvolí ty „správné“ množiny dat a porovnává se proti „správným“ pracím.

Autoři ve svých pracích většinou nevyhodnocují časovou náročnost metod, ale pokud tak učiní, uvádí některé z těchto údajů: časovou složitost, průměrnou dobu detekce pro jeden obraz (rámec), počet snímků za sekundu (FPS) a údaje o použitém hardwaru jako model a frekvence procesoru a velikost operační paměti.

Jedním z široce používaných datasetů pro vyhodnocování metod je Pascal VOC dataset [16]. Pascal Visual Object Classes (VOC) je projekt, který zveřejňoval standardizované datasety pro vyhodnocování, umožňuje vyhodnocovat a porovnávat mezi metodami a vyhlašoval soutěže. Kategorie soutěží byli klasifikace, detekce objektů, segmentace, klasifikace akcí a detekci části lidského těla. Dalším a novějším datasetem je COCO (Common Objects in Context) dataset [33]. Pomocí datasetu lze vyhodnocovat metody detekce objektů, segmentace a klasifikace. Aktuálně obsahuje více než anotovaných 200 000 obrazů, 1,5 milionu objektů a 80 kategorií objektů.

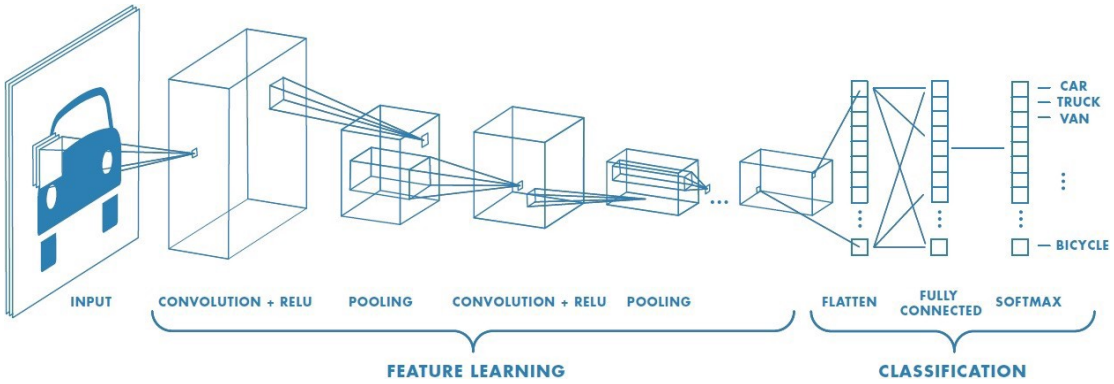
2.6 Konvoluční neuronové sítě

Konvoluční neuronová síť (CNN - *Convolutional Neural Network*) je algoritmus strojového učení. Jedná se o typ neuronové sítě, kde je alespoň v 1 vrstvě použita konvoluce místo násobení matic [22]. V této podkapitole budou stručně popsány pouze konvoluční neuronové sítě a jejich příklady. Použití běžných plně propojených dopředných neuronových sítí pro zpracování obrazu není vhodné, protože by počet neuronů byl velmi velký. Konvoluční neuronová síť tento problém řeší konvolucí, která snižuje počet volných parametrů a umožňuje tak vytvářet hlubší sítě. Konvolucí se využívá prostorových závislostí mezi sousedními pixely na rozdíl od plně propojených dopředných neuronových sítí. Díky tomu je konvoluční neuronová síť méně náchylná na overfitting, situaci, kdy síť je tak úzce přizpůsobená na trénovací data, že je těžké generalizovat pro nová data.

Konvoluční neuronové sítě byly inspirovány zrakovými systémy zvířat, vzor spojení mezi neurony je podobný organizaci zvířecí vizuální kůry. Jednotlivé neurony odpovídají na vzruchy pouze v omezené oblasti zorného pole, známé jako receptivní pole. Jednotlivá receptivní pole se částečně překrývají tak, že je pokryté celé zorné pole.

Použití je především ve zpracování obrazu, jmenovitě klasifikace obrazů a videí, segmentace a kosegmentace, detekce objektů, rozpoznávání obličejů, rozpoznávání akcí atd., dalšími příklady je zpracování zvuku a přirozeného jazyka.

Konvoluční neuronová síť je organizována do vrstev, kde každá vrstva má svou šířku, výšku a hloubku. Síť se skládá ze vstupní, výstupní a několika skrytých vrstev. Skrytými vrstvami je konvoluční vrstva, pooling vrstva, plně propojená vrstva a normalizační vrstva. Typickou architekturu lze vidět na obr 2.13. Vstupní obraz je připojen na první konvoluční vrstvu, následuje několik konvolučních a pooling vrstev, které provádí extrakci rysů a dále



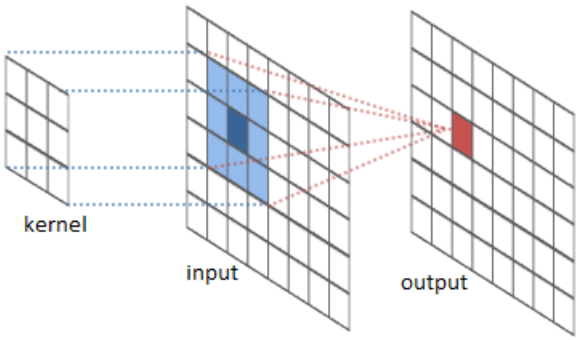
Obrázek 2.13: Typická architektura konvoluční neuronové sítě [46].

jsou na ně připojeny plně propojené vrstvy normalizační vrstva, které provádí klasifikaci. Velikost obrazu se postupně zmenšuje.

Konvoluce je matematická operace, kde pro dvourozměrný diskrétní obraz je dána vzorcem:

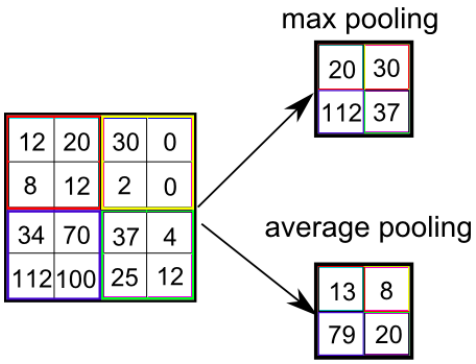
$$(f * h)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x - i, y - j) \cdot h(i, j) \tag{2.11}$$

kde f je vstupní obraz, h je kernel (filtr), x a y značí souřadnice v obraze a i a j značí souřadnice kernelu. Hodnoty vstupního obrazu jsou násobeny příslušnými hodnotami kernelu a poté sečteny. Konvoluce je zobrazena na obr 2.14. Dopředný průchod sítí se skládá z 2 kroků. V prvním se vypočítá průběžná hodnota Z konvolucí předchozí vrstvy s tensorem W (3D filtr) a přičtením biasu. Ve druhém kroku se na průběžnou hodnotu Z aplikuje nelineární aktivační funkce g. Stejně jako běžná neuronová síť se konvoluční síť učí iterativními změnami vah a biasů.



Obrázek 2.14: Konvoluce obrazu [2].

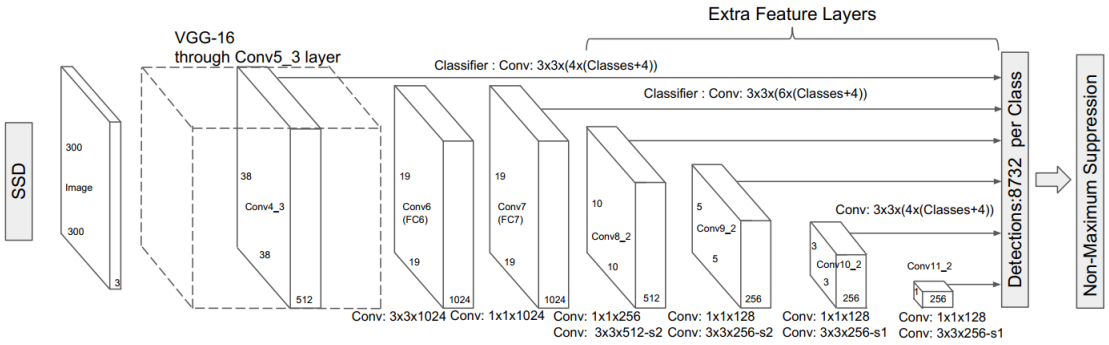
Pooling vrstva slouží k zmenšení rozměrů předchozí vrstvy a tím zrychlení výpočtu. Při poolingu se více hodnot zkombinuje do jedné. Používají se 2 typy poolingu: Max pooling, kde je nová hodnota maximem hodnot části předchozí vrstvy pokryté kernelem nebo Average pooling, kde je nová hodnota průměrem hodnot části předchozí vrstvy pokryté kernelem. Ilustrace na obr 2.15.



Obrázek 2.15: Pooling [4].

2.6.1 Single Shot MultiBox Detector

Liu et al. navrhli konvoluční neuronovou síť Single Shot Multibox Detector (SSD) [35], jejímž výstupem je kolekce bounding boxů a pravděpodobností s jakou se v bounding boxech vyskytuje objekt určité třídy. Metoda patří mezi tzv. Single shot detektory, kdy je objekt detekován přímo z feature mapy, na rozdíl od metod založených na návrhu regionů. Autoři se inspirovali algoritmem YOLO [41], který se také řadí mezi Single shot detektory a díky mnoha zlepšením dosáhli vyšší přesnosti.

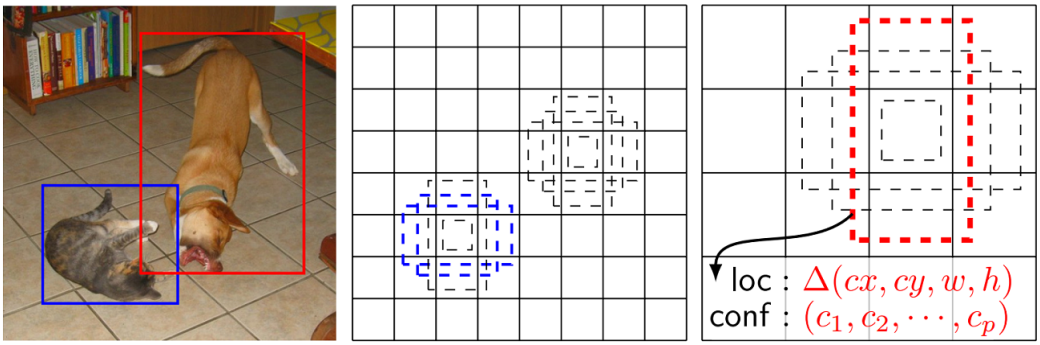


Obrázek 2.16: Model sítě Single Shot MultiBox Detector [35].

Síť v počátečních vrstvách obsahuje tzv. feature extraktor, tedy síť pro klasifikaci obrazu. Tato síť je zkrácena před klasifikačními vrstvami a za ni jsou přidány nové vrstvy. V originální implementaci byla použita síť VGG-16 [48], ale je možné použít jiné sítě pro klasifikaci obrazu. Nové přidané konvoluční vrstvy vytváří feature mapy se zmenšující se velikostí. Feature mapa je obraz vytvořený průchodem vrstvou. Z jednotlivých feature map v různých velikostech jsou predikovány detekce pomocí konvolučních filtrů. Konvoluční filtr má velikost 3×3 pro feature mapu o velikosti $m \times n$ a vytváří pravděpodobnost pro třídu objektů nebo ofset tvaru vůči souřadnicím výchozího bounding boxu. Na obr 2.16 jsou konvoluční filtry pro predikci detekcí vyznačeny šipkami z feature map.

Pro každou pozici na každé feature mapě je přiřazena množina výchozích bounding boxů. Na každé pozici je vypočítána pravděpodobnost výskytu určité třídy objektů ve výchozím bounding boxu spolu s ofsety vůči tvarům výchozích bounding boxů. Konkrétně je na každé pozici pro k výchozích bounding boxů vypočítána pravděpodobnost pro třídu objektu c a 4 ofsety. Celkem je pro $m \times n$ feature mapu použito $(c+4)kmn$ filtrů. Na obr 2.17

jsou zobrazeny výchozí bounding boxy na feature mapách s různou velikostí. Z predikovaných detekcí jsou poté v posledním kroku vybrány konečné detekce.



Obrázek 2.17: Ilustrace konvolučních filtrů pro predikci detekcí [35].

Při trénování je nutné přiřadit ground-truth bounding boxy k určitým výstupům detektoru. Poté je loss funkce a zpětná propagace aplikována tzv. end-to-end. Z možných výchozích bounding boxů, které mají Intersection over Union s ground-truth boxem větší než $0,5$ je náhodně vybrán 1. Pro dobré výsledky je pro Single Shot MultiBox Detector klíčová augmentace trénovacích dat. Díky tomu je dosaženo větší generalizace a model je schopný lépe detekovat různé velikosti a tvary objektů.

V porovnání s ostatními hlavními algoritmy založenými na konvolučních neuronových sítích je SSD méně přesný, ale výrazně rychlejší než Faster-RCNN [43]. YOLO je výrazně rychlejší, ale méně přesný detektor [24]. Tyto závěry jsou obecné, protože jsou vydávány nové verze detektorů, záleží také na použitém datasetu pro trénink, optimalizaci modelu, vstupním rozlišení obrazu a dalších detailech. SSD podobně jako YOLO hůře detekuje malé objekty. Jedním ze způsobů jak tento problém řešit je zvýšit rozlišení obrazu, což ale vždy znamená pomalejší detekci.

2.6.2 MobileNet

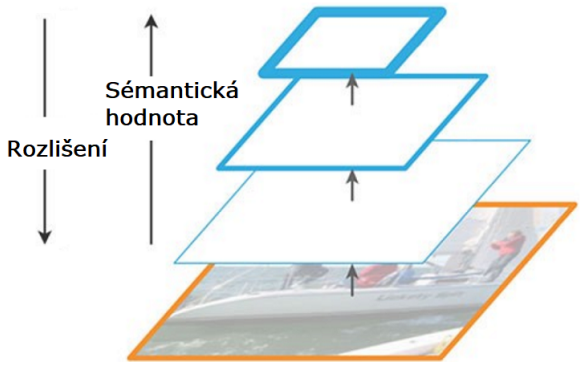
Ačkoli konvoluční neuronové sítě jsou známy dlouhou dobu, masivního rozšíření se dočkaly až v posledních letech. Průkopnickou prací byla AlexNet [31], která v roce 2012 vyhrála soutěž ImageNet Challenge: ILSVRC 2012 a ukázala obrovský potenciál konvolučních neuronových sítí pro počítačové vidění. To, co bylo dříve považováno za nemožné, se najednou stalo možným a konvoluční sítě začaly pronikat do všech oblastní počítačového vidění a nejen tam. Umožnil to také zvyšující se výpočetní výkon hardwaru, zejména grafických karet. Byly publikovány další práce, které přinesly sítě s výrazně vyšší přesností a rychlostí, důležití zástupci jsou VGG-16 [48] a ResNet [25], které ovlivnily další práce. Jsou publikovány nové práce, kde se autoři snaží zlepšovat výsledky sítí různými modifikacemi architektury sítí. Trendem je použití hlubších sítí, větší hloubka jednotlivých vrstev, použití menších kernelů atd.

V poslední době se pozornost výzkumníků obrátila na větší efektivnost sítí, kdy optimalizují sítě tak, aby dosahovaly přibližně stejných výsledků s menšími nároky výpočetní výkon a menší velikostí sítí. Sítě jsou tak rychlejší nejen při průchodu sítí ale i při tréninku a zároveň zabírají méně místa v paměti. Jednou z takových prací je MobileNet [27], kde autoři prezentují síť, která je výrazně optimalizovaná ve snaze umožnit použití sítě na mobilních zařízeních a vestavěných systémech.

Model je efektivní díky použití tzv. hloubkově oddělitelných konvolucí (*depthwise separable convolutions*) namísto standardních konvolucí. Standardní konvoluce filtruje a kombinuje vstupy do nových výstupů v jednom kroku. Při hloubkově oddělitelné konvoluci je filtrování a kombinování odděleno do zvláštních vrstev. Nejdříve hloubková konvoluce aplikuje 1 filtr na každý kanál (rozměr hloubky – např. vstupní RGB obraz má 3 kanály) a poté bodová (pointwise) konvoluce aplikuje 1×1 konvoluci a zkombinuje tak výstupy hloubkové konvoluce. Výsledkem je rapidní snížení počtu operací a velikosti modelu. Oproti stejnému modelu obsahujícím standardní konvoluci dosáhli autoři práce pouze o $1,1\,\%$ horší přesnosti na datasetu ImageNet pro klasifikaci obrazů, zatímco snížili počet Mult-Adds (násobení a sčítání) operací o $88,3\,\%$ a počet parametrů o $85,6\,\%$.

2.6.3 Feature Pyramid Network

Detekce objektů v různých měřítcích je obtížné zejména pro malé objekty. Feature mapy v konvolučních neuronových sítích se přirozeně zmenšují s počtem vrstev a síť tak nabývá tvaru pyramidy. Feature mapy s vysokým rozlišením na začátku sítě mají nízko úrovňové rysy (featury), zatímco feauture mapy s nízkým rozlišením mají vysoko úrovňové rysy, sémantická hodnota každé další feature mapy se tak zvyšuje, jak je vidět na obr 2.18. Algoritmus Single Shot Multibox Detector (SSD) [35] používá k detekci několik feature map, ale nepoužívá spodní vrstvy. To vede k jeho horší výsledkům při detekci malých objektů [28].

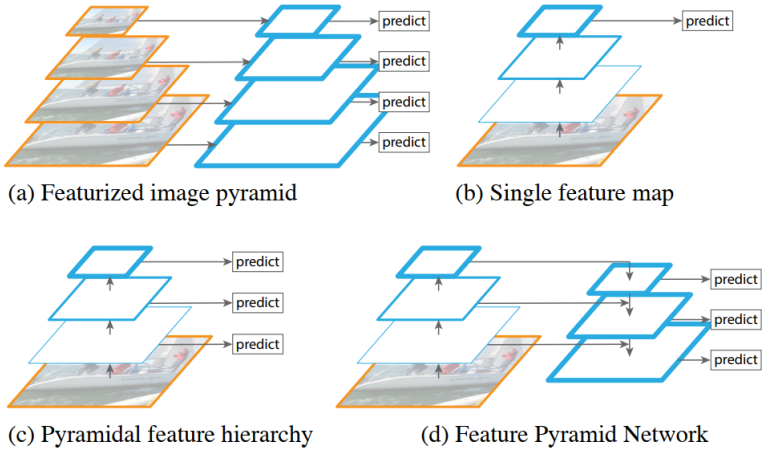


Obrázek 2.18: Hierarchie feature map. S dalšími vrstvami se zvyšuje sémantická hodnota a zároveň klesá rozlišení [28] (upraveno).

Řešením problému detekce objektů v různých měřítcích je použití obrazu v několika měřítcích a pro každý obraz vytvořit feature mapu. Je tak vytvořena tzv. Featurized image pyramid, kde jsou predikovány detekce z několika feature map 2.19(a). Tento přístup je pomalý, protože pro každá feature mapa je vypočítána zvlášť. Problém je i s paměťovými nároky při trénování. Proto běžné algoritmy predikují detekce pouze z jedné feature mapy za cenu horších výsledků 2.19(b).

Jak již bylo uvedeno výše, algoritmus SSD využívá přirozené pyramidální hierarchie k detekci objektů z několika feature map v různém rozlišení 2.19(c). Jednotlivé feature mapy jsou na sobě nezávislé a spodní feature mapy s vysokým rozlišením nejsou použity. Odpovědí na problémy tohoto a výše uvedených řešení je Feature Pyramid Network (FPN) [34]. Díky propojením feature map shora dolů a postranními propojeními mezi feature mapami na stejné úrovni je dosaženo vysoko úrovňové sémantické hodnoty na všech feature mapách.

Feature pyramida je složena z cesty zespoda nahoru, z cesty shora dolů a postranních propojení. Cesta zespoda nahoru je dopředný výpočet sítě stejně jako u každé konvoluční neuronové sítě. V konvoluční síti je často několik vrstev, jejichž výstupem jsou feature mapy o stejném rozlišení. Takto po sobě jdoucí vrstvy jsou nazvány *stupněm*. Výstup jednoho stupně, tedy výstup poslední vrstvy ve stupni je referenční feature mapa. Rozlišení feature mapy se po každém stupni zmenšuje na polovinu. Cesta shora dolů prokresluje z vyšší vrstvy do nižší sémanticky hodnotnější vysoko úrovňové rysy. Vyšší vrstva je dvakrát zvětšena a poté je spojena s feature mapou se stejným rozlišením z cesty zespoda nahoru pomocí postranního propojení. Tento krok pomáhá lépe predikovat pozici objektu. Celé schéma je zobrazeno na obr 2.19(d). Zakomponování FPN do detekční konvoluční sítě významně zlepšuje jak Precision, tak Recall. Zejména malé objekty jsou detekovány lépe.

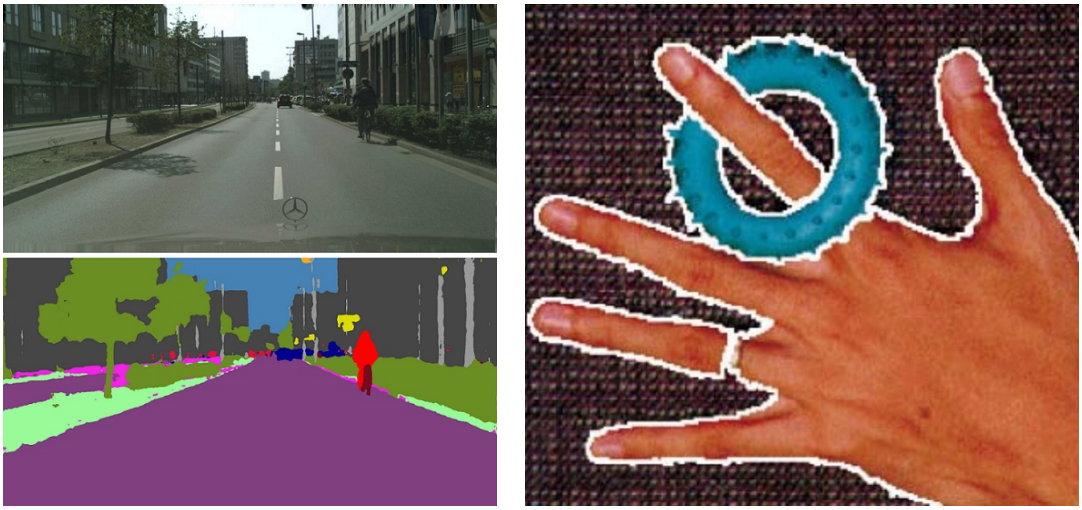


Obrázek 2.19: (a) Použití obrazu v několika měřítcích pro vytvoření feature pyramidy. Pro každý obraz probíhá zvlášť výpočet, což je v součtu pomalé. (b) Použití pouze jedné feature map pro dosažení rychlejší detekce. (c) Použití přirozené hierarchie konvolučních sítí pro detekci v několika vrstvách. (d) Feature Pyramid Network, kde je vytvořena feature pyramida. Díky propojení mají všechny feature mapy větší sémantickou hodnotu [34].

2.7 Segmentace

Segmentace je proces rozdělení obrazu do několika segmentů (superpixelů). Cílem je zjednodušit nebo změnit reprezentaci obrazu pro snazší analýzu [47]. Pixely patřící jednotlivým segmentům spolu sdílí určité vlastnosti jako barva, jas nebo textura. Segmentaci lze také popsat jako proces přiřazení labelu (štítku) každému pixelu v obraze. Aplikací segmentace je velmi mnoho: detekce objektů, vyhledávání obrazů, detekce a rozpoznání tváře, rozpoznávání otisků prstů, dohledové systémy, strojové vidění, atd.

Metody segmentace se dělí na dvě hlavní skupiny – segmentace s dohledem (*supervised*) a segmentace bez dohledu (*unsupervised*). Metody segmentace s dohledem mohou na základě předchozí informace dosáhnout sémantické segmentace – tedy segmentace konkrétního objektu. Předchozí informace (object prior) je získána metodami jako např. detekce významného objektu (*saliency object detection*), metody strojového učení (AdaBoost, SVM, CNN) a metody s lidskou interakcí. Naopak metody segmentace bez dohledu segmentují obraz na mnoho jednotných a homogenních oblastí na základě společných vlastností jako je barva nebo textura [37]. Rozdíl mezi oběma skupinami lze vidět na obr 2.7.



Obrázek 2.20: Příklad segmentace obrazů. Vlevo segmentace s dohledem, vpravo segmentace bez dohledu [51].

Mezi segmentační metody patří metody shlukování (Mean-Shift), metody založené na růstu regionů, metody založené na konvolučních neuronových sítích, metody aktivních obrysů (Active Contours) založené na hledání hran (známé jako Snakes) nebo na Level set funkci a metody založené na Markovských náhodných polích. Metoda aktivních obrysů s Level set funkcí a metody založené na Markovských náhodných polích jsou popsány v rámci podkapitoly Kosegmentace 2.8.

Podkategorií segmentace je segmentace video objektů, která je definována jako segmentace objektu napříč rámci video sekvence. Na rozdíl od segmentace jednoho obrazu zde lze využít pohybu objektu mezi rámci pro kvalitnější výsledky. Takové metody jsou označovány jako hybridní metody, příkladem je [53], kde je využit optický tok.

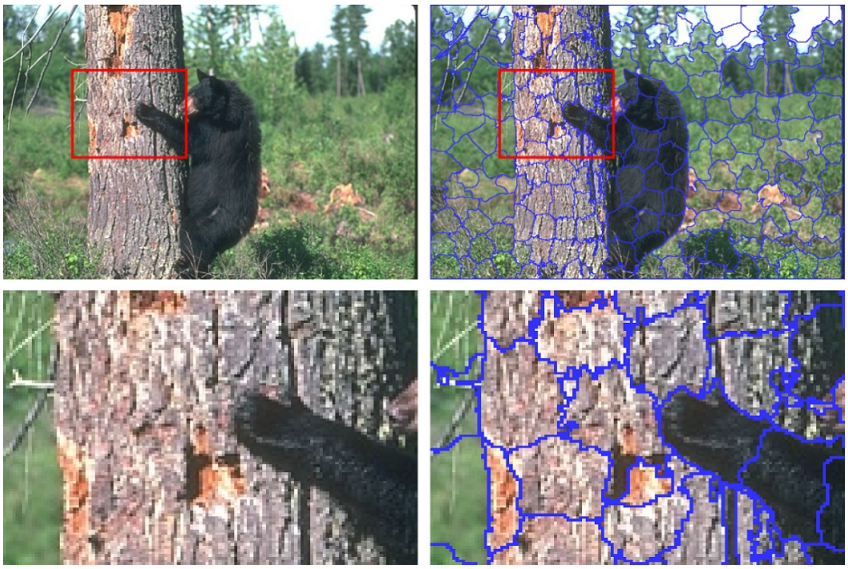
2.7.1 Sémantická segmentace založená na konvolučních neuronových sítích

V posledních letech se pro sémantickou segmentaci začali široce používat konvoluční neuronové sítě (CNN), kvůli jejich schopnosti extrahovat kompaktnější a smysluplnější rysy z obrazu než klasické algoritmy (např. HOG). Ukazuje se, že CNN dosahují jasně lepších výsledků než starší přístupy [20].

Jedním z přístupů jak dosáhnout lepších výsledků je použití pravděpodobnostních modelů jako Markovská náhodná pole a Conditional Random Fields (CRF) [20]. CRF jsou variantou Markovských náhodných polí. Příkladem kdy je k poslední vrstvě připojeno CRF je DeepLab [11]. Vysoká přesnost se obvykle pojí s delším časem nutným pro segmentaci. Zhao *et al.* [57] proto navrhli Image Cascade network (ICNet), která dosahuje 30 FPS na obrazech s rozlišením 1024×2048 a zároveň vysoké přesnosti. Principem sítě je použití kaskády vstupů, kde je vstupem trojice obrazů v nízkém, středním a vysokém rozlišení. Využívá se tak efektivně sémantické informace v nízkém rozlišení spolu s detaily z vysokého rozlišení.

2.7.2 Texture-Aware Superpixel Segmentation

Jedna z posledních prací zaměřující se na segmentaci bez dohledu je Texture-Aware Superpixel Segmentation [21]. Giraud *et al.* vytvořili metodu, která segmentuje obraz nejen na základě barvy, ale bere v potaz i texturu. Metoda je vylepšením metody SLIC [7] a je založená na k-means shlukování. K výrazu pro prostorovou vzdálenost a výrazu pro barvu přidává výraz pro průměrnou vzdálenost pixelů s podobnou texturou okolí pixelu. Ve výsledku je jeden shluk segmentem (superpixelem) v obraze. Výhodami této metody jsou univerzálnost, nízká výpočetní náročnost a malý počet parametrů.



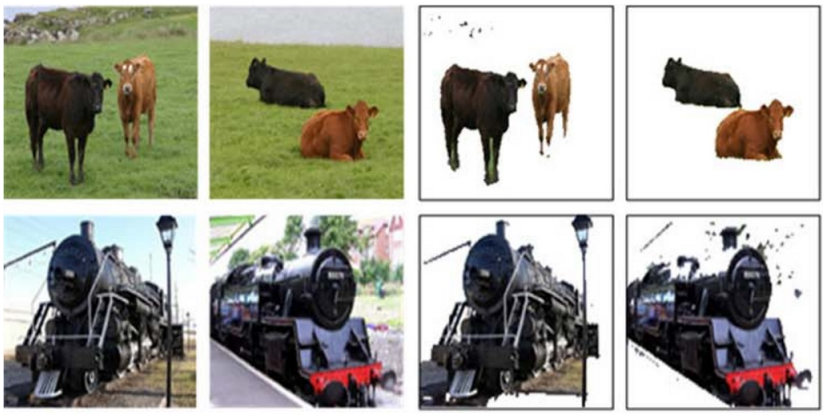
Obrázek 2.21: Výsledky metody Texture-Aware Superpixel Segmentation na vysoce texturovaných obrazech [21]. Vlevo vstupní obraz, vpravo výsledná segmentace.

2.8 Kosegmentace

Kosegmentace (*cosegmentation*) je speciální případ segmentace (2.7), kdy je současně segmentována společná část dvou a více obrazů [9][37][52]. Metoda segmentuje objekty z vícera obrazů extrahováním společných objektů mezi obrazy [37], viz obr 2.8. Kosegmentaci lze použít pro detekci objektů, jako v této práci. Techniky kosegmentace lze rozdělit na techniky založené na Markovských náhodných polích (MRF) a ostatní. Mezi ně patří metody založené na shlukování [30] a také Active contour model. Kosegmentace má mnoho potencionálních aplikací jako klasifikace obrazů, rozpoznávání objektů a vyhledávání obrazů.

Metoda příbuzná kosegmentaci je kosegmentace video objektů (*video object cosegmentation*), která separuje objekty od pozadí ve videu a to buď interaktivně, nebo automaticky [55]. Tato metoda může fungovat s jedním nebo více videi. Příklady prací jsou [55], kde je použit prostorově-časový auto-kontext model kombinovaný s modelováním vzhledu pro označování superpixelů (segmentů) a [54], kde jsou extrahovány objekty, poté sledovány napříč videi a kosegmentovány s využitím algoritmů SVM a CNN. Metody popsané v obou pracích jsou hybridními metodami.

Kosegmentaci podobná metoda je kolokalizace, která se zaměřuje na nalezení bounding boxu ohraničujícího společné objekty ve více obrazech [52].



Obrázek 2.22: Příklad kosegmentace. Vlevo původní dvojice obrazů, vpravo výsledky metody [37].

2.8.1 Metody založené na Markovských náhodných polích

Markovská náhodná pole (*Markov Random Field* – MRF) se používají nejen pro kosegmentaci, ale i pro mnoho dalších úloh v oblastech počítačového vidění a počítačové grafiky jako segmentace, syntéza textur, registrace obrazu a komprese obrazu. Jedná se o pravděpodobnostní model, který je generalizací Markovských procesů [32].

Markovské náhodné pole

Markovské náhodné pole nebo také Markovská síť je definováno jako graf $G = (V, E)$. $V = \{1, 2, \ldots, N\}$ je množina uzlů, každý z nich je spojen s náhodnou proměnnou u_j, $j = 1 \ldots N$. E značí množinu hran mezi uzly. Okolí uzlu i, značeno $\mathcal{N}_i$ je množina uzlů přilehlých k uzlu i, tedy $j \in \mathcal{N}_i$ pokud a pouze když $(i, j) \in E$. Graf G je označován jako Markovské náhodné pole pokud a pouze když jsou splněny následující podmínky:

$$p(u) > 0, \forall u \ \in U \tag{2.12}$$

kde $p(u) = p(U = u)$ je vícerozměrná pravděpodobnost, U je skupina náhodných proměnných definovaných v rámci náhodného pole. Platí, že všechny pravděpodobnosti jsou kladné, jedná se tedy o podmínku Pozitivity.

Druhá je podmínka Markovianity:

$$p(u_i \mid \{u_j\}_{j \in V \setminus i}) = p(u_i \mid \{u_j\}_{j \in \mathcal{N}_i}) \tag{2.13}$$

která popisuje, že náhodná proměnná u_i závisí jen na sousedních uzlech [32][29].

Labeling problém

Kosegmentace i segmentace je zde definována jako labeling problém, kdy je každému pixelu přiřazen label (štítek) [50]. Jednotlivé pixely, segmenty nebo rohy představují *místa*, množina míst S je:

$$S = \{1, \ldots, N\}. \tag{2.14}$$

Každý obraz má množinu labelů L:

$$L = \{1, \ldots, M\}. \tag{2.15}$$

Label může popisovat intenzitu pixelu, objekt nebo hranu. V případě kosegmentace i segmentace je množina L binární:

$$L = \{0, 1\}. \tag{2.16}$$

Kde 1 znamená, že pixel náleží objektu a 0 nenáleží objektu.

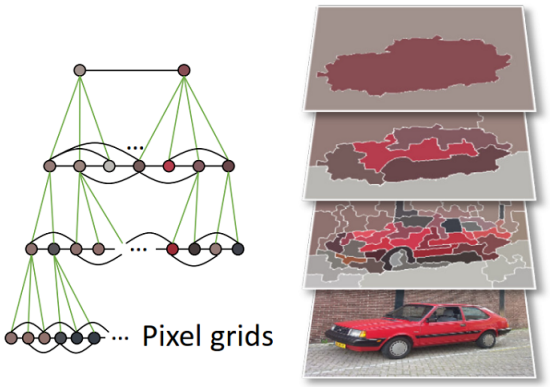
Na problém přiřazení labelu každému místu lze pohlížet jako na hledání zobrazení z S do L:

$$f : S \to L. \tag{2.17}$$

Model a optimalizace

Pro kosegmentaci je nutné zvolit vhodný model a jeho energetickou funkci. Příklad modelu je na obr 2.23, kde je vidět hierarchický MRF model. Minimalizací energetické funkce hledáme nejoptimálnější konfiguraci pole. Mezi optimalizační metody patří graph cut, simulované žíhání nebo belief propagation.

V práci [52] je použit hierarchický MRF model pro kosegmentaci a dense correspondence (hluboká shoda – podobnost obrazů) s využitím algoritmů HOG pro extrakci rysů a graph cut pro optimalizaci energetické funkce. V jedné z posledních prací zabývajících se kosegmentací [36] je použit dynamický MRF model, autoři se zaměřují na detekci a segmentaci objektů v zašuměných videích.



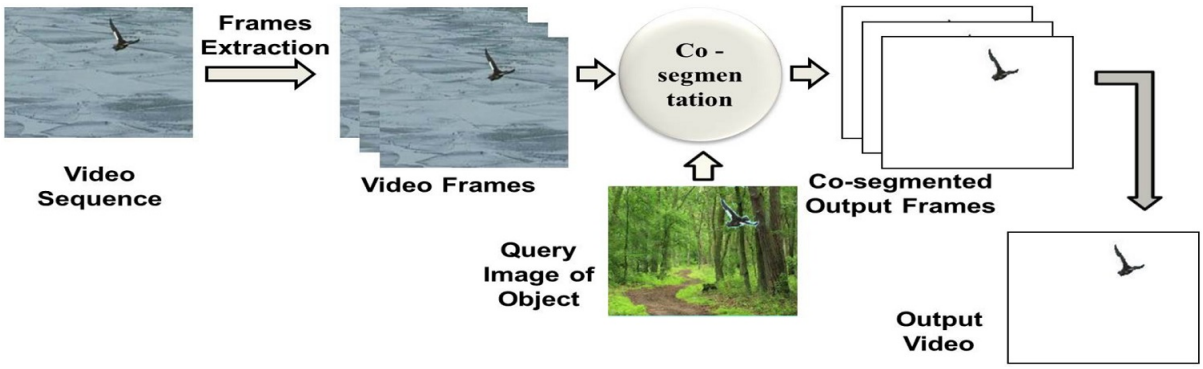
Obrázek 2.23: Ilustrace MRF modelu nad obrazem, nejspodnější vrstva grafu reprezentuje jednotlivé pixely, další vrstvy reprezentují segmenty, které postupně splývají v jeden segment - segmentovaný objekt [52].

2.8.2 Color reward strategy and Active Contour Model

Meng *et al.* [37] navrhli kosegmentační model zahrnující strategii odměňování barev a aktivní obrysový model. Metoda pracuje na principu separace popředí od pozadí aktivním obrysem (křivkou). Je vytvořena energetická funkce se dvěma protichůdnými cíli. Jeden je podobnost popředí mezi obrazy, druhý je konzistence pozadí v každém obraze. Pro reprezentaci regionů je použit barevný histogram a pro měření podobnosti popředí a konzistence

pozadí je použita strategie odměn. Energetická funkce je formulována jako level set funkce a iterativně minimalizována.

Aktivní obrysový model je založen na matematickém základě popsaném v mnohokrát citované práci dvojice Chan a Vese [17]. Bandyopadhyay [9] rozšířil uvedenou metodu na detekci objektů ve videu pomocí vyhledávacího obrazu. Princip je následující: vyhledávací obraz obsahuje hledaný objekt (y), video sekvence je rozdělena na jednotlivé rámce a nad každým rámcem je provedena kosegmentace spolu s vyhledávacím obrazem. Výsledkem je detekovaný objekt (y) v každém rámci videa. Výhodou použití kosegmentace pro detekci objektů ve videu je to, že nepotřebuje žádnou předchozí informaci ani lidský dohled.



Obrázek 2.24: Schéma práce navržené Bandyopadhyayem pro detekci objektů ve videu pomocí kosegmentace [9].

Energetická funkce

Energetická funkce pro kosegmentaci založenou na aktivním obrysovém modelu je dána:

$$\begin{aligned} E(C_k) = \mu \cdot Length(C_k) + \nu \cdot Area(C_k^i) \\ - \lambda_k^i \int_{C_k^i} f\left[I_k(x,y),\, r(C_{1-k}^i)\right] \mathrm{d}x\mathrm{d}y \\ - \lambda_k^o \int_{C_k^o} f\left[I_k(x,y),\, r(C_k^o)\right] \mathrm{d}x\mathrm{d}y \end{aligned} \tag{2.18}$$

kde $I_k,\ k \in \{0,1\}$, je $k$tý obraz páru obrazů, C_k značí křivku v obraze I_k, regiony uvnitř křivky jsou značeny $r(C_{1-k}^i)$ a regiony vně křivky jsou reprezentovány $r(C_k^o)$. V případě, že pracujeme s *Obrazem 0*, tj. $k = 0$, tak regiony uvnitř křivky jsou $r(C_1^i)$ a regiony vně křivky $r(C_0^o)$. Výraz $f[I_k(x,y), r(C_{1-k}^i)]$ je použit k měření podobnosti mezi pixely v popředí obrazu I_k a regiony uvnitř křivky druhého obrazu $r(C_{1-k}^i)$. Konzistence pozadí je reprezentována výrazem $f[I_k(x,y),\ r(C_k^o)]$, který měří podobnost mezi pixely pozadí obrazu I_k a vnějších regionů $r(C_k^o)$.

První dva výrazy na pravé straně 2.18 popisují délku křivky a plochu regionů uvnitř křivky. Výrazy $\mu > 0, \nu > 0, \lambda_k^i > 0$ a $\lambda_k^o > 0$ jsou parametry modelu, kde μ je váha pro délku křivky, ν je váha pro plochu regionů uvnitř křivky, váha $\lambda_k^i > 0$ udržuje jednotnost popředí a $\lambda_k^o > 0$ jednotnost pozadí obrazu.

Reprezentace regionů

Regiony jsou reprezentovány normalizovaným barevným histogramem h. Hodnota barvy v RGB obraze je reprezentována 3D vektorem. Histogram je vytvořen s ohledem na pravděpodobnost každé barvy v regionu.

Míra podobnosti

Míra podobnosti f pro hodnotu pixelu p je $f(p,\ r(C_k)) = h(p)$. Pokud je hodnota pixelu v rozsahu s malou četností hodnot, tj. hodnota má malou pravděpodobnost, míra podobnosti bude malá. Naopak hodnota pixelu v rozsahu s velkou četností vrátí velkou míru podobnosti. Obyčejně se jednotlivé hodnoty pixelu (barvy) od sebe příliš neliší. Pokud tedy je vzdálenost barev dvou pixelů menší než β, tak budou mít pixely stejnou barvu. Pro hodnotu pixelu p je potom míra podobnosti dána:

$$f\ (p,\ r(C_k)) = \sum_{|p'-p|\leq\beta} h(p') \tag{2.19}$$

kde $|p' - p||$ značí největší vzdálenost ze všech vzdáleností mezi hodnotami barevných kanálů RGB.

Formulace level set funkce

Energetická funkce je minimalizována řešením parciální diferenciální rovnice (PDE) level set formulace 2.18. Tedy celková rovnice je změněna na:

$$\begin{aligned}\Phi_k^{n+1}(x,y) = \Phi_k^n(x,y) + \delta(\Phi_k^n(x,y)) \cdot \{\mu \cdot \mathrm{div}(\frac{\nabla(\Phi(x,y))}{|\nabla(\Phi(x,y))|}) - \nu \\ +\lambda_{\Phi k}^i \cdot f\ [I_k(x,y),\ r_\Phi(C_{1-k}^i)] - \lambda_{\Phi k}^o \cdot f\ [I_k(x,y),\ r_\Phi(C_k^o)]\}\end{aligned} \tag{2.20}$$

kde Φ značí level set energetickou funkci křivky. Hodnota level set energetické funkce pro pixel (x, y) v n-té iteraci je reprezentována výrazem $\Phi_k^n(x, y)$. Výraz $\delta(\Phi)$ je Diracova míra definována vzorcem:

$$\delta(\Phi) = \frac{\mathrm{d}(H(\Phi))}{\mathrm{d}\Phi} \tag{2.21}$$

kde $H(\Phi)$ je Heavisideova funkce křivky Φ.

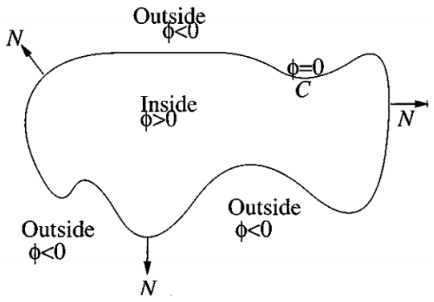
Heavisideova funkce nebo také jednotkový skok je definována jako:

$$H(x) = \begin{cases} 0 & \text{pro } x < 0 \\ 1 & \text{pro } x \geq 0 \end{cases} \tag{2.22}$$

Inicializovaný obrys Φ_k^0 se minimalizací energetické funkce 2.20 začne smršťovat směrem k hranícím objektu (ů). Po určitém počtu iterací se obrys zastaví na hranici, kde je hodnota energetické funkce minimální. Tato hranice separuje společný objekt (y) v obou obrazech a region definovaný křivkou nacházejícím se v rámci videa je považován za hledaný objekt.

Separaci objektu (ů) ilustruje obr 2.25. Z něj je patrné, že křivka C se nachází tam, kde $\Phi = 0$ a dále že pixely s $\Phi > 0$ leží v regionech uvnitř křivky a náleží tedy popředí, zatímco pixely s $\Phi < 0$ leží v regionech vně křivky a náleží pozadí.

Postupný vývoj křivky je zachycen na obr 2.26, kde u dvou párů obrazů je zobrazeno pět stavů vývoje křivky odpovídajících iteracím 1, 101, 201, 301 a 401.



Obrázek 2.25: Separace popředí a pozadí pomocí křivky C [17].

Meng *et al.* [37] použil ve svých experimentech tyto parametry: $\mu = 0,01, \lambda_k^i = \lambda_k^o = 1, \beta = 15$ a $\nu = 0,001$. Velikost parametrů mu a ν ovlivňuje výslednou masku, při malých hodnotách se mohou vyskytnout redundantní regiony, naopak velké hodnoty mohou odstranit redundantní regiony, zároveň ale i části společných objektů. Počet iterací nastavili na $N_i = 1500$. Počáteční křivka Φ_k^0 je definována jako čtverec, jehož strany jsou blízko okrajům obrazu ve vzdálenosti $\gamma = 5$ pixelů, aby pokrýval většinu plochy společných objektů.

Objekt (y) definovaný vyhledávacím obrazem je detekován provedením kosegmentace nad každým extrahovaným rámcem videa. Schéma metody je zobrazeno na obr 2.24. Algoritmus metody je popsán v 1. Protože hodnota energetické funkce je počítána pro každý pixel, je počet iterací je spojen s počtem pixelů obrazu n. Výsledná časová složitost metody je úměrná k $O(n^2)$ [37].

Obrázek 2.26: Vývoj křivky, zde pět stavů odpovídajícím iteracím 1, 101, 201, 301 a 401 [37].

Algorithm 1: Metoda pro detekci objektů založená na vyhledávacím obraze

Input: Vyhledávací obraz **and** Video sekvence

1: **Initialization:** $\Phi_k^0 = 0,\ k = 0,\ 1,\ n = 0$ a ostatní parametry
2: $temp\_\Phi_k^n \leftarrow \Phi_k^0$
3: Etrahuj video rámec $F_j \in \{f_1, f_2, \ldots f_t\}$
4: **for** $j \leq t$ **do**
5: **for** $n \leq maxiteration$ **do**
6: Vypočti $r_\Phi(C_{1-k}^i)$ a $r_\Phi(C_k^o)$ pro každý obraz I_k
7: Vypočti Φ_k^{n+1} řešením PDE v $temp\_\Phi_k^n$ za použití rovnice 2.20
8: **if** $(\Phi_k^{n+1} = temp\_\Phi_k^n) \parallel (n = maxiteration)$ **then**
9: **goto** *Krok 16.*
10: **end**
11: **else**
12: $temp\_\Phi_k^n \leftarrow \Phi_k^{n+1}$
13: **end**
14: **end for**
15: **end for**
16: **Extrahuj společný objekt (y):** Pixely s hodnotou $\Phi_k^{n+1} = \Phi_k^n > 0$ náleží hledanému objektu (ům) pro F_j a Vyhledávací obraz.
17: **Reinicializace:** $\Phi_k^0 = 0,\ k = 0,\ 1,\ n = 0$ a ostatní parametry
18: $temp\_\Phi_k^n \leftarrow \Phi_k^0$

Output: Video sekvence s detekovaným objektem (y).

Kapitola 3

Návrh systému

Cílem této práce je navrhnout, implementovat a otestovat systém pro detekci a lokalizaci dronu či hejna dronů. V této kapitole bude popsán návrh systému. Popis implementace a testování systému se nachází v následujících kapitolách. Detekční algoritmus by se měl co nejlépe vypořádat se všemi výzvami a problémy spojenými s detekcí létajících objektů. Hlavními výzvami je zcela nekontrolované prostředí, tj. vliv počasí, mnoho nezávislých pohybů, změna osvětlení scény, komplexní pozadí, atd., dále velká variabilita vzhledu dronů, nutnost detekce dronu ve velké vzdálenosti a tím pádem velmi malá velikost dronu v obraze, vysoká přesnost nutná pro reakci na přítomnost dronu (např. sestřelení), vysoká rychlost detekce řádově v desítkách až stovkách milisekund, vysoká rychlost pohybu dronů, video zachycené pohybující se kamerou a další problémy popsané v podkapitole 2.4.

Požadavky na systém jsou: systém bude pracovat pokud možno v reálném čase, vstupem systému bude video zachycené pohybující se pozemní PTZ kamerou nebo video výstup z pohybující se kamery, detekce dronu či hejna dronů bude co nejpřesnější a systém se co nejvíce vypořádá s výzvami a problémy uvedenými v předchozím odstavci. Výstupem systému bude video sekvence, kde v případě přítomnosti jednoho či více dronů budou tyto drony vyznačeny bounding boxem (co nejmenším obdélníkem ohraničujícím objekt).

3.1 Použitý software

Systém je navržen jako desktopová aplikace naprogramovaná v jazyce C++. Pro práci s obrazem bude využita knihovna OpenCV[1] (verze 4.1). Knihovna OpenCV je open source knihovna obsahující algoritmy pro zpracování obrazu a počítačové vidění. Algoritmus pro detekci bude založený na konvoluční neuronové síti. CNN model bude retrénován na Google Cloud ML Engine[2] pomocí platformy TensorFlow[3] (verze 1.13.1), protože má širokou komunitu uživatelů, pokročilé nástroje a je k dispozici dostatek materiálů. Konkrétně pro detekci objektů bude využito TensorFlow Object Detection API[4]. Pro anotaci datasetu bude použit nástroj LabelImg[5] (verze 1.8.1). Pro uživatelské rozhraní bude pro svoji univerzálnost a jednoduchost tvorby použita knihovna Qt[6] (verze 5.12.2). Akceleraci na výkonné grafické

[1] https://opencv.org/
[2] https://cloud.google.com/ml-engine/
[3] https://www.tensorflow.org/
[4] https://github.com/tensorflow/models/tree/master/research/object_detection
[5] https://github.com/tzutalin/labelImg
[6] https://www.qt.io/

kartě umožní platforma NVIDIA CUDA[7] (verze 10.0). Vývoj programu bude probíhat ve vývojovém prostředí Visual Studio 2017 (verze Community). Knihovny OpenCV a Qt jsou multiplatformní, aplikace tedy bude přeložitelná na různých systémech. V rámci této práce bude přeložena pouze pro operační systém Microsoft Windows 10. Dokumentace zdrojových souborů bude vygenerována programem Doxygen[8].

3.2 Návrh aplikace

Původní návrh počítal s použitím kosegmentace pro detekci objektů. Algoritmus je popsán v předchozí kapitole, konkrétně v práci pana Bandyopadhyaye [9]. Později jsem si uvědomil, že přes všechny výhody, které této algoritmus má, je pro detekci dronů navzdory názvu práce *Object of interest detection in video sequence using co-segmentation: A new era in video surveillance* naprosto nevhodný. Hlavním důvodem je extrémně vysoká výpočetní náročnost a tím i čas potřebný pro detekci dronu v jednom rámci videa. Algoritmus počítá energetickou funkci ve dvou obrazech pro každý pixel. I s nejmodernějším a nejrychlejším hardwarem a vysoce paralelizovanou a optimalizovanou implementací by výpočet jednoho rámce trval několik minut. Tento výrok potvrzuje navazující práce pana Soomra [49], kde jsou namísto pixelů použity superpixely a jejich počet je omezen na pouhých 1000. I přesto nejrychlejší výpočet trvá desítky sekund, což znemožňuje použití algoritmu pro detekci dronů, kde je vyžadován výpočet v reálném čase, případně v jednotkách sekund.

Taktéž nelze použít jinak běžně používané algoritmy pro detekci chodců nebo automobilů, protože nejsou uzpůsobeny pro detekci objektů ve videu zachyceném pohybující se kamerou a nevyhovují požadavkům na systém. Z metod založených na pohybu lze vyloučit použití metody odčítání pozadí, která je nevhodná pro video zachycené pohybující se kamerou [56][45]. Dále je možné vyloučit optický tok, který má problémy s rychlostí výpočtu, přesností, schopností detekce malých objektů, schopností rozlišit dron od jiných objektů a také schopností detekovat nepohybující se objekty [45]. Obecně nelze doporučit metody založené pouze na pohybu. Na druhou stranu jsou pro daný účel velmi vhodné hybridní metody pracující jak v prostorové tak časové doméně. Tradiční metody založené na vzhledu používající ručně vytvořené rysy jako Histogram of Oriented Gradients (HOG) [15] nebo Speeded Up Robust Features (SURF) [10] v kombinaci s klasifikačním algoritmem jako AdaBoost [19] či Support vector machines (SVM) [14] jsou pomalé a obecně dosahují horších výsledků než konvoluční neuronové sítě (CNN) [8]. CNN představují nejmodernější přístup nejen pro detekci objektů, ale pro všechny oblasti počítačového vidění a jsou předmětu aktivního výzkumu.

Z výše uvedených důvodů jsem návrh změnil. Namísto kosegmentace bude algoritmus pro detekci objektů založený na konvoluční neuronové síti, konkrétně na algoritmu Single Shot MultiBox Detector (SSD) [35]. Tento algoritmus je nejmodernějším a nejpokročilejším přístupem pro daný problém. V porovnání s alternativními typy CNN pro detekci objektů jako jsou Faster R-CNN [43] a YOLO [41][42] představuje nejlepší kompromis mezi přesností a rychlostí detekce. Faster R-CNN je přesnější, ale pomalejší než SSD, zatímco YOLO je rychlejší, ale méně přesná metoda než SSD [35][24]. Podrobnosti lze nalézt v podkapitole 2.6.

Pro natrénování CNN bude ručně vytvořen a anotován zcela nový dataset obsahující různorodé obrázky dronů. Zdrojem dat pro dataset budou videozáběry poskytnuté vedoucím práce a videa a obrázky stažené z internetu, převážně z webu YouTube. Na internetu se mi

[7] https://developer.nvidia.com/cuda-zone
[8] http://www.doxygen.nl/

i přes důkladné hledání nepodařilo najít jakýkoliv veřejně dostupný dataset obrázků dronů, rovněž zdroje na internetu byly překvapivě omezené. Rozantsev, Lepetit a Fua [45] vytvořili dataset dronů, ale obrazy jsou šedotónové a v nízkém rozlišení. Aker a Kalkan [8] rovněž vytvořili dataset, ale ten je vytvořen uměle a není veřejně dostupný. Vytvořený dataset tak bude pravděpodobně naprosto unikátní. Počítá se možností zveřejnění a zpřístupnění datasetu dalším výzkumníkům. Pro svou malou velikost v řádu tisíců příkladů bude vhodný jen pro retrénování předem natrénované sítě.

Vstupem aplikace bude video sekvence čtená ze souboru nebo video výstup z kamery připojené přes rozhraní USB. Poté budou z videa extrahovány jednotlivé rámce. Další možnou úpravou pro urychlení výpočtu je vynechávání rámců, kde bude zpracován každý n-tý rámec. Pomocí modulu dnn knihovny OpenCV bude realizován průchod sítí. Detekce bude probíhat na procesoru. Pro akceleraci detekce bude aplikace také umožňovat detekci na grafické kartě.

Výstupem aplikace bude video sekvence s drony vyznačenými bounding boxy zobrazovaná v okně aplikace. Aplikace také bude umožňovat uložit výstup do souboru.

3.3 Uživatelské rozhraní

Okno aplikace bude rozděleno na dvě hlavní části: na levé straně bude video sekvence a na pravé ovládací prvky. Aplikace bude nabízet možnost výběru vstupu, výběr složky obsahující CNN model, spuštění nahrávání a cestu k souboru, výběr zařízení pro detekci (procesor nebo grafická karta), dobu detekce v jednom rámci a spuštění/pozastavení/zastavení detekce. Dále bude nabízet možnost vynechání rámců a tím plynulejší činnost detektoru. Stav aplikace bude vypsán v logu.

Kapitola 4

Implementace systému

V této kapitole je popsána implementace systému navrženého v předchozí kapitole. Implementace se skládá z více částí: výběr, instalace a konfigurace potřebných nástrojů; získání, výběr vhodných dat a následné vytvoření datasetu; natrénování modelu konvoluční neuronové sítě a naprogramování demonstrační aplikace. Jednotlivé části jsou detailně popsány níže.

4.1 Použité nástroje

Pro následující části implementace bylo zapotřebí vybrat vhodné nástroje, programy a knihovny. Poté je nainstalovat a nakonfigurovat. Jakkoli se tento úkol může zdát snadný, při jeho plnění jsem se setkal s mnoha problémy. Ať už z důvodu nesprávných a navzájem nekompatibilních verzí nástrojů, chybné konfigurace, nedostatečné nebo špatné dokumentace či chyb v samotném softwaru třetích stran. Chyby v softwaru třetích stran a nedostatečná dokumentace lze vysvětlit tím, že je vyvíjen spíše dobrovolníky pod open-source licencí, je relativně nový a prochází rychlým vývojem (OpenCV, TensorFlow). Nejdůležitější použité nástroje, programy, platformy a knihovny jsou uvedeny v návrhu systému 3.1, avšak k dosažení výsledku bylo nutné použít další software, celkem jich bylo více než 25 (počítáno bez automaticky stažených závislostí). Podrobný seznam použitého softwaru nutný pro reprodukci práce nebo alespoň spuštění programu lze nalézt v příloze A. Důležité pro implementaci byly skripty v jazyce Python.

4.2 Vytvoření datasetu

Vzhledem k tomu, že řešení je založeno na konvoluční neuronové síti, vznikla potřeba trénovací množiny dat. Zdrojem byli videa poskytnuté vedoucím práce z měření výzkumné skupiny STRaDe z Ústavu inteligentních systémů. Videa jsou získány pozemní pohybující se kamerou Panasonic HC-VX981K[1] a zachycují let dronů v přírodě, včetně vzletu a přistávání. Kamera má 20× optický zoom a rozlišení 4K (3840×2160). Dron je na pozadí oblohy nebo terénu (louka, les) v různých vzdálenostech od kamery, v rozmezí jednotek až stovek metrů. Některá videa simulují nálet dronu, kdy dron nejdříve letí daleko kamery a poté se rychle vrací přímo ke kameře. Byly použity 2 drony: větší DJI Matrice 600 Pro[2] a menší

[1] https://shop.panasonic.com/cameras-and-camcorders/camcorders/HC-VX981K.html
[2] https://www.dji.com/cz/matrice600-pro/info

DJI Spark[3]. První má 6 vrtulí a velikost za letu $1668 \times 1518 \times 727$ mm, jedná se o velký profesionální dron. Druhý má 4 vrtule a velikost $143 \times 143 \times 55$ mm, jde o levný mini dron určený pro rekreační účely. Rozlišení videí je 2560×1440 pixelů a celková délka všech videí je 17 minut a 7 sekund. Součástí dat byla i videa a fotky zachycená infračervenou kamerou, ale ukázalo se, že pro detekci jsou data nepoužitelná, protože ve větších vzdálenostech dron přestal být viditelný.

Později byla dodána další část videí. Bohužel videa nejsou příliš kvalitní, na většině snímků nelze dron vidět vůbec nebo je příliš malý. Data jsou měřena kamerou Z na stejném místě jako předešlá. Byl použit pouze dron X. Rozlišení videí je 1920×1080 pixelů a celková délka všech videí je 28 minut, ale dron je přijatelně viditelný na 38 sekundách, a proto je informační hodnota minimální.

Z důvodu nedostatku dat a jejich malé různorodosti jsem se rozhodl získat další data na internetu a díky tomu dosáhnout vyšší přesnosti a lepší generalizace modelu [8]. První úvaha použít jednotlivé obrazy stažené automaticky skriptem z nějakého internetového vyhledávače či databáze se ukázala jako lichá, protože pokud obraz skutečně obsahoval dron, tak šlo většinou o katalogovou fotku. Stáhnul jsem proto videa zachycující létající dron v terénu z webu YouTube. Videa mají rozlišení 1280×720 nebo 1920×1080 pixelů. Předpoklad, že videí zachycující létající dron v terénu bude mnoho, se nenaplnil. I když u cenově dostupných dronů je dostatek videí, jde v drtivé většině o promo videa, návody, rozbalování nového dronu a záběry krajiny zachycené dronem. Hledání se stalo extrémně časově náročnou činností.

Trénování CNN modelu vyžaduje jednotlivé anotované obrazy, a proto bylo nutné videa nejdříve převést. K tomuto účelu jsem naprogramoval vlastní konzolovou aplikaci pro převod videí na obrazy využívající knihovnu OpenCV. Vstupními parametry nástroje jsou zdrojové video, cílová složka pro uložení obrazů a výška výsledného obrazu. Dalšími parametry jsou možnost nastavení kroku, kdy je uložen jen každý n-tý rámec videa a možnost vyříznutí, kdy je uložen pouze střed rámce, pokud je zvolená výška menší než původní. Nastavení kroku je nutné z důvodu rozdílnosti obrazů, po sobě jdoucí rámce se příliš neliší. Vyříznutí je vhodné pro získání malých objektů z velkého obrazu. Ve vyříznutém obrazu má objekt větší poměr velikosti k obrazu. Vyříznutí je důležité pro vytváření menších obrazů, které tak mají menší velikost a tím má menší velikost i trénovací dataset. Menší velikost trénovacího datasetu znamená menší paměťové nároky při trénování. Při změně výšky je zachován poměr stran a před převodem je vypsán celkový počet výsledných obrazů a výsledné rozlišení. Obrazy jsou ukládány ve formátu jpg se jménem img$X$, kde X je celé číslo. Některá videa bylo zapotřebí sestříhat, protože obsahovala větší části, na kterých se dron nevyskytoval nebo nebyl viditelný v požadované velikosti (příliš velký nebo malý). Pro střih videí byl použit program OpenShot Video Editor.

Vytvořené obrazy se často od sebe neliší i při větším kroku, obraz nemusí nutně obsahovat dron, dron může být oříznut okrajem obrazu nebo dron je velmi malý nebo nezřetelný. Z těchto důvodů je vhodná pouze malá podmnožina vytvořených obrazů. Výběr správných obrazů je tak velmi zdlouhavý. Po výběru je nutné přejmenovat a promíchat všechny obrazy, aby při dělení datasetu na trénovací a testovací část byli v obou částech rovnoměrně zastoupeny obrazy ze všech videí. Pro tento účel byl napsán skript v jazyce Python, jehož vstupním parametrem je složka s obrazy.

Množství obrazů získaných z videí bylo nedostačující, a proto část datasetu byla vytvořena uměle. Byly vytvořeny výřezy dronů v různých velikostech a s různou rotací. Tyto

[3] https://www.dji.com/cz/spark/info

výřezy poté byly umístěny do obrazů pozadí. Pro znásobení počtu výřezů byl vytvořen skript `resize_rotate_image.py`. Tento skript generuje ze vstupního výřezu 4 různé velikosti a navíc je výřez natočen v 5 pozicích, celkem je tedy z 1 výřezu vygenerováno 20 výřezů. Natočení je použito, protože z videí dronů je patrné, že při letu je dron často natočený, například při kroužení. Pro editaci obrazů byl použit program GIMP. Obrazy byly automaticky staženy pomocí skriptu `image_downloader.py`. Uměle vytvořený dataset z principu nemůže pokrýt všechny možné kombinace a obrazy nevypadají přirozeně, lepší volbou jsou obrazy získané v terénu.

Obrázek 4.1: Anotovaný příklad v nástroji LabelImg.

Obrazy byly ručně anotovány pomocí nástroje LabelImg, který umožňuje vytvářet anotace ve formátu Pascal VOC [16] nebo YOLO. Ve vybraném formátu Pascal VOC jsou anotace ukládány v XML souboru se stejným jménem jako anotovaný obraz. Anotace obsahuje informace jako jméno anotovaného obrazu, rozlišení obrazu a pozici bounding boxu. Na obr 4.1 a obr 4.2 lze vidět anotované příklady, kde dron je vyznačen bounding boxem.

Dataset vytvořený z videí má 712 příkladů v rozlišeních 888×500 až 1422×800 a umělý dataset má 1288 příkladů s jednotnou výškou obrazu 700 px, celkem tedy 2000 příkladů. S 2000 příklady jde o malý dataset, který je vhodný pro fine-tuning předem natrénovaného modelu. Z důvodu značné různorodosti vzhledu dronů jsou v datasetu obsaženy pouze 3 modely dronů v různých velikostech a v různém prostředí, všechny modely jsou vrtulové. Aby byl model schopen rozpoznat všechny existující typy modelů, bylo by nutné je zahrnout do datasetu. Takový úkol je více než náročný, na trhu jsou dostupné drony mnoha velikostí, barev, tvarů, drony se liší počtem vrtulí, mohou mít křídla, různý pohon a nést různý náklad od pokročilých kamer, přes různé senzory až po balíky, který rovněž mění vzhled dronu. Příklady různorodosti dronů jsou na obr 4.3.

Obrázek 4.2: Anotovaný příklad v nástroji LabelImg.

Obrázek 4.3: Příklady různorodosti vzhledu dronů [1].

4.3 Trénování modelu konvoluční neuronové sítě

Pro trénování modelu byla použita platforma TensorFlow, která má širokou komunitu uživatelů, nabízí pokročilé nástroje a je podporována v modulu dnn knihovny OpenCV. Detekce objektů je jedna z mnoha oblastí, ve které se používají konvoluční neuronové sítě. TensorFlow Object Detection API je open source framework vystavěný na platformě TensorFlow nabízející možnost vytvářet, trénovat, vyhodnocovat a nasazovat CNN modely pro detekci objektů. Trénování sítě od začátku je velmi zdlouhavé a je k němu potřeba velká datová množina. Proto je použit tzv. transfer learning, kde je použit předtrénovaný model na velkém a více obecném datasetu určený pro podobný problém. Poté je model natrénován na datasetu pro konkrétní problém. Při transfer learningu se tak využívá naučených vah předtrénovaného modelu oproti náhodným váhám při trénování od začátku.

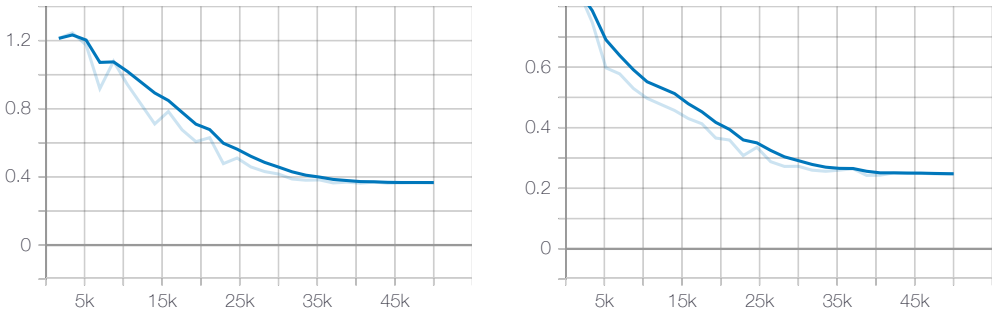
Z nabízených modelů pro detekci objektů byl zvolen model `ssd_mobilenet_v1_fpn_coco` kvůli vysoké přesnosti a současně dostatečné rychlosti. Autoři uvádějí 32 mean Average Precision (mAP) na COCO datasetu [33] a orientační rychlost detekce 56 ms na grafické kartě

NVIDIA GeForce GTX TITAN X[4]. Model používá síť Mobilenet [27] spolu Feature pyramid network (FPN) [34] pro extrakci rysů, vstupní obraz má vysoké rozlišení 640×640 a je natrénován na COCO datasetu. FPN síť spolu s vyšším rozlišením výrazně zlepšuje detekci malých objektů. Detailně jsou konvoluční neuronové sítě popsány v podkapitole 2.6.

Platforma TensorFlow pracuje pouze s binárním formátem TFRecord používaným pro reprezentaci datasetu. Datasety v jiném formátu je proto nutné převést. Pro převod z Pascal VOC formátu generovaného nástrojem LabelImg byl použit upravený skript `create_pascal_tf_record.py` dostupný na přiloženém DVD.

Pro spuštění trénování je nutný kromě datasetu konfigurační soubor. V tomto souboru je definovaný model a jeho parametry a parametry trénování. Pro model je možné nastavit parametry jako počet detekovaných tříd, vlastnosti anchor box generátoru, vstupní rozlišení sítě, aktivační funkci extraktoru rysů, loss funkci, inicializační funkci extraktoru rysů, počet vrstev sítě a další. Nastavení trénování definuje použitý model, cesty k trénovací a validační části datasetu, způsob vyhodnocování, počet kroků, možnosti augmentace dat atd. Konfigurační soubor byl upraven pro danou úlohu – byly nastaveny cesty k souborům, zvětšen počet kroků, upraven počet detekovaných tříd a možnosti augmentace dat.

Z důvodu urychlení trénování jsem zvolil vzdálené trénování na Google Cloud Platform. Na výpočetním serveru jsou aktuálně dostupné vysoce výkonné grafické karty NVIDIA Tesla P100 a NVIDIA Tesla K80, které výrazně urychlují trénování sítě. Online lze sledovat výstup logu nebo pro podrobnější informace použít nástroj TensorBoard, který je součástí platformy TensorFlow. TensorBoard spouští lokální server, který generuje webovou stránku s výsledky. Naneštěstí není zatím implementovaná podpora TensorBoardu pro Google Cloud Platform, a proto jsem byl nucen nainstalovat TensorFlow i na OS Linux a TensorBoard spustit zde.

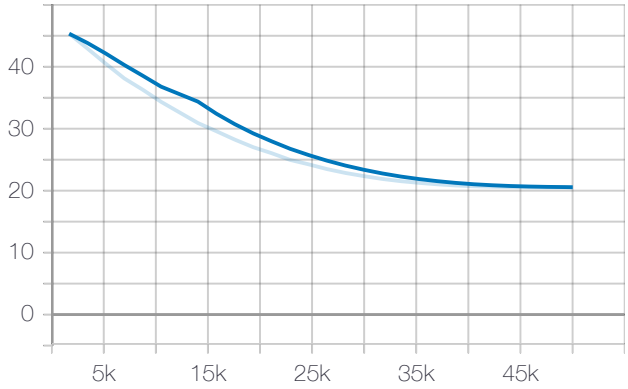


Obrázek 4.4: Graf Loss funkce. Vlevo klasifikační Loss funkce, vpravo lokalizační Loss funkce.

V první fázi trénování byl počet anotovaných příkladů 576. Tento malý dataset byl rozdělen na 80 % trénovací část a 20 % testovací část. Batch size byl kvůli nedostatku paměti snížen na 16 příkladů. Po 75 000 krocích byla hodnota Loss funkce cca $0,5$. Model dosáhl Average Precision $0,45$ a Average Recall $0,5$, což je nedostačující, ale očekávaný výsledek z důvodu malého a nevyváženého datasetu a nedostatečně natrénovaného modelu.

Ve druhé fázi trénování byl použit vytvořený dataset s 2000 příklady. Trénovací část má 1800 příkladů a testovací 200. Batch size zůstal na 16 příkladech. Výsledná hodnota Loss funkce je $20,103$. Hodnota je relativně velká, je zde prostor pro další zlepšení modelu. Model dosáhl Average Precision $0,769$, což je výrazně lepší výsledek než v předchozí fázi.

[4] https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/



Obrázek 4.5: Graf celkové Loss funkce.

Trénování bylo spuštěno třikrát, na obr 4.4 a obr 4.5 lze vidět vývoj Loss funkcí z posledního běhu. Podrobné vyhodnocení detektoru je v následující kapitole.

4.4 Výsledná aplikace

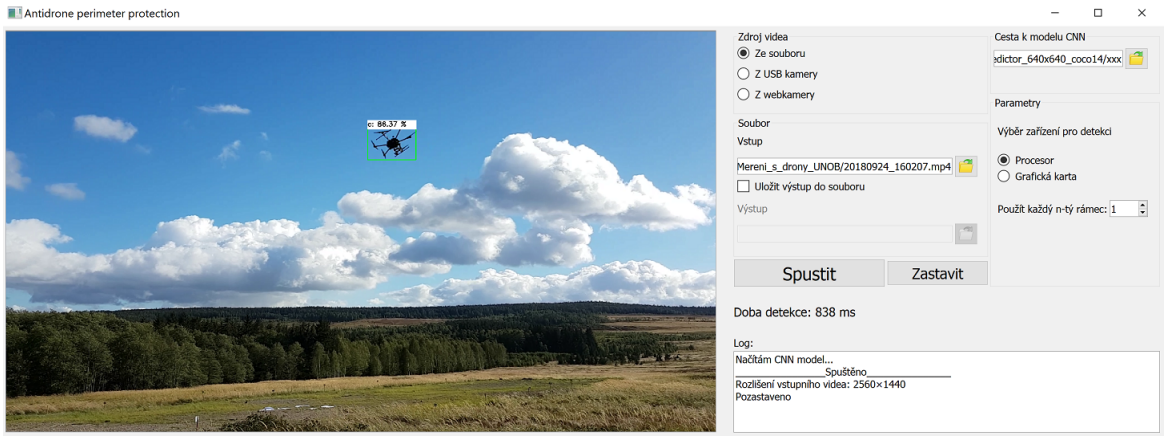
Výsledná aplikace je implementována modulárně, skládá se ze 3 tříd a to:
`Antidrone_perimeter_protection`, `VideoPlayer` a `Detector`. Třídy jsou navzájem oddělené a kód tak může být znovupoužitelný pro jinou aplikaci zabývající se prací s videem.

První třída slouží k obsluze grafického uživatelského prostředí, implementuje akce widgetů, ovládá widgety a zobrazuje výstup detektoru, logu a dobu detekce. Aplikace v levé části zobrazuje video výstup, v pravé části jsou ovládací prvky. Jako zdroj videa je možné zvolit videosoubor, USB kameru nebo webkameru. Pro funkčnost USB kamery je nutné mít v systému nainstalovaný ovladač pro danou kameru. Je možné uložit výstup detektoru do videosouboru. Videosoubor výstupu je zakódován kodekem H.264 a uložen do formátu mp4. Před spuštěním detektoru je zapotřebí zvolit adresář s modelem CNN. Volitelně může uživatel vybrat zařízení pro detekci, standardně je vybrán procesor, druhou možností je grafická karta. Další funkcionalitou je vynechávání rámců pro urychlení detekce. Grafické uživatelské rozhraní je implementováno knihovnou Qt.

Třída `VideoPlayer` provádí všechny operace s videi: otevírání videosouborů, výstupů z kamer, ukládání videosouboru, čtení videa. Dále měří dobu detekce. Prosté spuštění zpracování videa by znemožnilo uživateli jakkoli ovládat aplikaci a zároveň by se nepřekreslovalo okno videa. Je tedy nutné vytvořit vlákno určené jen pro zpracování videa. Vytvoření vlákna je implementováno děděním z třídy `QThread` knihovny Qt. Umožňuje tak ovládání grafického uživatelského rozhraní, vykreslování snímků videa i pozastavení a znovuspuštění zpracování videa. Otevírání a přehrávání videí zajišťuje knihovna OpenCV, která k tomuto účelu používá vestavěnou knihovnu FFmpeg[5], takže aplikace je schopná otevřít všechny formáty podporované touto knihovnou. Překreslování okna videa je implementováno Qt signály.

Detektor je implementován ve třídě `Detector`, která načítá model sítě a poté provádí samotnou detekci. Video je zpracováváno snímek po snímku a pro detekci je volána metoda `detect`. V ní je snímek zmenšen na pevné rozlišení 640×640 a převeden do formátu vhodného pro model CNN. Poté je zavolána metoda pro průchod sítí z modulu dnn. Vrácen je

[5] https://ffmpeg.org/



Obrázek 4.6: Výsledná aplikace.

seznam detekcí. Detekce je soubor šesti hodnot: číslo detekované třídy, pravděpodobnost s jakou se v bounding boxu vyskytuje objekt dané třídy a 4 souřadnice bounding boxu. Pokud je pravděpodobnost výskytu větší než zvolený práh, je do původního snímku vykreslen bounding box včetně pravděpodobnosti výskytu.

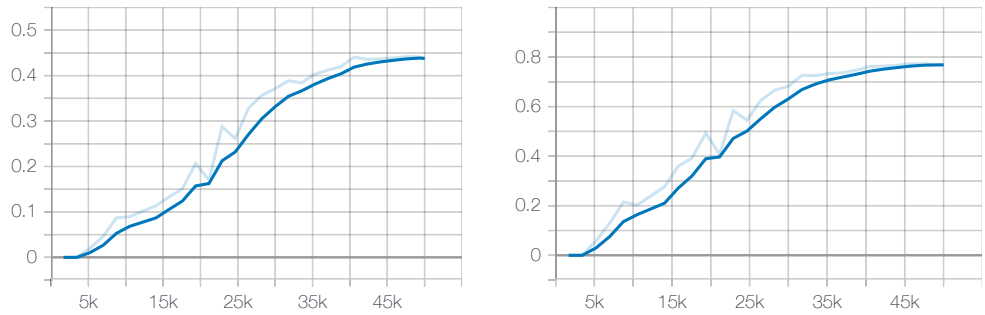
Natrénovaný model CNN je před jeho použitím pro detekci nutné vyexportovat. K tomu slouží skript z TensorFlow Object Detection API. Takto vyexportovaný model má naneštěstí jinou vnitřní strukturu než původní model před trénováním. Proto je nutné změnit strukturu tak, aby byl model CNN kompatibilní s dnn modulem. K tomu je zapotřebí skript TransformGraph.py dostupný na přiloženém DVD. Pro takto připravený model je ještě nutné vytvořit textovou verzi grafu modelu pro modul dnn.

Aplikace umožňuje akceleraci detekce na grafické kartě podporující framework OpenCL. Na počítačích s více grafickými kartami je vybrána karta, která je v OpenCL výchozí. Změna karty je prováděna proměnnou prostředí `OPENCV_OPENCL_DEVICE`. Z neznámých důvodů nefunguje nastavení této proměnné prostředí v možnostech sestavení ve Visual Studiu, byl jsem tak nucen nastavit proměnnou jako permanentní.

Kapitola 5

Testování

Kvalita detektoru byla ověřena při trénování a na vybraném videu. Výhodou trénování detektorů objektů založených na konvolučních neuronových sítích pomocí platformy TensorFlow je možnost přímého vyhodnocování během trénování. Je možné zvolit COCO metriky pro průběžné vyhodnocování, uživatel tak má přehled o kvalitě detektoru. Pro zobrazení průběžných výsledků nabízí TensorFlow nástroj Tensorboard, jak je popsáno v podkapitole x. TensorBoard generuje grafy Precision a Recall podle COCO metrik[1]. Kromě standardní metriky Average Precision, kdy je jako pozitivní počítána detekce s Intersection over union větším než 0,5, obsahují COCO metriky Average Precision podle velikosti objektu. Malé objekty mají plochu menší než 32^2 pixelů, střední mají plochu od 32^2 do 96^2 pixelů a větší objekty spadají do kategorie velkých. Protože dron je v obraze většinou velmi malý, je rozhodující ukazatelem kvality detektoru dronů Average Precision^{small}, tedy kategorie malých objektů. Zároveň platí, že čím menší objekt, tím náročnější je jeho detekce.

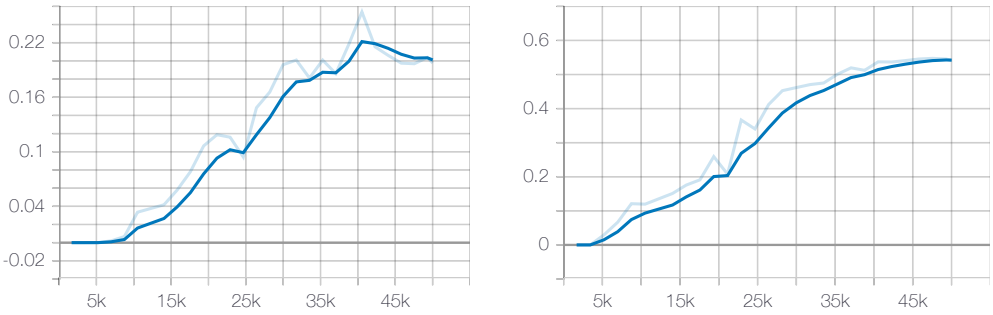


Obrázek 5.1: Graf Average Precision. Vlevo standardní COCO AP 0.5:0.95, vpravo AP 0.5 (Pascal VOC metrika).

Výsledný model má Average Precision 0, 769. Na obr 5.1 jsou vidět grafy Average Precision natrénovaného detektoru. Hodnotu 0, 769 Average Precision lze hodnotit jako velmi vysokou, vzhledem k vysoké hodnotě Loss funkce. Značí to, že při dalším trénování by model dosáhl výrazně lepších výsledků. Na obr 5.2 jsou grafy pro AP small a AP medium. Podle očekávání je hodnota AP small výrazně menší než u AP medium.

Na obr 5.3 lze vidět úspěšnou detekci. Dron je dostatečně velký a kontrastní vůči pozadí. Neúspěšná detekce je zobrazena na obr 5.4. Důvodem je malá velikost dronu a malý kontrast vůči pozadí.

[1] http://cocodataset.org/?source=post_page----------#detection-eval



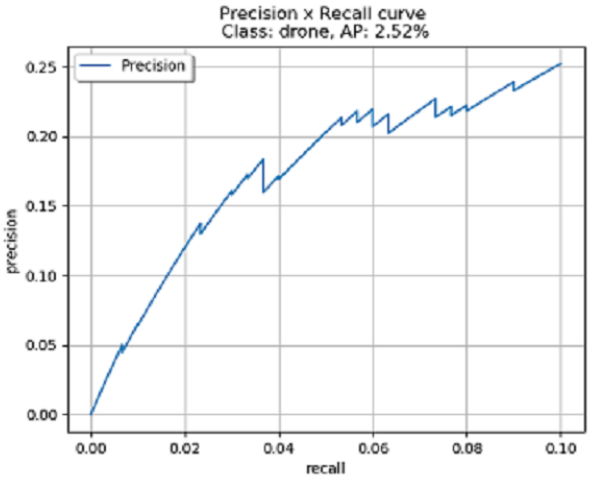
Obrázek 5.2: Graf Average Precision. Vlevo AP small pro malé objekty, vpravo AP medium pro objekty střední velikosti.

Obrázek 5.3: Úspěšná detekce. Vlevo detekce, vpravo ground-truth obraz.

Obrázek 5.4: Neúspěšná detekce. Vlevo detekce, vpravo ground-truth obraz.

Pro vyhodnocení detektoru bylo použito video z prvního měření. Toto video nebylo použito pro vytvoření datasetu, pro model tak jde o neviděná data. Video bylo zkráceno na 300 snímků při zachování původního rozlišení 2560×1440. Drony jsou zachyceny v různých pozicích, většina dronů má střední velikost. Video bylo převedeno na jednotlivé obrazy pomocí vlastní aplikace a obrazy poté anotovány v nástroji LabelImg. Pro výpočet a vykreslení Average Precision do grafu byl použit nástroj Object Detection Metrics. Tento nástroj pracuje s dvěma složkami obsahujícími anotace v jednoduchém formátu pro ground-truth obrazy a pro detekce. Nástroj LabelImg tvoří anotace ve formátu Pascal VOC, proto je nutné je nejdříve převést na formát pro Object Detection Metrics. K tomuto účelu byl napsán skript `parse_pascal_to_txt.py`. Anotace pro detekce jsou generovány z aplikace. Na grafu 5.5 lze vidět výslednou Precision-Recall křivku. Výsledky nejsou nijak dobré, Average Precision

je jen $0,025$. Při přehrání výsledného videa je vidět, že na začátku videa je chybně detekovaná obloha jako dron. Zhruba uprostřed videa je dron větší, lze vidět úspěšná a velmi přesné detekce. Se zmenšující se velikostí přesnost klesá.



Obrázek 5.5: Precision-Recall křivka pro video `test_video.mp4`.

Rychlost detekce je průměrně 850 ms na čtyřjádrovém procesoru `Intel i7-6700HQ`[2], kde při plném využití je jedno jádro přetaktováno až na 3,5 GHz. Tuto rychlost lze hodnotit jako velmi dobrou, vzhledem k vysokému vstupnímu rozlišení 640×640 a použití Feature Pyramid Network.

[2] https://ark.intel.com/content/www/us/en/ark/products/88967/intel-core-i7-6700hq-processor-6m-cache-up-to-3-50-ghz.html

Kapitola 6

Závěr

Tato práce je součástí většího projektu na detekci dronů, který reaguje na současné bezpečnostní hrozby plynoucí ze zvyšujícího se rozšíření bezpilotních létajících strojů a možností zneužití této technologie. Cílem této práce bylo prostudovat současné metody pro detekci pohybujících se objektů ve videu a vybranou metodu implementovat a vyhodnotit. Tento cíl byl splněn.

Byla důkladně prostudována literatura popisující problematiku detekce pohybujících se objektů ve videu zachyceném pohybující se kamerou. Známé metody byly zařazeny do kategorií a stručně porovnány jejich vlastnosti. Vzhledem k tomu, že daný úkol je velmi náročný, byly popsány a ilustrovány problémy spojené s detekcí pohybujících se objektů s ohledem na detekci dronů. Dále byly popsány používané metriky pro vyhodnocení metod detekce objektů. Z metod pro detekci objektů byly vybrány tři přístupy: konvoluční neuronové sítě, segmentace a kosegmentace. Pro řešení práce byla prvně zvolena metoda založená na kosegmentaci, která se později ukázala jako nevhodná. Byla proto zvolena metoda založená na konvolučních neuronových sítích.

Problémem při studování metod detekce objektů byl nedostatek literatury týkající se přímo detekce dronů. Byly nalezeny pouze dvě práce explicitně popisující detekci dronů, důvodem je to, že problém je relativně nový. S tím souvisí i akutní nedostatek materiálů jako předpřipravené datasety, fotky a videa zachycující létající drony v porovnání například s detekcí chodců.

Pro trénování konvoluční neuronové sítě bylo potřeba vytvořit dataset. Množství dodaných videí dronů bylo nedostačující, a proto byly vyhledány další na internetu. I s novými videi byl počet z nich získaných obrazů nedostačující a byly vytvořeny umělé obrazy z výřezů dronů a stažených fotek pozadí. Pro vytvoření datasetu bylo vytvořeno několik nástrojů: pro převod videí na obrazy, hromadné přejmenování souborů a generování výřezů dronů. Obrazy byly ručně anotovány. Výsledný dataset má 2000 příkladů. Na tomto datasetu byla natrénována síť Single Shot MultiBox Detector.

Navržený systém pro detekci dronů byl implementován za použití metody založené na konvoluční neuronové síti. Výsledná aplikace demonstruje detekci dronů ve videu s možností použití výstupu připojené kamery jako zdroje videa a uložení výsledku.

Teoretická část práce může posloužit čtenářům jako úvod do problematiky detekce objektů. Vytvořený dataset a nástroje jsou použitelné pro další práce. Natrénovaná síť dosahuje Precision X a Recall Y, což lze považovat za dobrý výsledek vzhledem k velmi malé velikosti dronů v obraze a limitům použité metody.

Výsledky použitého přístupu je možné vylepšit použitím výrazně většího datasetu. Současně je potřeba zachytit co nejvíce možných typů, velikostí, barev a tvarů dronů v různých

pozicích a v různém prostředí reálného světa. Další možností je optimalizace modelu pro daný dataset. Z hlediska rychlosti detekce je nutné použít akceleraci na vysoce výkonné grafické kartě nebo několika kartách. S takovouto konfigurací se bude rychlost detekce blížit zpracování v reálném čase. Obecně ale při detekci dronů není vhodné spoléhat jen na detekci z videa, protože dron ve videu není ve větší vzdálenosti bez použití zoomu viditelný. Dalším problémem je nedostatek světla zejména v noci a také vliv počasí na kvalitu videa. Řešením je použití více zdrojů informace jako radar, řídící signály dronu, infrakamera nebo mikrofon. Otázkou zůstává další pokračování celého projektu, protože jsou již dostupné vysoce profesionální komerční řešení pro detekci dronů.

Literatura

[1] *Google obrázky*. [Online; navštíveno 05.07.2019].
URL https://www.google.cz

[2] *Understanding Convolutions*. 2014, [Online; navštíveno 12.07.2019].
URL http://colah.github.io/posts/2014-07-Understanding-Convolutions/img/RiverTrain-ImageConvDiagram.png

[3] *East Coast Snowstorm Time Lapse 22 - 24 January 2016*. 2016, [Online; navštíveno 20.02.2019].
URL https://www.youtube.com/watch?v=VSaFnWeUHPo

[4] *What is max pooling in convolutional neural networks?* 2017, [Online; navštíveno 12.07.2019].
URL https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

[5] *2019 Most BRUTAL/AGONIZING Fatal Car Crash Compilation in USA/Europe #31*. 2018, [Online; navštíveno 22.02.2019].
URL https://www.youtube.com/watch?v=X7dtSlBIePQ

[6] *Webcams.travel Brno Malinovského náměstí*. 2019, [Online; navštíveno 22.02.2019].
URL https://cz.webcams.travel/webcam/fullscreen/1206291667

[7] Achanta, R.; Shaji, A.; Smith, K.; aj.: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE transactions on pattern analysis and machine intelligence*, ročník 34, 05 2012, doi:10.1109/TPAMI.2012.120.

[8] Aker, C.; Kalkan, S.: Using Deep Networks for Drone Detection. 08 2017, doi:10.1109/AVSS.2017.8078539.

[9] Bandyopadhyay, S.: Object of interest detection in video sequence using co-segmentation: A new era in video surveillance. 03 2018: s. 1–6, doi:10.1109/RAIT.2018.8389022.

[10] Bay, H.; Ess, A.; Tuytelaars, T.; aj.: Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, ročník 110, 06 2008: s. 346–359, doi:10.1016/j.cviu.2007.09.014.

[11] Chen, L.-C.; Papandreou, G.; Kokkinos, I.; aj.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, ročník 40, č. 4, 2018: s. 834–848.

[12] Cheng, M.-M.; J Mitra, N.; Huang, X.; aj.: Salient Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 37, 10 2011, doi:10.1109/TPAMI.2014.2345401.

[13] Comaniciu, D.; Ramesh, V.; Meer, P.: Real-Time Tracking of Non-Rigid Objects using Mean Shift. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ročník 2, 05 2000.

[14] Cortes, C.; Vapnik, V.: Support Vector Network. *Machine Learning*, ročník 20, 09 1995: s. 273–297, doi:10.1007/BF00994018.

[15] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, ročník 2, 06 2005.

[16] Everingham, M.; Van Gool, L.; Williams, C. K. I.; aj.: The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, ročník 88, č. 2, Červen 2010: s. 303–338.

[17] F. Chan, T.; Vese, L.: Active Contour Without Edges. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, ročník 10, 02 2001: s. 266–77, doi:10.1109/83.902291.

[18] Falanga, D.; Kleber, K.; Mintchev, S.; aj.: The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly. *IEEE Robotics and Automation Letters*, ročník PP, 12 2018: s. 1–1, doi:10.1109/LRA.2018.2885575.

[19] Freund, Y.; E Schapire, R.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. ročník 55, 12 1999: s. 119–139, doi:10.1006/jcss.1997.1504.

[20] Geng, Q.; Zhou, Z.; Cao, X.: Survey of recent progress in semantic image segmentation with CNNs. *Science China Information Sciences*, ročník 61, č. 5, 2018: str. 051101.

[21] Giraud, R.; Ta, V.; Papadakis, N.; aj.: Texture-Aware Superpixel Segmentation. 01 2019.

[22] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, 326 s., http://www.deeplearningbook.org.

[23] Guo, Z.: *Object detection and tracking in video*. [Online; navštíveno 23.01.2019]. URL http://medianet.kent.edu/surveys/IAD01F-objdetection/index.html

[24] Han, S.; Shen, W.; Liu, Z.: *Deep Drone: Object Detection and Tracking for Smart Drones on Embedded System*. [Online; navštíveno 24.01.2019]. URL https://web.stanford.edu/class/cs231a/prev_projects_2016/deep-drone-object__2_.pdf

[25] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. 06 2016: s. 770–778, doi:10.1109/CVPR.2016.90.

[26] Horn, B.; G. Schunck, B.: Determining Optical Flow. *Artificial Intelligence*, ročník 17, 08 1981: s. 185–203, doi:10.1016/0004-3702(81)90024-2.

[27] Howard, A. G.; Zhu, M.; Chen, B.; aj.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, ročník abs/1704.04861, 2017, 1704.04861.
URL http://arxiv.org/abs/1704.04861

[28] Hui, J.: *What do we learn from single shot object detectors (SSD, YOLOv3), FPN & Focal loss (RetinaNet)?* [Online; navštíveno 19.06.2019].
URL https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d

[29] Jepson, D. F. . A.: *Markov Random Fields*. [Online; navštíveno 26.03.2019].
URL
http://www.cs.toronto.edu/~fleet/courses/2503/fall11/Handouts/mrf.pdf

[30] Joulin, A.; Bach, F.; Ponce, J.: Discriminative clustering for image co-segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 06 2010: s. 1943–1950, doi:10.1109/CVPR.2010.5539868.

[31] Krizhevsky, A.; Sutskever, I.; E. Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, ročník 25, 01 2012, doi:10.1145/3065386.

[32] Li, S. Z.: *Markov Random Field Modeling in Computer Vision*. Berlin, Heidelberg: Springer-Verlag, 1995, ISBN 4-431-70145-1, [Online; navštíveno 26.03.2019].
URL http://www.nlpr.ia.ac.cn/users/szli/MRF_Book/MRF_Book.html

[33] Lin, T.; Maire, M.; Belongie, S. J.; aj.: Microsoft COCO: Common Objects in Context. *CoRR*, ročník abs/1405.0312, 2014, 1405.0312.
URL http://arxiv.org/abs/1405.0312

[34] Lin, T.-Y.; Dollár, P.; Girshick, R.; aj.: Feature Pyramid Networks for Object Detection. 12 2016.

[35] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. ročník 9905, 10 2016: s. 21–37, doi:10.1007/978-3-319-46448-0_2.

[36] Liu, Z.; Wang, L.; Hua, G.; aj.: Joint Video Object Discovery and Segmentation by Coupled Dynamic Markov Networks. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, ročník 27, 07 2018, doi:10.1109/TIP.2018.2859622.

[37] Meng, F.; Li, H.; Liu, G.; aj.: Image Cosegmentation by Incorporating Color Reward Strategy and Active Contour Model. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, ročník 43, 09 2012, doi:10.1109/TSMCB.2012.2215316.

[38] Owczarek, M.; Strumillo, P.; Baranski, P.: Pedestrian tracking in video sequences: a particle filtering approach. 09 2015, doi:10.15439/2015F158.

[39] Přemysl, K.: *Základy počítačové grafiky Studijní opora*. [Online; navštíveno 22.01.2019].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIZG-IT%2Ftexts%2Fizg_opora.pdf&cid=11496

[40] R. Rodríguez-Canosa, G.; Thomas, S.; Cerro, J.; aj.: A Real-Time Method to Detect and Track Moving Objects (DATMO) from Unmanned Aerial Vehicles (UAVs) Using a Single Camera. *Remote Sensing, vol. 4, issue 4, pp. 1090-1111*, ročník 4, 04 2012: s. 1090–1111, doi:10.3390/rs4041090.

[41] Redmon, J.; Farhadi, A.: YOLO9000: Better, Faster, Stronger. 07 2017: s. 6517–6525, doi:10.1109/CVPR.2017.690.

[42] Redmon, J.; Farhadi, A.: YOLOv3: An Incremental Improvement. 04 2018.

[43] Ren, S.; He, K.; Girshick, R.; aj.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, 06 2015, doi:10.1109/TPAMI.2016.2577031.

[44] Rother, C.; Minka, T.; Blake, A.; aj.: Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs. *CVPR*, ročník 1, 07 2006: s. 993– 1000, doi:10.1109/CVPR.2006.91.

[45] Rozantsev, A.; Lepetit, V.; Fua, P.: Flying Objects Detection from a Single Moving Camera. 11 2014, doi:10.1109/CVPR.2015.7299040.

[46] Saha, S.: *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [Online; navštíveno 12.07.2019].
URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[47] Shapiro Linda, G.: *Computer vision*. New Jersey: Prentice-Hall, vyd. 1 vydání, 2001, ISBN 0-13-030796-3.

[48] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*, 09 2014.

[49] Soomro, N.: An Efficient Image Co-segmentation Algorithm based on Active Contour and Image Saliency. 02 2016.

[50] Svoboda, T.: *Image as Markov Random Field and Applications*. [Online; navštíveno 28.03.2019].
URL http://cmp.felk.cvut.cz/~drbohlav/TeachPres/mrf.pdf

[51] Szeliski, R.: *Computer Vision : Algorithms and Applications*. London: Springer, 2010, ISBN 9781848829343.

[52] Taniai, T.; Sinha, S.; Sato, Y.: Joint Recovery of Dense Correspondence and Cosegmentation in Two Images. 06 2016: s. 4246–4255, doi:10.1109/CVPR.2016.460.

[53] Tsai, Y.-H.; Yang, M.-H.; Black, M.: Video Segmentation via Object Flow. 06 2016: s. 3899–3908, doi:10.1109/CVPR.2016.423.

[54] Tsai, Y.-H.; Zhong, G.; Yang, M.-H.: Semantic Co-segmentation in Videos. ročník 9908, 10 2016: s. 760–775, doi:10.1007/978-3-319-46493-0_46.

[55] Wang, L.; Hua, G.; Sukthankar, R.; aj.: Video Object Discovery and Co-Segmentation with Extremely Weak Supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník PP, 10 2016: s. 1–1, doi:10.1109/TPAMI.2016.2612187.

[56] Yazdi, M.; Bouwmans, T.: New Trends on Moving Object Detection in Video Images Captured by a moving Camera: A Survey. *Computer Science Review*, ročník 28, 05 2018: str. 157–177, doi:10.1016/j.cosrev.2018.03.001.

[57] Zhao, H.; Qi, X.; Shen, X.; aj.: Icnet for real-time semantic segmentation on high-resolution images. 2018: s. 405–420.

Příloha A

Použité nástroje

Pro implementaci systému bylo zapotřebí vybrat vhodné nástroje, programy a knihovny. Z důvodu dobré podpory, obsahu vzorových aplikací a široké komunity uživatelů byla pro práci s obrazem využita knihovna OpenCV. Dalšími důvody je programovací jazyk knihovny C++ a podpora akcelerace na GPU. Pro grafické uživatelské rozhraní byla použita knihovna Qt z důvodu předchozí zkušenosti s touto knihovnou a jednoduchostí tvorby. Knihovna má mnoho uživatelů, takže lze nalézt dostatek návodů a dotazů s problémy. Jako detekční algoritmus bylo zvolené řešení založené na konvoluční neuronové síti, bylo tedy nutné vybrat vhodnou platformu pro trénování a manipulaci s modelem CNN. Zvolená platforma TensorFlow je v současnosti jedna z nejpoužívanějších platforem pro práci s neuronovými sítěmi, má širokou komunitu uživatelů včetně mnoha velkých společností a k ní dostatek informací. Akcelerace na grafické kartě byla vybrána platforma NVIDIA CUDA. Pro vývoj aplikací přímo pro tuto platformu je nutné použít vývojové prostředí Visual Studio. Bylo tedy vybráno pro vývoj a překlad výsledné aplikace už jen s ohledem na vývoj v operačním systému Microsoft Windows 10. Všechny ostatní použité nástroje, programy, knihovny, interprety a balíčky a další software byl vybrány s ohledem na kompatibilitu nebo vyžadovány pro výše uvedený nebo jiný software. Většina softwaru je šířena pod open source licencí nebo se jedná o freeware. V tabulkách A.1 a A.2 je přehled použitého softwaru a jeho verze. Uvedený software je bez automaticky stažených závislostí. Většina softwaru bude fungovat i na novějších verzích, ale můžou nastat problémy s kompatibilitou.

Tabulka A.1: Použité nástroje.

Název	**Verze**	**Popis**	**Poznámka**
Cmake	3.14.2	Kompilace	Sestavení a instalace OpenCV
cocoapi	-	API pro COCO dataset	Tensorflow Object Detection API
contextlib2	0.5.5	Python modul	TensorFlow Object Detection API
Cython	0.29.7	Python modul	TensorFlow Object Detection API
Doxygen	1.8.6	Generování dokumentace	
GIMP	2.10.12	Editace obrazů	Pro vytvoření umělého datasetu
Git	2.21.0	Verzovací nástroj	Pro stažení OpenCV, CNN modelů a dalších
Git-bash	-	Unix příkazový řádek pro Git	Sestavení a instalace OpenCV
Google Cloud SDK		SDK pro Google Cloud Platform	Pro trénování na cloudu
Jupyter notebook	4.4.0	Python modul	TensorFlow Object Detection API
LabelImg	1.8.1	Anotace datasetů	
Linux Ubuntu	16.4	Operační systém	Pro podporu Google Cloud v TensorBoardu
lxml	4.3.3	Python modul	TensorFlow Object Detection API
Matplotlib	3.0.3	Python modul	TensorFlow Object Detection API
NVIDIA CUDA	10.0	Platforma pro výpočty na NVIDIA GPU	GPU akcelerace
NVIDIA CUDA cuDNN	7.5.0.56	Knihovna pro neuronové sítě na NVIDIA GPU	GPU akcelerace
NVIDIA TensorRT	5.0.4.3	knihovna pro urychlení průchodu modelem sítě na NVIDIA GPU	GPU akcelerace
OpenCV	4.1.0	Knihovna pro počítačové vidění	
OpenShot	2.4.4	Editace videí	Střih videí
Oracle VM VirtualBox	5.2.8	Virtualizace OS	Pro Linux

Tabulka A.2: Použité nástroje (pokr.).

Název	**Verze**	**Popis**	**Poznámka**
Pillow	6.0.0	Python modul	TensorFlow Object Detection API
Protobuf	3.7.1	Python modul	TensorFlow Object Detection API
Python 2	2.7.16	Programovací jazyk	Pro Google Cloud SDK
Python 3	3.7.3	Programovací jazyk	
Qt	5.12.2	GUI	
Qt Visual Studio Tools	2.3.2	Doplněk pro Visual Studio	Pro sestavení Qt projektu
TensorFlow	1.13.1	Platforma pro neuronové sítě	
Visual Studio 2017	Community	Vývojové prostředí	
Windows Path Editor	1.7	Editace Path v OS Windows	Pro instalaci OpenCV a dalších
Windows Software Development Kit	10.0.17134.12	SDK	Pro Qt, OpenCV, NVIDIA CUDA, cuDNN, TensorRT
Windows Software Development Kit	10.0.15063.674	SDK	Pro Qt, OpenCV, NVIDIA CUDA, cuDNN, TensorRT
Windows Software Development Kit	10.0.16299.15	SDK	Pro Qt, OpenCV, NVIDIA CUDA, cuDNN, TensorRT
Windows Software Development Kit	10.0.17763.132	SDK	Pro Qt, OpenCV, NVIDIA CUDA, cuDNN, TensorRT