Sledování osoby v obraze

Van Martin Ha

Zvolte typ práce
20XX

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** nascannované zadání s. 1 ***

*** nascannované zadání s. 2 ***

*** naskenované Prohlášení str. 1 ***

*** naskenované Prohlášení str. 2 ***

ABSTRAKT

Text abstraktu v jazyce práce

Klíčová slova: klíčové slovo, klíčové slovo

ABSTRACT

Text abstraktu ve světovém jazyce (angličtině)

Keywords: keywords, keywords

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Jakubu Novákovi, Ph. D za konzultace, rady a odborné vedení při vypracovávání práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Počítačové vidění je obor umělé inteligence (AI), která napodobuje funkci lidského oka. Zařízení jako jsou fotoaparáty a kamery, poskytují digitální obraz z reálného života. Ty slouží jako zdroj pro počítačové vidění k rozeznání a zpracování různých objektů. Průběh zpracování můžeme rozdělit na snímání a digitalizaci, předpracování, segmentaci, klasifikace a rozpoznání obrazu. Historicky bylo vytváření systémů počítačového vidění považováno za výhradní doménu odborných výzkumníků a inženýrů v oblasti zpracování signálů a umělé inteligence a bylo obecně omezeno na vojenské účely. Implementaci počítačového vidění v současné době nalezneme v různých odvětvích jako jsou medicína, automobilový průmysl, bezpečnost a mnoho dalších.

I. TEORETICKÁ ČÁST

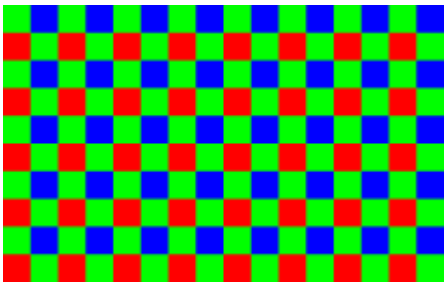
1 PRINCIP POČÍTAČOVÉHO VIDĚNÍ

1.1 Obraz

Digitální obraz je vyjádřen jako spojitá funkce f (x, y) dvou souřadnic v rovině, kde x a y jsou souřadnice bodu, který je zachytávaný na čipu videokamery. Tento signál je vzorkovaný do matice s M řádky a N sloupci. Kvantování obrazu přiřadí každému spojitému vzorku celočíselnou hodnotu, u šedo tónového obrazu například 256 hodnot. Dostáváme tak matici přirozených čísel, které představují funkční hodnotu funkce f (x, y) a tou je úroveň jasu v bode (x, y). Tento bod se nazývá pixel. [1]

1.1.1 Jak kamera snímá obraz

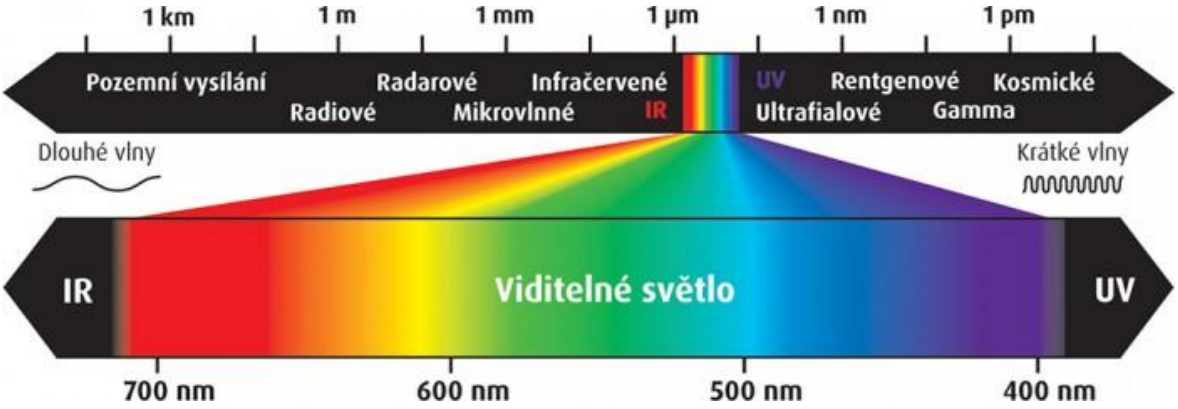
Přes objektiv kamery dopadá světlo na senzor kamery (CCD nebo CMOS), ten se skládá miliónů digitálních bodů. Předtím než světlo dopadne na senzor, prochází mozaikově (Bayerově) uspořádaných RGB filtrem, který nechá projít jen jednu barvu na jeden bod senzoru. Senzor poté uloží dlouhý vektor nezpracovaných dat. Jednotlivé barvy se interpolují s nejbližšími pixely, aby se určila plná hodnota RGB pro každý bod senzoru. [2]



Obrázek 1 Bayerově uspořádaný filtr [7]

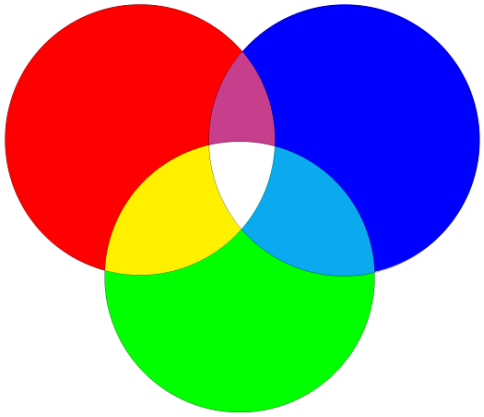
1.1.2 Barevný model RGB

Světlo je tvořeno elektromagnetickými vlnami. Tyto vlny se šíří z nějakého zdroje světla. Různé barvy mají různé vlnové délky, nejdelší vlnová délka světla, kterou lidé mohou vidět, je červená, nejkratší je fialová. Pokud se díváme na objekt, vlnové délky odraženého světla určují, jakou barvu vidíme. Barva se již dlouho používá v malbě, ve fotografiích a ve filmech k zobrazení okolí podobným způsobem, jakým lidé vidí svět ve skutečnosti. Lidé vnímají jen úzkou část elektromagnetického spektra s vlnovou délkou přibližně od 380 nm do 740nm. Viditelné barvy se nazývají spektrální barvy. [3] [4]



Obrázek 2 Spektrum viditelného světla [5]

Počítač ukládá obraz ve formě pixelů, každý pixel je reprezentován třemi 8bitovými kanály (0–255 v celočíselné podobě), dohromady tvoří 24bitovou hodnotu. Každá hodnota zastupuje jednu ze tří základních barev – červenou, zelenou a modrou. Můžeme vytvořit jakoukoliv barvu smícháním několika intenzit RGB, jednotlivé složky barev se sčítají a vytvářejí světlo větší intenzity. [3]



Obrázek 3 RGB [6]

1.2 Předpracování obrazu

Předzpracování nezvyšuje obsah informací, ale naopak typicky snižuje. Nicméně je velice užitečné v různých situacích, protože potlačuje nežádoucí zkreslení nebo vylepšuje některé vlastnosti obrazu důležité pro další zpracování. Existují různé metody, které upravují obraz – geometrické transformace, úprava jasu a další. [5]

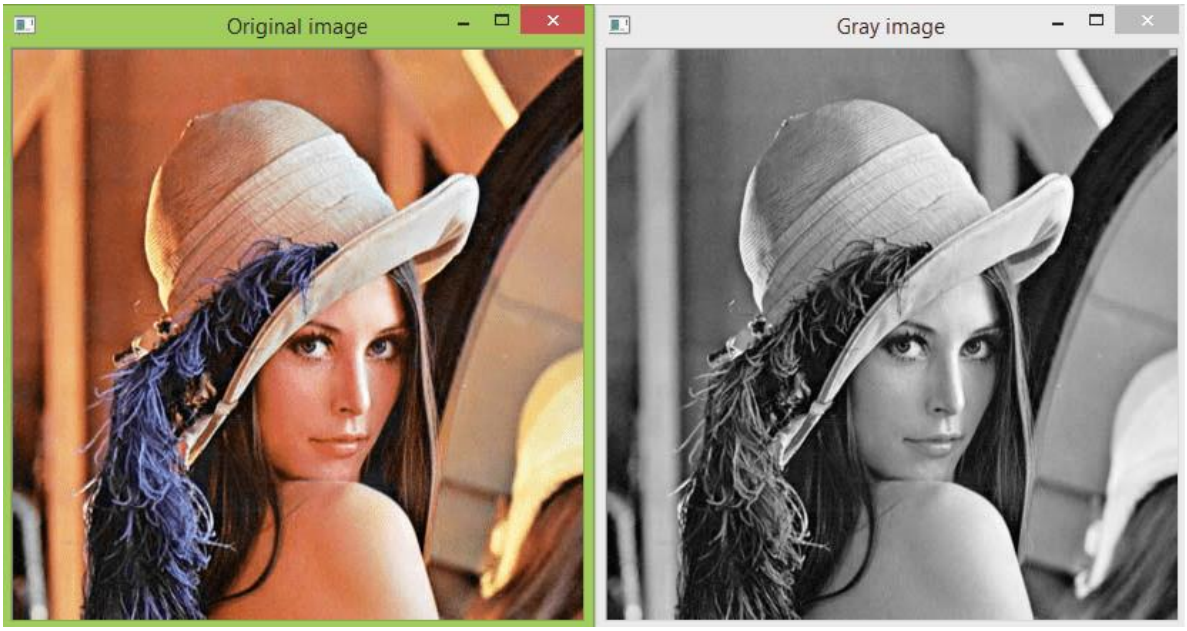
1.2.1 Konvertování obrazu do odstínu šedé

Používá se v případech, kdy barva není potřebná k rozpoznání a interpretaci obrazu. Jelikož barevné obrázky obsahují více informací (Barevné obrázky obsahují 3 barevné kanály) než černobílé obraz, mohou přidat zbytečnost složitost. [5]

Pro převod z barevného obrazu na černobílý obraz používáme vzorec [8]:

$$GRAY = 0.299R + 0.587G + 0.114B \tag{1}$$

Vážená metoda (1), neboli nazývaná také jako světelná metoda. Oči jsou nejcitlivější na zelené světlo, méně citlivé na červené světlo a nejméně citlivé na modré světlo. Důvodem toho mají barvy různou váhu. [9]



Obrázek 4 Grayscale [10]

1.2.2 Vyhlazování obrazu

Tato metoda má za cíl potlačit šum. Jeho nevýhodou je, že způsobuje rozmazání hran, které jsou důležité informací snímku. Vyhlazování obrazu využívá redundanci v obrazových datech k potlačení šumu, obvykle nějakou formou zprůměrování hodnot jasu v sousedních pixelech. Jelikož vyhlazování představuje problém rozostření ostrých hran, které jsou důležité pro detekci hran. Proto budeme uvažovat o metodách, které nám uchovají detaily hran. [1]

Konvoluční maska pro, která se používá pro oblast 3 x 3 [1]:

$$h = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

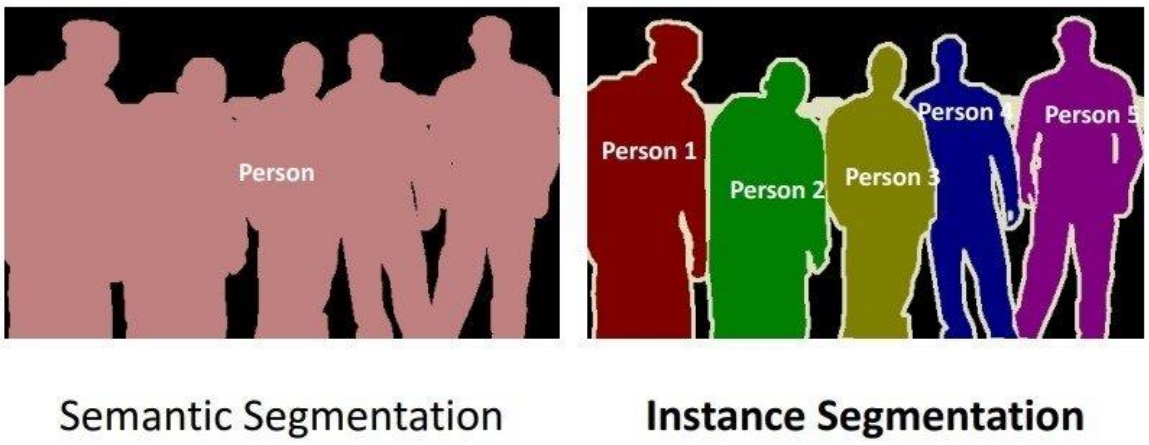
1.2.3 Standardizace obrazu

U některých algoritmů je potřeba změnit velikost snímků na jednotnou velikost. To znamená, že snímky musí být přepracovány a zmenšeny tak, aby měly identickou výšku a šířku, než je může zpracovat algoritmus.

1.3 Segmentace

Aby se nemusel zpracovávat celý snímek současně, můžeme snímek rozdělit na jednotlivé části nazývané segmenty. Takto rozdělíme snímek na oblasti s důležitými informacemi a na oblasti se žádnou užitečnou informací. Pomocí segmentace seskupujeme pixely, které mají podobné atributy. Každý objekt tak můžeme rozdělit ohraničujícím okrajem. Segmentace vytváří masku po pixelech pro každý objekt, který dokáže popsat tvar objektu. Nejednoznačné data můžou komplikovat procesu segmentace, a proto je třeba v nejčastějším případě si dávat pozor na vyskytující se šum. [11]

Segmentaci dělíme na dva typy, sémantickou segmentaci a segmentaci instance. Sémantická segmentace klasifikuje pixely do jedné sady kategorií bez ohledu na instance objektu. Používá se k naleznutí podobných objektů. Segmentaci instance klasifikuje pixely do každé instance objektů, identifikuje objekty z jedné třídy jednotlivě. [11]



Obrázek 5 Segmentace [11]

1.4 Klasifikace

Klasifikace předpovídá označení (kategorizuje) pro něco, co se nachází na obrázku. Aby mohl klasifikátor předpovídat příslušnou třídu objektu, musí být nejprve trénovaný. Je trénován na velkém datasetu obrázků s objektem v různých úhlech a za různých podmínek.

Jeden z možných způsobů klasifikace je využití vhodného algoritmu, např. histogram orientovaných gradientů, který nám navrhne mapu příznaků a ty pak poskytne klasifikátoru (support vector machines - SVM) pro klasifikaci objektu. Další možností je využití Konvoluční neuronové sítě (CNN) [12]

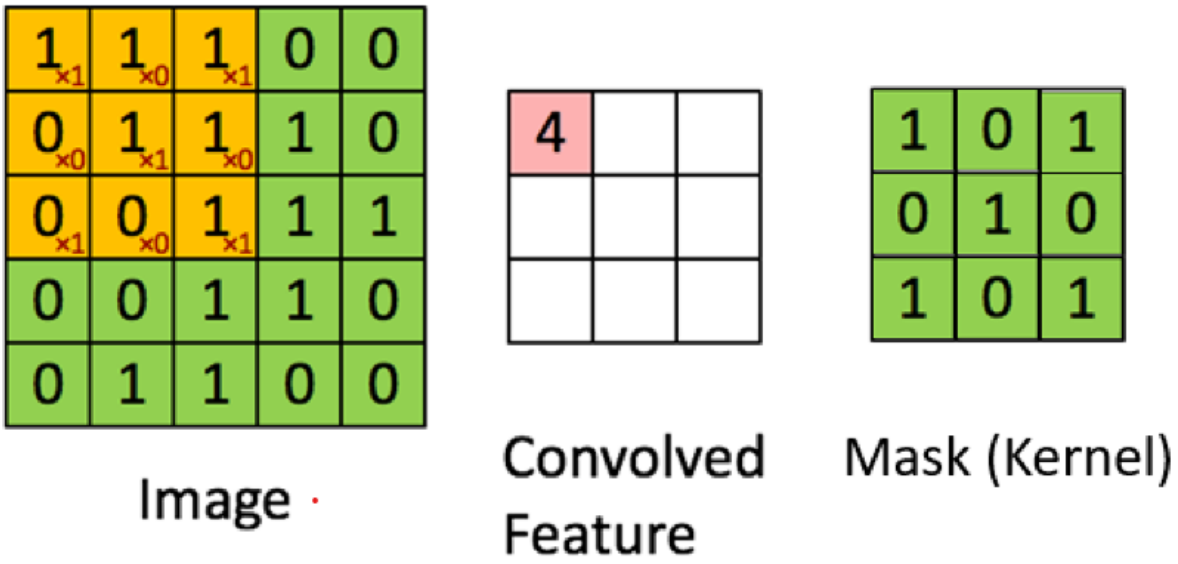
1.4.1 CNN – Konvoluční Neuronové Sítě

Jedná se o algoritmus hlubokého učení, který dokáže přijmout obrázek a zpracovat ho. Požadované předpracování je mnohem nižší ve srovnání s jinými klasifikačními algoritmy. Zatímco u primitivních metod jsou příznaky vytvořeny ručně, s dostatečným učením konvoluční neuronové sítě mají schopnost se tyto příznaky naučit a rozpoznat. Úlohou konvoluční neuronové sítě je zmenšit obrázky do podoby, která je snazší na zpracování, aniž by došlo ke ztrátě kritických informací. [13]

Architektura konvoluční neuronové sítě se skládá ze tří hlavních vrstev [13]:

- Konvoluční vrstva
- Sdružovací vrstva
- Plně propojená vrstva.

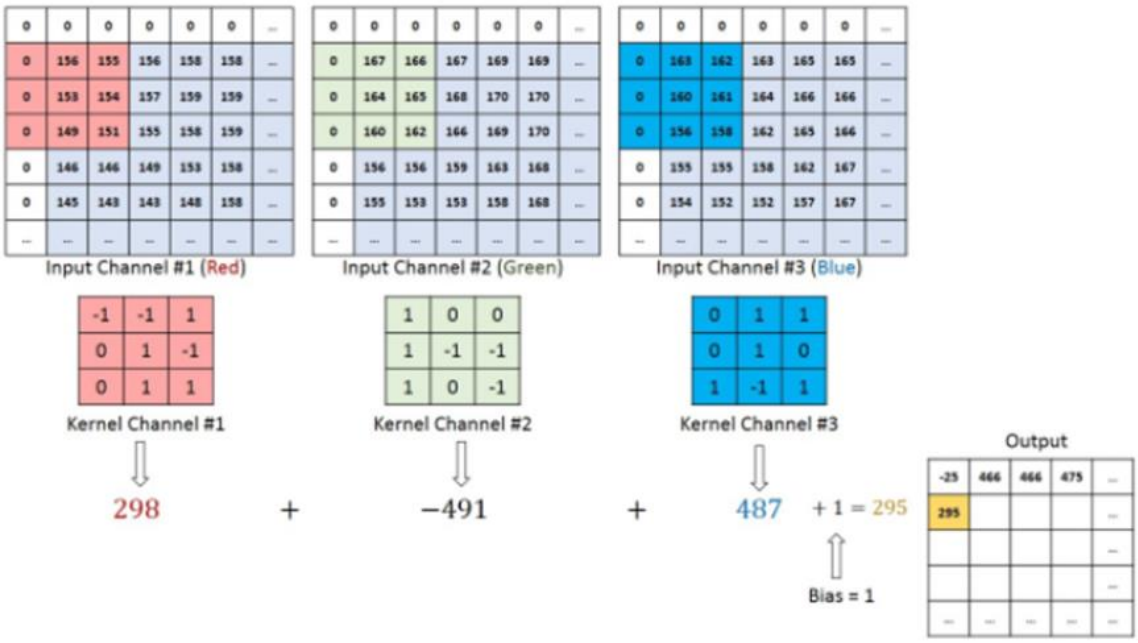
1.4.1.1 Konvoluční vrstva



Obrázek 6 Konvoluční vrstva [13]

Tato vrstva aplikuje masku na vstupní obrázek. Velikost konvoluční masky si můžeme zvolit na velikost např 3x3x1. Maska se posouvá po kroku o délce 1 a při každém kroku se provede násobení jednotlivých pixelů s hodnotami masky. Maska se, ale nemusí posouvat po kroku

o délce 1, ale i o větších délkách, dojde tak ke zmenšení obrázku. Pohybuje se doprava s určitou hodnotou kroku, dokud nenarazí na konec a poté se posouvá dolů se stejnou hodnotou kroku a opakuje proces, dokud neprojede celý obrázek. [13]

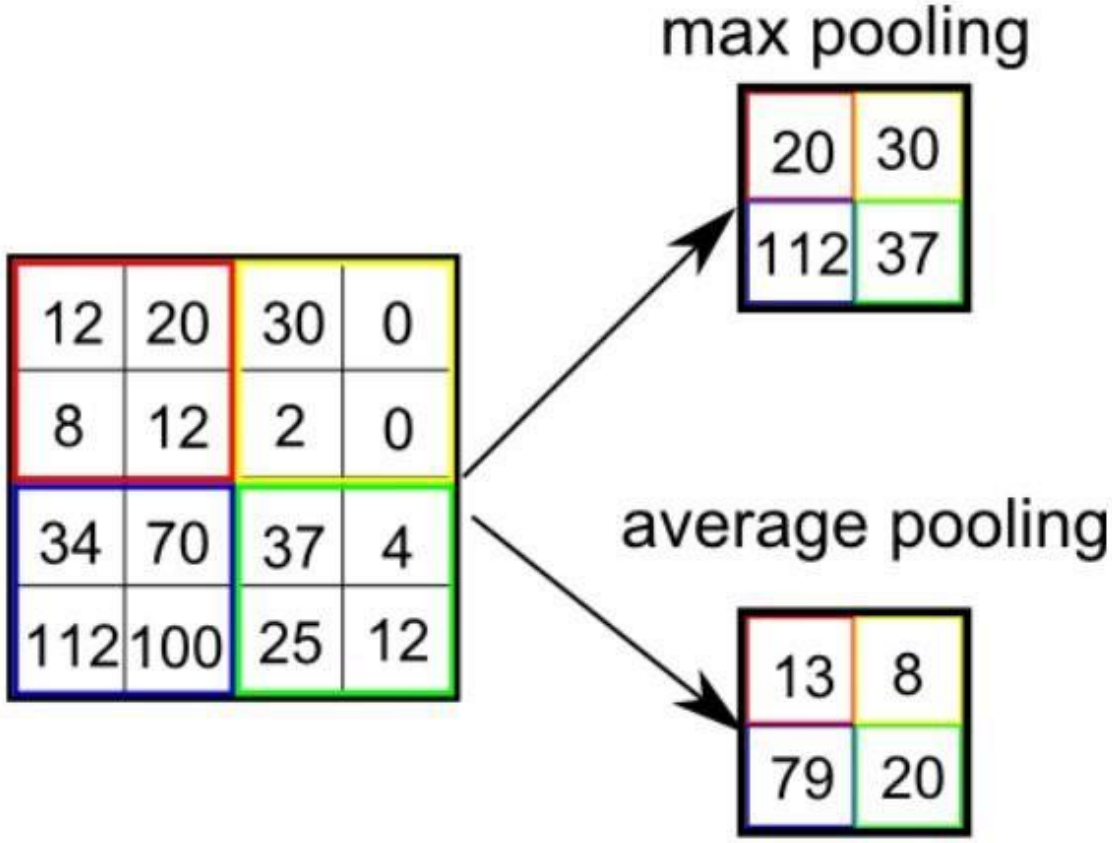


Obrázek 7 Konvoluční maska 3x3x1

Maska 3x3x3 se například používá, pokud máme tři barevné kanály pro každou základní barvu (RGB). Maska se posouvá mezi jednotlivými kanály a jejich výsledky se sečítají. Je, ale nutné přičíst hodnotu prahu, tudíž ještě přičíst jedna. [13]

1.4.1.2 Sdružovací vrstva (pooling)

Sdružovací vrstva je zodpovědná za redukování prostorové velikosti konvolučního příznaku. Sníží se tím potřebný výpočetní výkon ke zpracování dat. Je také užitečný pro extrakci příznaků, které jsou rotačně a pozičně neproměnné, čímž se udržuje proces efektivního trénování modelu. [13]



Obrázek 8 Sdružovací vrstva [13]

Jsou dva typy sdružování: Max pooling a Average pooling. Max pooling vrací nejvyšší hodnotu z části obrázku pokrytá maskou. Average pooling vrací průměr všech hodnot z části obrázku pokrytá maskou. [13]

1.4.1.3 Plně propojená vrstva.

Spojuje každý neuron v jedné vrstvě s každým neuronem v jiné vrstvě. Následuje za sdružovací vrstvou a transformuje její výstupní mapu na vektor. [13]

2 OPENCV

OpenCV (Open Source Computer Vision Library) je open source knihovna pro zpracování obrazu, počítačového vidění a strojového učení. Jedná se o obrovskou knihovnu zahrnující komplexní sadu klasických i nejmodernějších algoritmů počítačového vidění a strojového učení. Je multiplatformní, podporuje Windows, Linux a Mac OS a rozhraní pro C++ (primární rozhraní OpenCV), Python, Java a MATLAB. OpenCV byla původně výzkumná iniciativa společnosti Intel pro náročné výpočty zatěžující procesor (raytracing a trojrozměrné zobrazování stěn) a byl oficiálně spuštěn v roce 1999. Hlavní bodem vize tohoto projektu byli:

- Sestavit volný, optimalizovaný a znovu použitelný zdrojový kód, který může běžet v reálném čase.
- Poskytnutí nějaké společné infrastruktury, která je uživateli lehce čitelná a transparentní. Což umožnuje uživatelů přispět k vývoji knihovny.
- Poskytnutí pro komerční účely, to je, aby byla výkonně optimalizovaná a volně dostupná.

Knihovna je využívaná v mnoha oborech jako je rozpoznávání a identifikace objektů, rozpoznání obličejů, detekce pohybu, extrahování třídimenzionálních objektů, rozšířená realita a další. Pracují s ní světoznámé firmy jako například Google, Microsoft, Intel, IBM a Toyota. [14] [15]

Obrázek 9 logo OpenCV [14]

3 ALGORITMY PRO DETEKCI

Detekce objektů je technologie, která v sobě zahrnuje počítačové vidění a zpracovávání obrazu. Používá se k detekci instancí vizuálních objektů určité třídy. Poskytují základní informace potřebné pro počítačové vidění. Detekce identifikuje a lokalizuje objekty, které se nachází na snímcích. [16]

V dnešní době je detekce používána v různých odvětvích jako je rozpoznávání obličejů, samořídící se auta, počítání objektů a další. [16]

Typy algoritmů detekce [16]:

- Ne neurální přístupy
    - Viola-Jones
- Neuronové sítě
    - Dvoufázové algoritmy
        - Návrh regionů (R-CNN, Fast R-CNN, faster R-CNN, Mask R-CNN)
    - Jednofázové algorimy
        - Single Shot Multibox Detektor (SSD)
        - You Only Look Once (YOLO)

3.1 Viola Jones

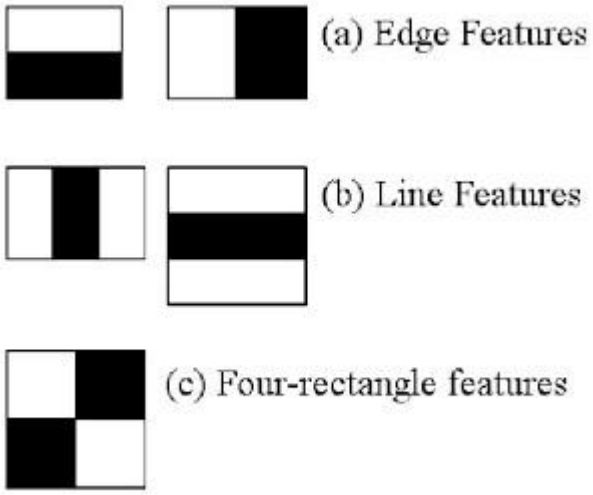
Vyvinutý Paulem Violou a Michaelem Jonesem v roce 2001. Umím rychle a přesně detekovat objekty ve snímcích, zejména obličeje. Tento systém detekce byl poprvé představen na procesoru 700 MHz Intel Pentium 3, a dokázal detekovat rychlostí 15 snímku za sekundu na snímcích o velikosti 384 x 288 pixelů, což bylo na tuto dobu naprosto nevídané. Navzdory svému věku je stále velice využívaný, a kromě toho je implementovaný v knihovně OpenCV. Algoritmus funguje na principu, že snímkem je posouván box a hledá v rámečku obličej. Hledá v něm rysy, které budou vysvětleny v následujících podkapitolách. K detekci používá následující koncepty Haarovy příznaky, Integrální obraz, AdaBoost algoritmus a Kaskádový Klasifikátor. [17][18][19]

3.1.1 Haarovy příznaky

Jsou to pole neboli kernely se světlou a tmavou oblastí, pomáhají nám určovat základní rysy objektů. Například víme, že oči mají tmavší intenzitu barev než okolí, nebo prostřední část je světlejší než okolí, což je nos. [17][18][19]

Viola Jones využívá tři typy kernelů k detekování rysů:

- Okrajové rysy (Edge features)
- Liniové rysy (Line features)
- Čtyřstranné rysy (Four-rectangle features)



Obrázek 10 Haarovy filtry [19]

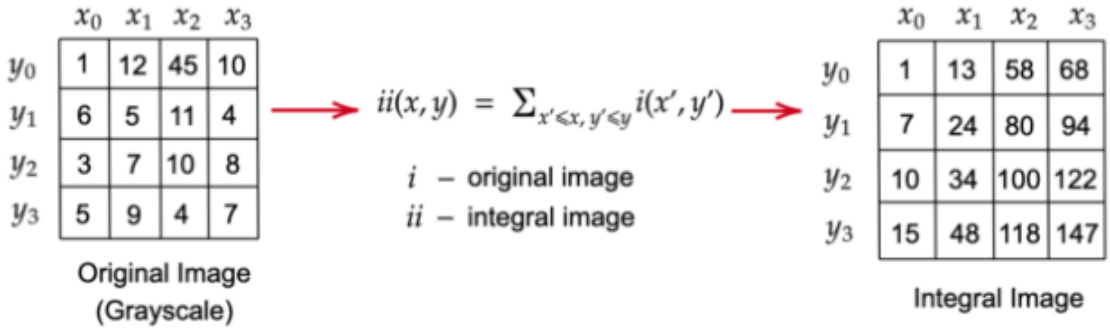
a) Okrajové rysy: Kernel obsahuje dva obdélníky vedle sebe. Obdélníky mají stejnou velikost a jsou vůči sobě svisle nebo vodorovně. Funkce je vypočítání rozdílu mezi součtem pixelů dvou obdélníků.

b) Liniové rysy: Kernel obsahuje tři obdélníky. Funkce je součet pixelů dvou vnějších obdélníků odečtených od součtu pixelů vnitřního obdélníku.

c) Čtyřstranné rysy: Tento kernel vypočítá rozdíl mezi diagonálními trojúhelníky.

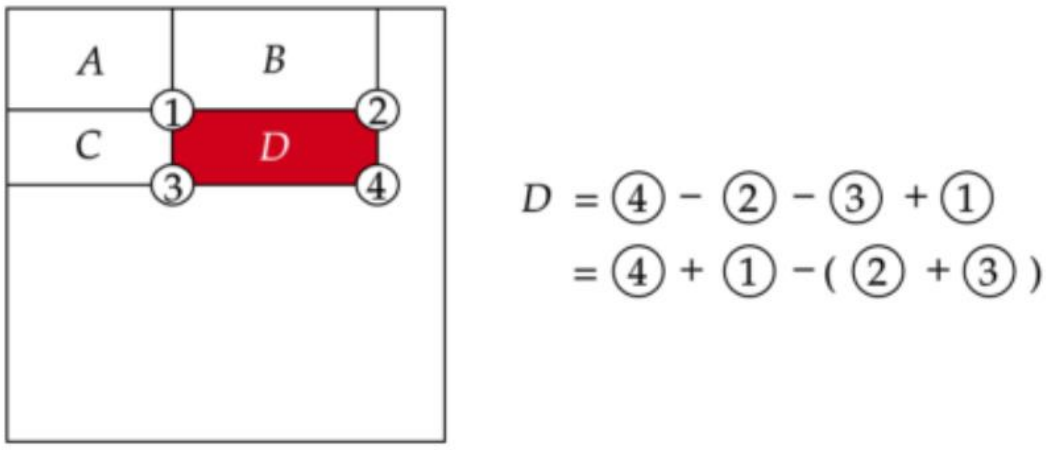
3.1.2 Integrální obraz

Jelikož musíme kernely vypočítat ve všech různých velikostech a oblastech, nastává problém s výpočetní náročností. Průměr v daných oblastech musíme vypočítat vícekrát, což nám výpočetní náročnost ještě zhoršuje. Abychom urychlili výpočet použijeme integrální obrazový přístup. [17][18][19]



Obrázek 11 Integrální obraz [18]

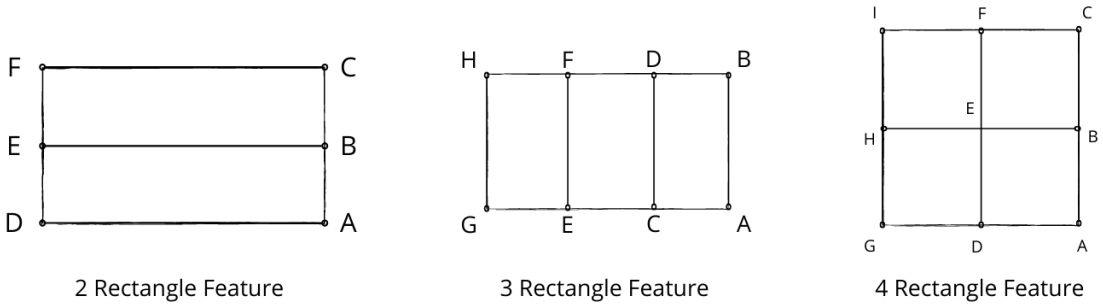
Integrální obraz je přechodná reprezentace obrázku, kde hodnota pro umístění (x, y) na integrálním obrázku se rovná součtu pixelů nad a nalevo (včetně svého umístění) od (x, y) umístění na původním obrázku (viz Obrázek nahoře). [17][18][19]



Obrázek 12 Součet hodnot pixelů v dané oblasti [18]

Abychom nemuseli procházet všemi pixely v dané oblasti, pomocí hodnot z integrálního obrazu lze vypočítat součet hodnot z pixelů (viz funkci na obrázku). To nám značně zjednodušuje časovou náročnost daného úkolu. [17][18][19]

3.1.3 Využití integrálního obrazu při výpočtu Haarových příznaků



Obrázek 13 Využití integrálního obrazu při výpočtu Haarových prvků [19]

- Okrajové rysy (Edge features)

$$2RF = A - 2B + C - D + 2E - F \tag{3}$$

- Liniové rysy (Line features)

$$3RF = A - B - 2C + 2D + 2E - 2F - G + H \tag{4}$$

- Čtyřstranné rysy (Four-rectangle features)

$$4RF = A - 2B + C - 2D + 4E - 2F + H - 2I + J \tag{5}$$

Jak můžeme vidět z předchozího obrázku a funkcí, že dosáhneme konstantního časového vyhodnocení výsledků bez ohledu na velikost oblasti. [17][18][19]

3.1.4 AdaBoost algoritmus

Algoritmus je schopen určit falešná pozitiva nebo true negativa, učí se z obrázků, které mu dodáme. Umožnuje přesnějšímu detekování objektů. Trénování algoritmu je velmi náročné, protože je potřeba kontrolovat každý obrázek, který mu dodáme. [17][18][19]

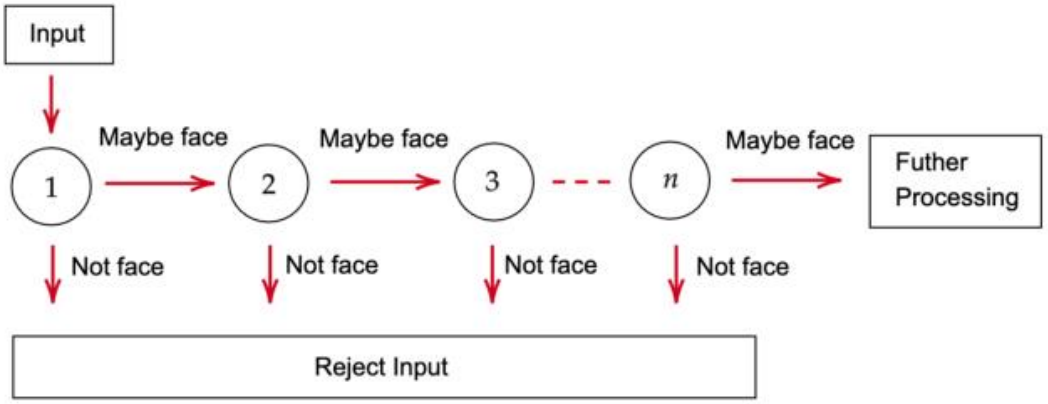
$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Obrázek 14 Adaboost, silné a slabé klasifikátory [17]

Máme rovnici, která nám určuje míru úspěšnosti, f_1,f_2 a f_3 jako příznaky a α_1, α_2 a α_3 jako váhy příslušných prvků. Každá z funkcí (f_x) jsou známé jako slabý klasifikátor. Jelikož slabý klasifikátor nemusí být tak dobrý, máme tady silný klasifikátor F(x). Protože jeden slabý klasifikátor není dostačující, kombinací dvou nebo více slabých klasifikátorů získáme silný klasifikátor. Jak stále přidáváme slabé klasifikátory, získáváme silnější a silnější silný klasifikátor. Tomu se říká soubor (ensemble). Jak se ale ujistíme, aby ty „nejlepší“ příznaky byli vepředu? Zde vstupuje Adaptive Boosting. Řekněme, že máme 10 obrázku, a 5 obrázků má obličej a 5 nemá. Model nám dá tři true positiva a dvě true negativa. Na dvou obrázcích má tedy falešná negativa, i když obsahují obličeje a třech obrázcích falešná pozitiva, i když obličeje neobsahují. V dalším kroku AdaBoost využije další příznak, ale ten, který nám bude nejlépe komplimentovat našemu nejlepšímu příznaku. Zvyšuje tedy důležitost obrázků, které byly chybně označeny jako falešně negativní, navyšuje váhu těchto obrázků v celkovém algoritmu. S přibývajícími novými funkcemi bychom se tedy na konci dostali k jednomu obrázku, kterému by byla přikládána vyšší váha. [17][18][19]

3.1.5 Kaskádový Klasifikátor

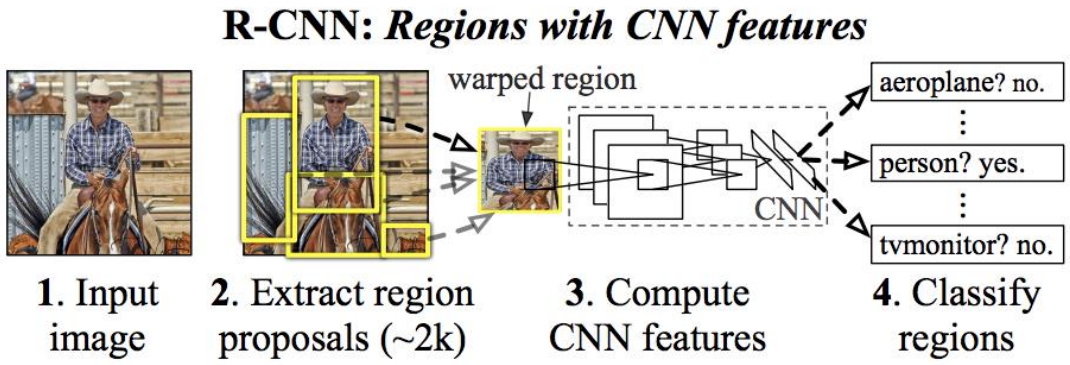
Kaskádový klasifikátor je vícestupňový klasifikátor, který může provádět detekci rychle a přesně. Každá fáze se skládá ze silného klasifikátoru vytvořeného algoritmem AdaBoost. Vstup je vyhodnocen, pokud klasifikátor vydá negativní výsledek, vstup je okamžitě odmítnut a ukončí se výpočet. Pokud klasifikátor vydá pozitivní výsledek, vstup se pošle do dalšího stupně. S každou úrovní se zvyšuje počet slabých klasifikátorů. [17][18][19]



Obrázek 15 Kaskádový klasifikátor [18]

3.2 RCNN

R-CNN je zkratka pro Konvoluční neuronové sítě založené na regionech (Region Based Convolutional Neural Networks), který je navržený speciálně pro detekci objektů. Vstupní obrázek prochází algoritmem nazývaným selektivní vyhledávání k extrahováním zhruba dvou tisíců oblastí z obrázku. Tak zvané oblasti zájmu, které můžou být reprezentovány vykresleným obdélníkem poté prochází algoritmem konvoluční neuronové sítě (CNN), ten nám vytvoří příznaky. Tyto příznaky poté projdou algoritmem SVM (support vector machine), a klasifikuje dané objekty v oblasti zájmu. [20]



Obrázek 16 R-CNN [21]

3.2.1.1 Selektivní vyhledávání

Využívá se přístup nazývaný „vyčerpávající vyhledávání“, používají se posuvné filtry různých velikostí k extrahování příznaků z obrázku. Se zvyšujícím se počtem filtrů se zvýší výpočetní náročnost. Vyčerpávající vyhledávání nepracuje samotně, ale také se segmentací

barev prezentovaných na obrázku. Vygeneruje se mnoho regionů oddělené barvami. Selektivní vyhledávání pokračuje s následujícím algoritmem nazývaný jako chamtivý algoritmus, ten vyhledá podobné barvy v oblastech a ty je pak spojí. [21]

Výpočet podobnosti mezi regiony lze vypočítat takto[21]:

$$S(a,b) = Stexture(a,b) + Ssize(a,b)$$

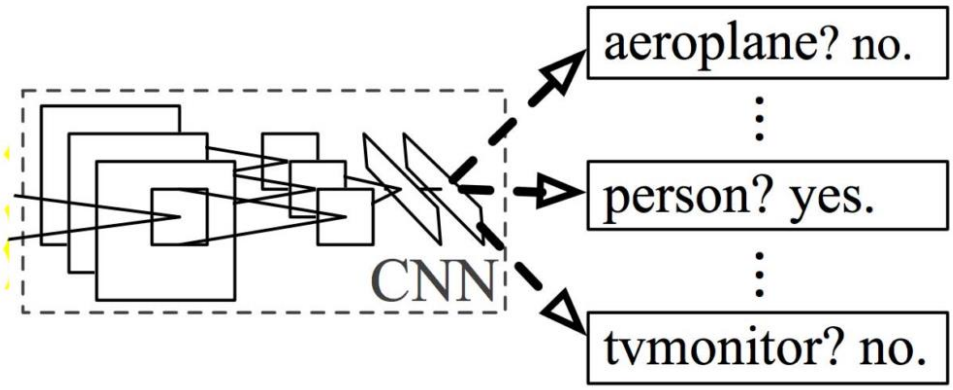
Kde Stexture(a,b) je vizuální podobnost a Ssize(a,b) je podobnost mezi oblastmi.

Obrázek 17 Selektivní vyhledávání [21]

Na obrázku můžeme vidět, jak se oblasti zvětšují s zvyšující podobností.

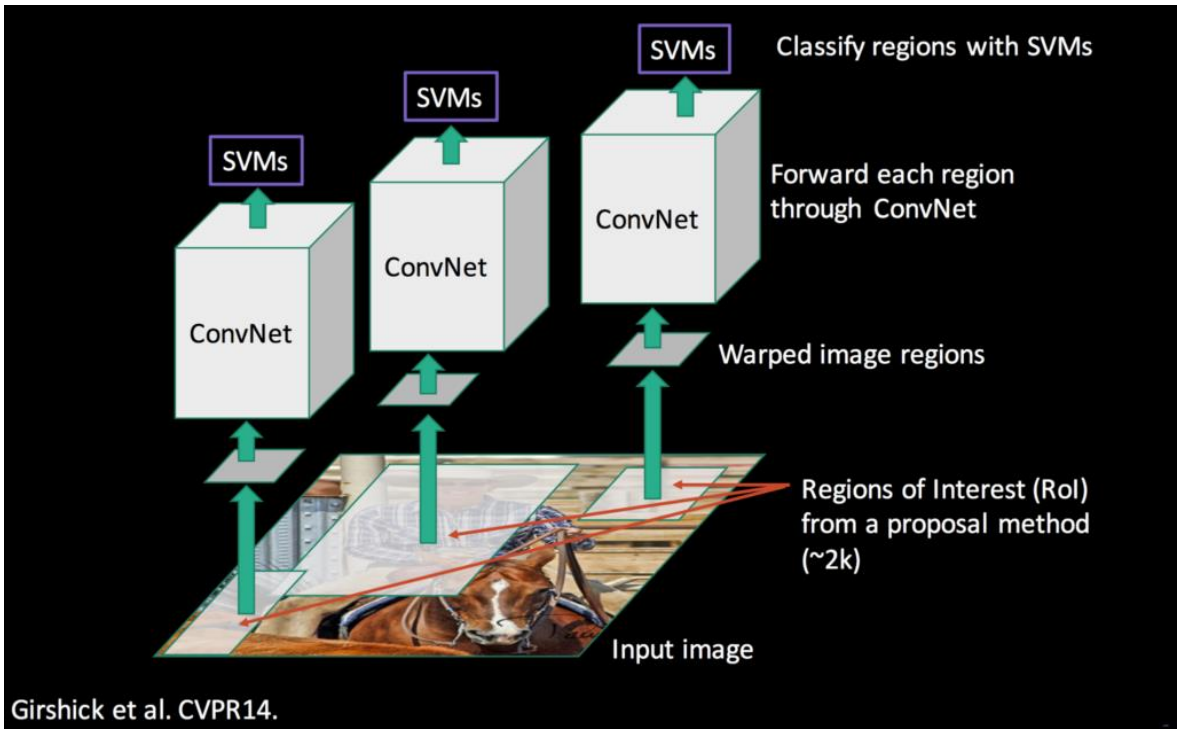
3.2.1.2 Klasifikace

Po výběru regionů obraz prochází konvoluční neuronovou sítí (CNN), kde model CNN extrahuje příznaky z regionu, jako výstup vytváří 4096rozměrný příznakový vektor. Ty jsou poté přiváděny do SVM pro klasifikaci přítomnosti objektu. Každý SVM navržen tak, aby se klasifikoval pro jednu třídu objektů (člověk, auto, atd.). Každý SVM vygeneruje skóre pro danou třídu, udávající pravděpodobnost, že návrh regionu tuto třídu obsahuje. [20] [21]



Obrázek 18 Klasifikace [21]

Celý R-CNN proces může být reprezentován následujícím obrázkem.

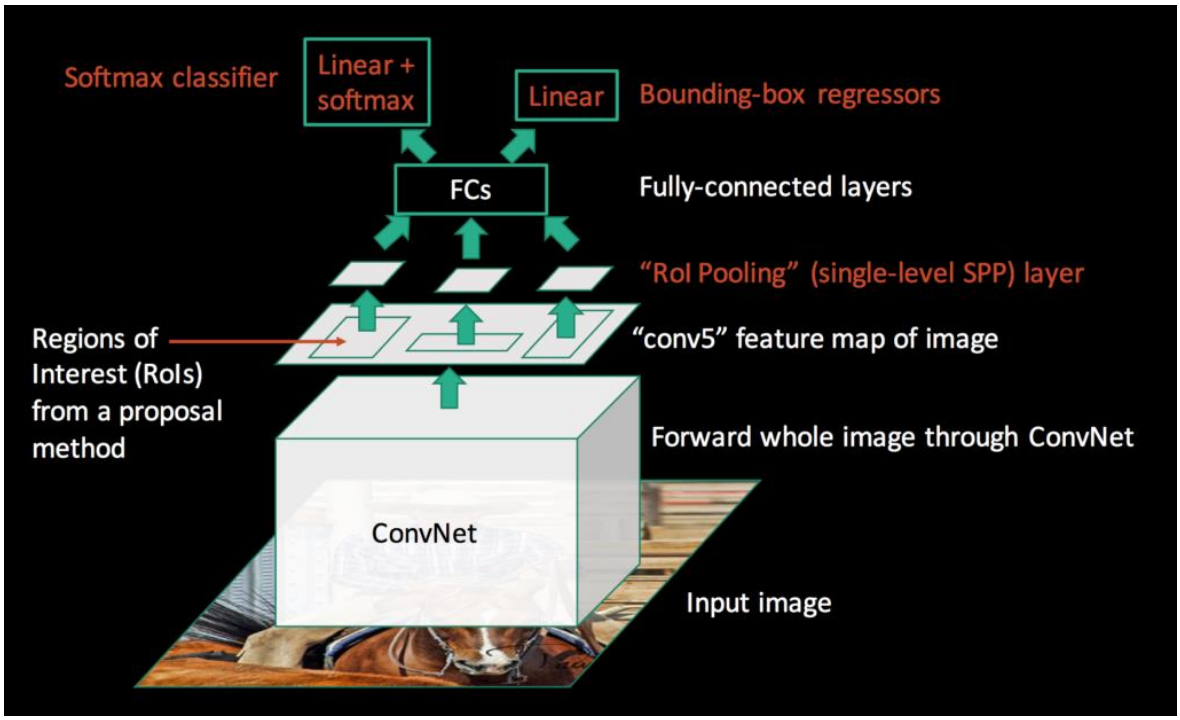


Obrázek 19 Proces R-CNN [21]

3.2.2 Fast R-CNN

Algoritmus R-CNN měl značné nevýhody. Trénování zabíralo značně velké množství času, protože byste museli klasifikovat 2000 oblastí zájmu. Nešlo jej implementovat v reálném čase. Algoritmus selektivního vyhledávání je pevný algoritmus. Proto v této fázi neprobíhá žádné učení. To vedlo ke generování špatných návrhů. [20] [21]

Vznikl proto algoritmus Fast R-CNN, který řeší některé nevýhody R-CNN. Namísto toho abychom vytvářeli návrh oblastí, které bychom dodávali do CNN, tak posíláme vstupní obrázek rovnou do CNN, ten nám vygeneruje konvoluční mapu příznaků. Z mapy konvolučních příznaků identifikujeme oblasti zájmu. Ty jsou přiváděny do vrstvy ROI pooling. Vrstva ROI pooling funguje tak, že každý návrh regionu rozděluje na matici buněk. Poté operace maximálního sdružování je aplikována na každou buňku v matici, aby se vrátila jedna hodnota. [20] [21]

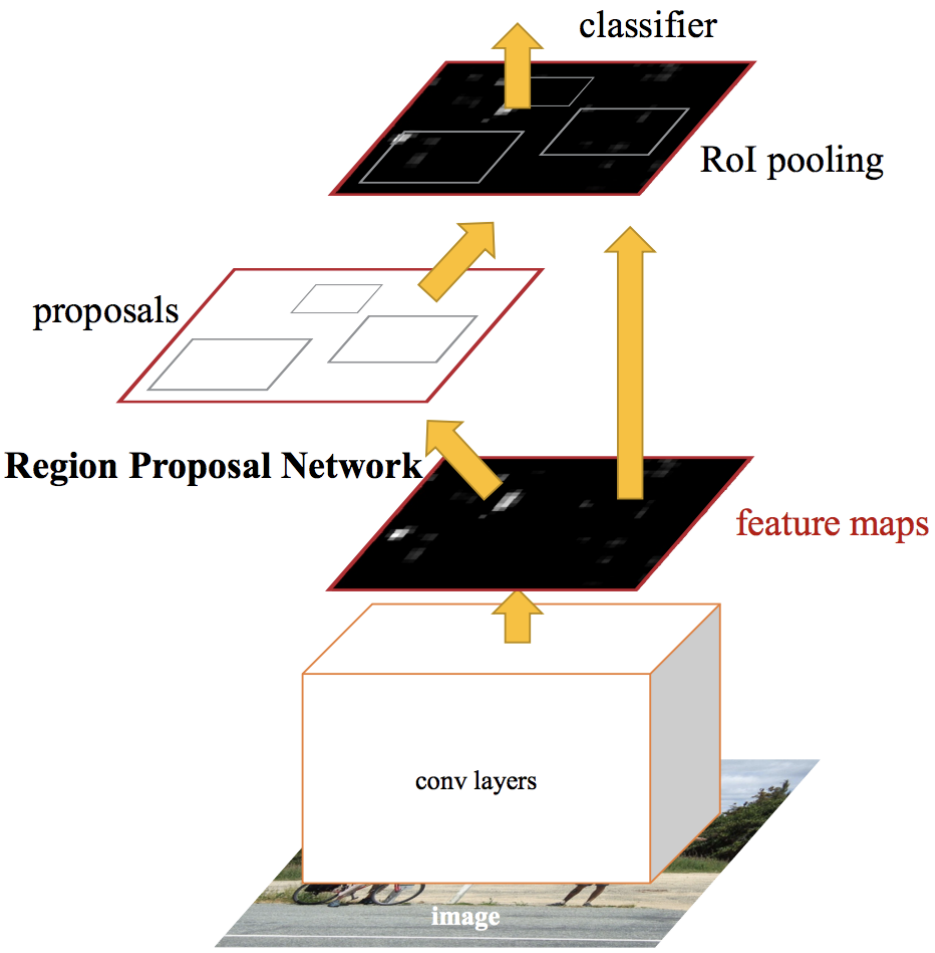


Obrázek 20 Proces Fast R-CNN [21]

Ke konci předáme to plně propojené vrstvě a využije se Softmax a lineární regrese pro klasifikaci prvků. Lineární regrese dále upřesňuje ohraničovací rámeček. [20] [21]

3.2.3 Faster R-CNN

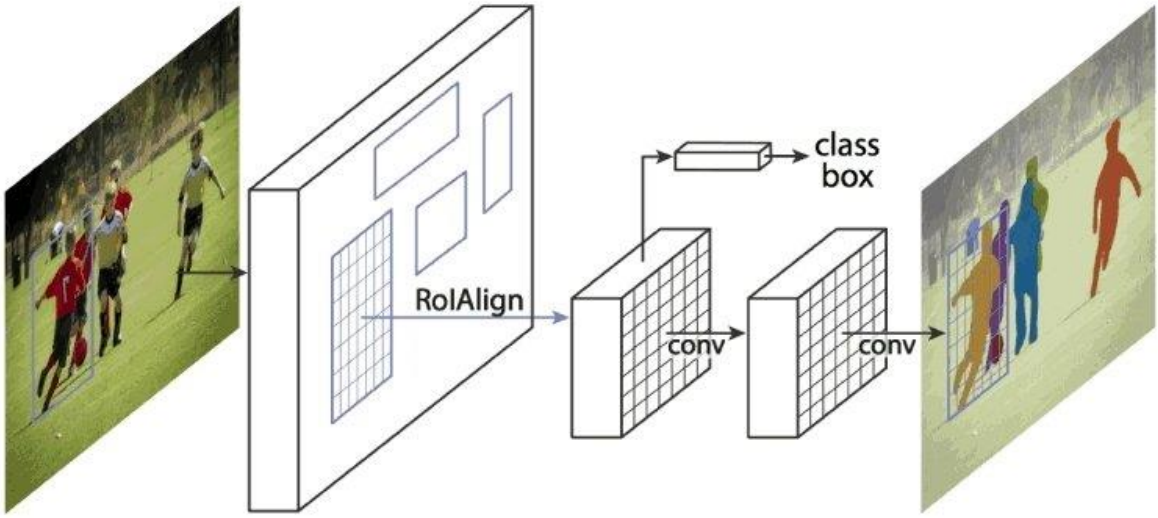
Jako u předchozího algoritmu vstupní obrázek je poslán přímo do CNN, který vygeneruje konvoluční mapu příznaků. Namísto použití selektivního vyhledávacího algoritmu na mapě příznaků k identifikaci oblasti zájmu, algoritmus používá RPN (regional proposal network), který nám vygeneruje návrhy oblastí. Oblasti poté přivádíme do vrstvy ROI pooling k dalšímu zpracování. [20] [21]



Obrázek 21 Proces Faster R-CNN [20]

3.2.4 Mask R-CNN

Mask R-CNN je Faster R-CNN doplněný o třetí výstup, přidává větev pro predikci masky objektu. Příznakové vektory jsou přiváděny do klasifikátoru masky, který má v sobě dvě vrstvy CNN, a ty generují třídu a masku pro každý objekt bez konkurence mezi třídami. [22]

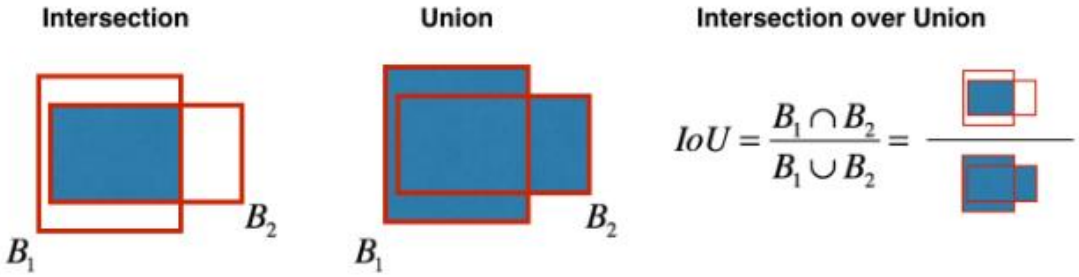


Obrázek 22 Proces Mask R-CNN [22]

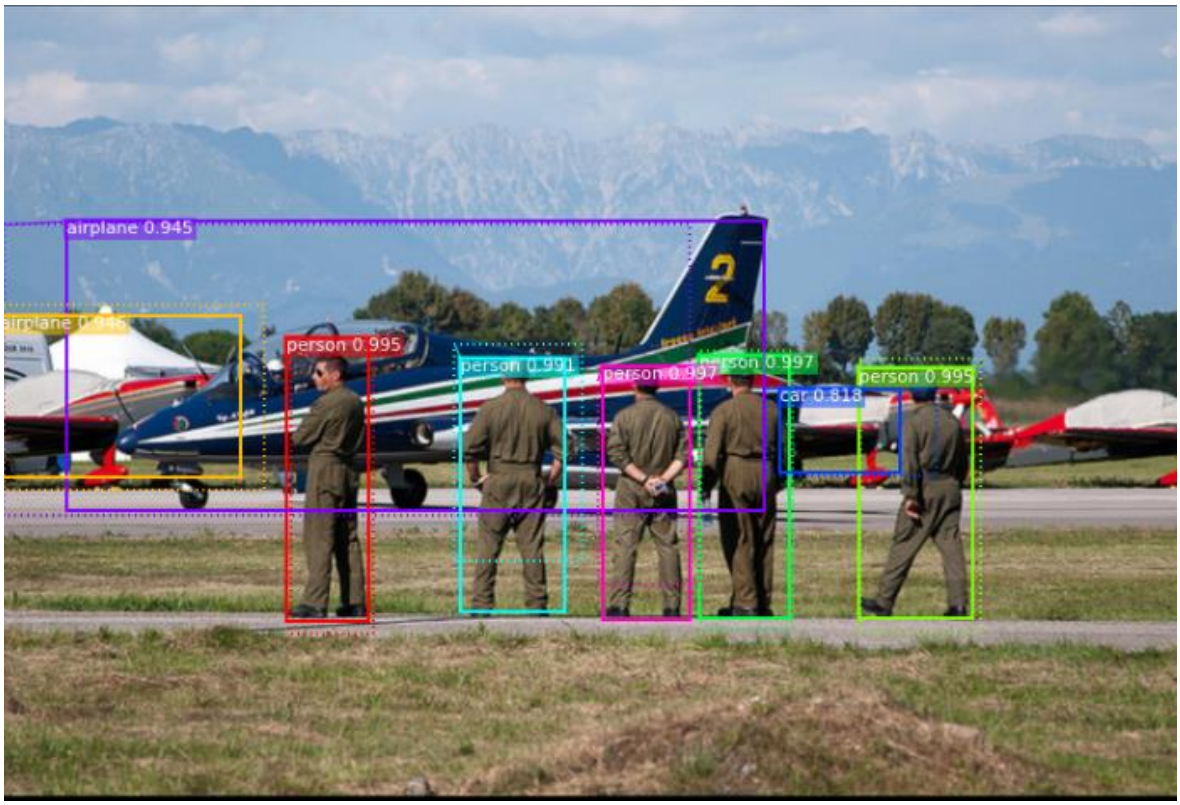
Mask R-CNN používá tzv. kotevní rámce k detekci objektů. Kotevní rámce jsou předdefinované ohraničující rámce určité výšky a šířky. Tyto rámce jsou definovány tak aby zachytili měřítko a poměr stan objektů, které chceme detekovat. Generuje tisíce předpovědí, odstraní se rámce, které jsou v pozadí a na zbývajících rámcích se provádí filtrace podle skóre spolehlivosti. [22]

Obrázek 23 Vygenerované kotevní rámce

Dalším krokem je NMS (Non-max suppression) algoritmus, ten nám pomůže zbavit se nepotřebných rámců. Tento problém řeší tak, že vezme rámec s největší pravděpodobností třídou a vypočítá, jak se prolínají s ostatními rámce v okolí, metoda se nazývá IoU (Intersection over Union). Poté zamítne ohraničující rámce, jejichž hodnota IoU je menší než prahová hodnota (<0.5). [22]



Obrázek 24 Intersection over Union [23]

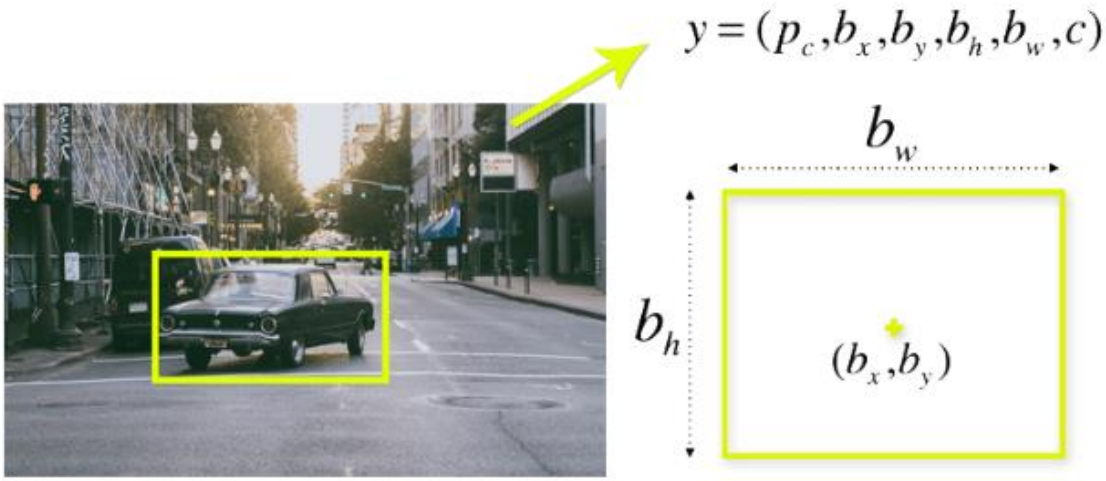


Obrázek 25 Konečný výstup [22]

3.3 YOLO

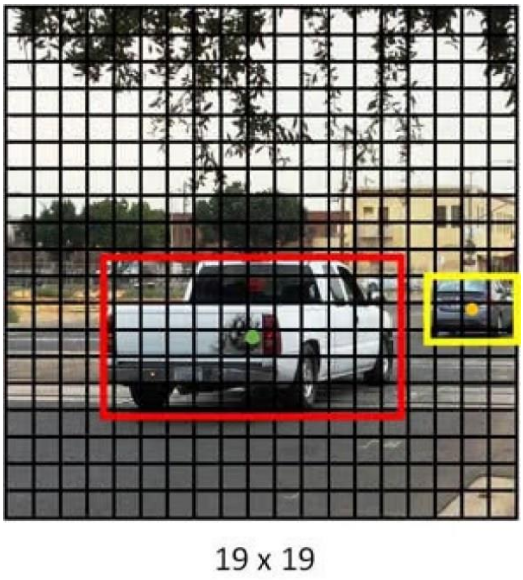
YOLO (you look only once) je algoritmus pro detekci objektů, který se liší od algoritmů založených na regionech, které jsou uvedeny výše. Algoritmus je založený na regresi, jen jedna konvoluční síť předpovídá ohraničující rámec a jejich klasifikaci. Nejprve musíme pochopit co, vlastně předpovídá. Každý ohraničující lze popsat pomocí těchto čtyř atribut [23]:

- Šířka (bw)
- Výška(bh)
- Střed ohraničujícího rámce (bx, by)
- Hodnota (c) odpovídající třídě objektu



Obrázek 26 Atributy ohraničujícího rámce [24]

Jak jsme si říkali YOLO nevyhledává oblasti zájmu. Vstupní obrázek rozdělí do mřížky o velikosti 19 x 19. Každá buňka v mřížce bude detekovat objekty, které se v ní objeví. Objekt leží v určité buňce pouze tehdy, kdy se v buňce objeví středové souřadnice ohraničujícího rámce. Díky této vlastnosti se středové souřadnice ohraničujícího rámce vždy počítají vzhledem k buňce, a výška a šířka se počítají vzhledem k celé velikosti obrázku. [23] [24] [25]

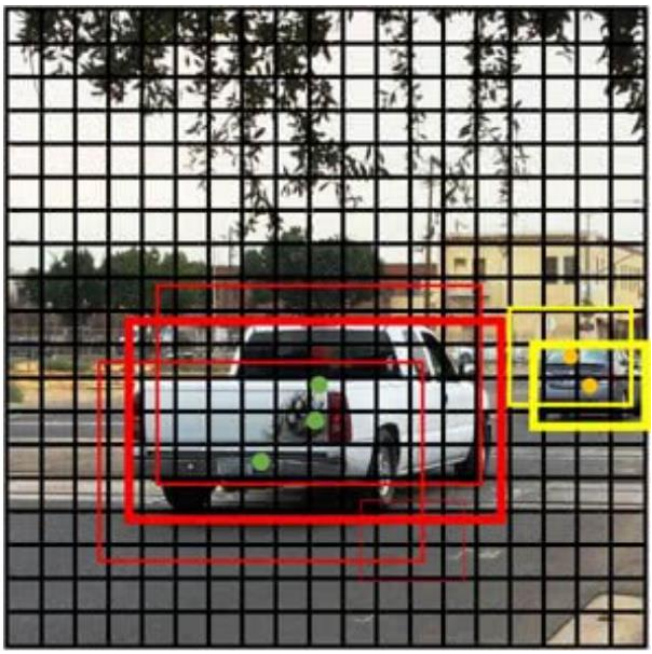


Obrázek 27 19x19 mřížka [25]

Rovnice, podle které YOLO určuje pravděpodobnost, že buňka obsahuje určitá třída [18]:

$$skore_{c,i} = p_c * c_i$$

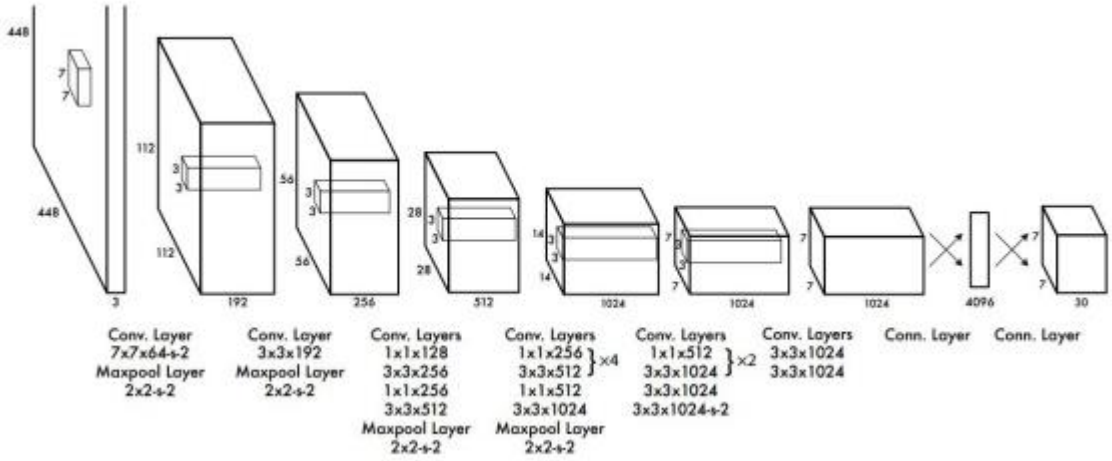
Kde p_c je pravděpodobnost, že v ohraničujícím rámci je nějaký objekt a c_i je pravděpodobnost, že existuje objekt určité třídy. Třída s největší pravděpodobností je poté přiřazena buňce. [23]



Obrázek 28 Vygenerované rámce [25]

Výstup algoritmu YOLO obsahuje mnoho ohraňujících rámců pro stejný objekt. Liší se tvarem a velikostí. Proto dalším krokem je NMS (Non-max suppression) algoritmus. [23] [24] [25]

Obrázek 29 proces NMS [23]



Obrázek 30 Architektura YOLO [23]

3.3.1 Yolov2

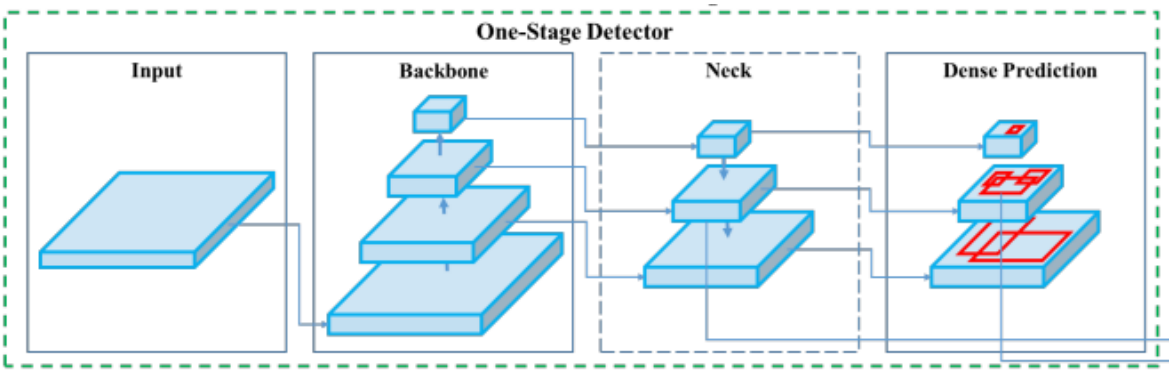
- Má 30 neurálních vrstev oproti 26
- Normalizace dávek po každé konvoluční vrstvě
- Zvýšení velikosti vstupních snímků 448-448 pixelů
- Odstranění plně propojených vrstev a používají se kotevní rámce (asi 1000 možných oblastí)
- Pro trénování se požívali obrázky různých hodnot od 320-680 pixelů

3.3.2 Yolov3

- 106 neurálních vrstev
- Detekce na třech různých měřítcích pro detekování objektů malých i extra velkých
- Devět kotevních rámců, tři na každé vrstvě
- Skóre k ohraničujícím rámcům

3.3.3 Yolov4

Algoritmus se skládá ze 3 částí Backbone, Neck a Head. Backbone funguje jako extraktor příznaků, skládá se především z modelů jako je VGG, ResNet, Darknet a další. Neck je další vrstva, která bere mapy příznaků z různých fází Backbonu. Neck používá tzv. Feature Pyramid Network (FPN), vytváří se pyramida příznaků v mnoha měřítcích. Head je část, která se stará o detekci a tvoří ohraničující rámce a pravděpodobnost tříd. [26]

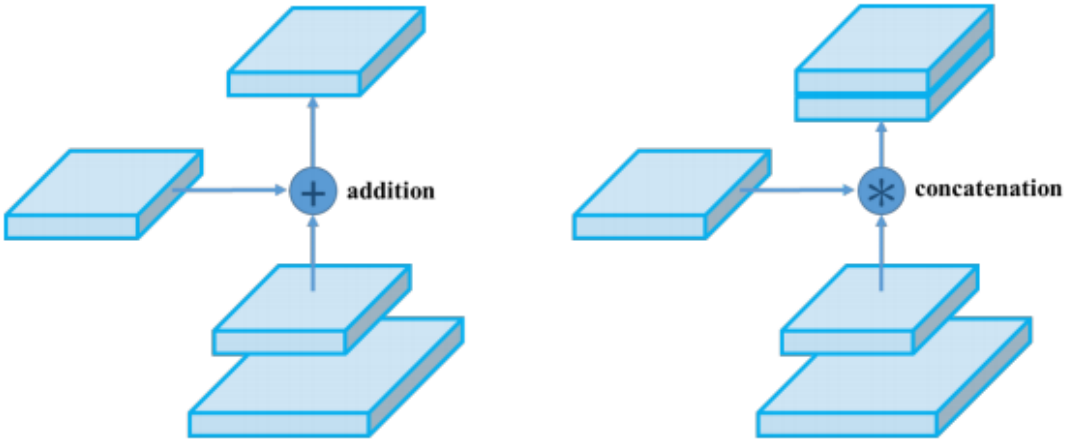


Obrázek 31 Proces YOLOv4 [26]

(BoF) zvyšuje schopnost zobecnění modelu. Provádí se zkreslení obrazu jako změna jasu, kontrastu, geometrické zkreslení atd. Dalším příkladem je zakrytí náhodných oblastí na obrázku. [26]

Bag of specials (BoS), zvyšuje náklady na přesnost detekce. Zahrnuje moduly pozornosti Squeeze-and-Excitation (SE) a Spatial Attention Module (SAM), jako SPP, ASPP a RFB. Yolov4 spíše využívá upravený SAM. V původní verzi se k mapě příznaků se prováděl max pooling a average pooling a poté se zřetězila. Poté se aplikovala konvoluční vrstva a vygenerovala pozornosti mapa. U modifikované verze nepoužívá max pooling a average pooling, místo toho se mapa příznaků pošle do konvoluční vrstvy a mapa se násobí. [26]

PANet je FPN, který využívá Yolov4. PAN je upravený, místo toho, aby mapy příznaků sčítal, tak je zřetězí. [26]



Obrázek 32 PANet a upravený PANet [26]

SPP provádí zvýšení receptivního pole oddělí nejdůležitější příznaky od backbonu.

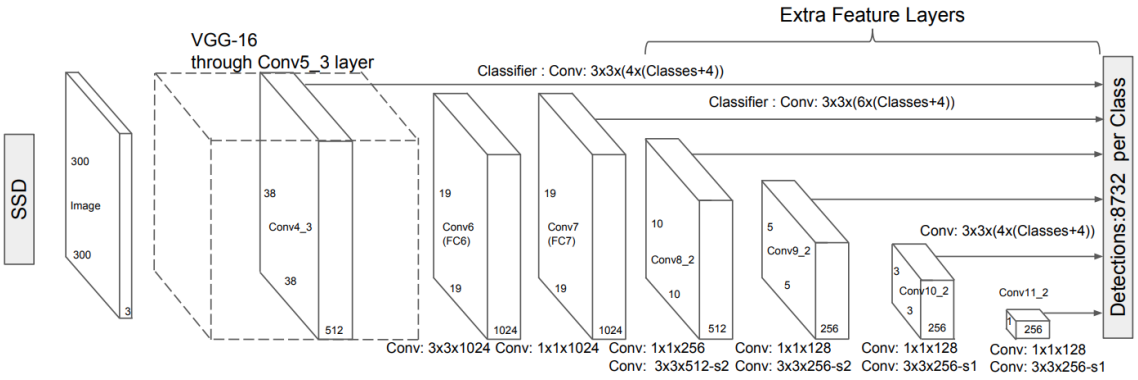
Yolov4 používá CSPDarkNet53 (Head), SSP + PANet (Neck), YOLOv3 (head).

3.3.4 Tiny Yolov4

- Má 29 konvolučních vrstev oproti 137 (Yolov4)
- Detekuje jen na dvou měřítcích
- Rychleji něž Yolov4
- Lepší pro real-time detekci

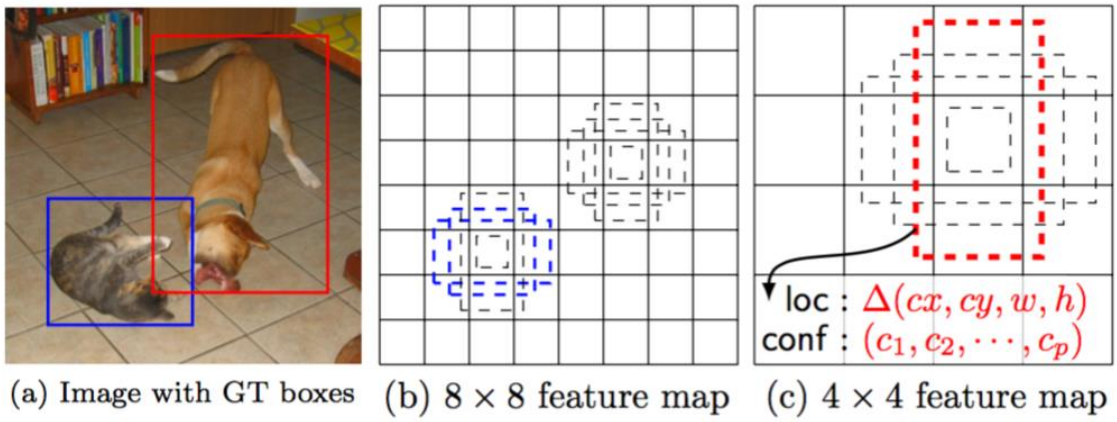
3.4 SSD (Single Shot Detector)

SSD využívá plně konvoluční přístup, ve kterém je síť schopna najít všechny objekty v obraze v jednom průchodu. Hlavním rozdílem tohoto algoritmu od ostatních je, že SSD používá architekturu VGG-16 a po nich jdou další konvoluční vrstvy. Po každém bloku konvolučních vrstev jsou generovány mapy příznaků různých velikostí, které umožňují detekovat malé i velké objekty. Modul starající se o detekci tak pracuje s různými vrstvami příznaků a má tak větší počet předpovědí. [27]



Obrázek 33 Architektura SSD [27]

Mapy příznaků rozdělíme do mřížek, kde každým buňkám je přiřazeno několik výchozích ohraničení různých aspektů odpovídající danému měřítku. Algoritmus ke konci provede NMS a IoU, které jsme si vysvětlili v předchozím algoritmu.



Obrázek 34 Generování map příznaků [27]

4 ALGORITMY PRO SLEDOVÁNÍ

Sledování je kombinace modelu pohybu a modelu vzhledu. Model pohybu sleduje směr a rychlost pohybu objektu, to mu umožnuje předpovídat novou polohu objektu. Model vzhledu se stará o učení algoritmu, jak objekt vypadá a jestli objekt se nachází v ohraničujícím rámci. Rozdíl mezi detekcí a sledováním je, že u detekování je to jednoduše identifikace a lokalizace objektů, mezitím co sledování je zaměření objektu a jeho sledování v sekvenci snímků. Sledování se nestará o identifikaci objektů, ale jen o samotné sledování pohybu objektů, což zjednodušuje komplexitu algoritmu a jeho výpočetní náročnost. Používání detekce k sledování je proto velice náročné a může sledovat jen jemu známe objekty. [28]

4.1 BOOSTING Tracker

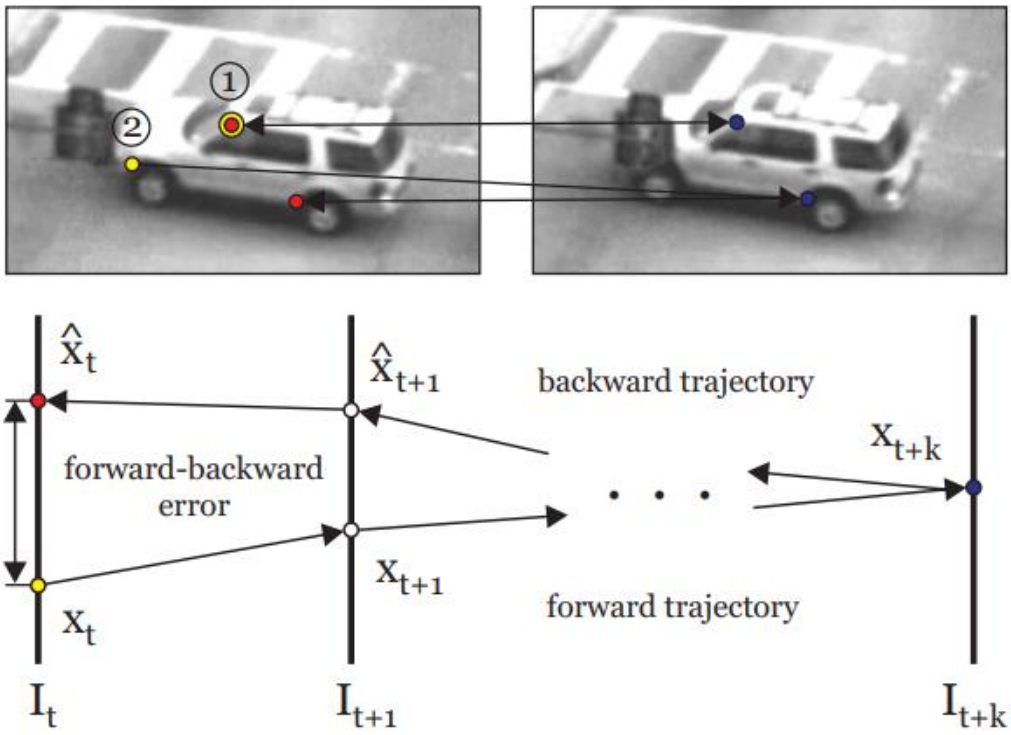
Princip tohoto sledovacího algoritmu je založen na online verzi AdaBoost algoritmu, který používá Viola Jones algoritmus pro detekci objektů. AdaBoost jsme si již vysvětlili, při podpisu Viola Jones (viz 3.1). Tento klasifikátor je trénován za běhu s pozitivními a negativními příklady objektu. Klasifikátoru se dodá ohraničující rámec s objektem, který chceme sledovat a ten funguje jako pozitivní příklad objektu a zbytek je považován jako pozadí. Při dodání nového snímku se klasifikátor spustí v sousedních oblastí předchozího umístění a zaznamená se skoré klasifikátoru. Oblast s největším skorém bude nové umístění objektu. Nové umístění je zaznamenán jako pozitivní příklad pro klasifikátor. S další iterací se klasifikátor aktualizuje o další data.

Základním principem Online AdaBoostu je zavedení tzv. selektorů. Jsou náhodně inicializovány a každý z nich obsahuje samostatné sdružení slabých klasifikátorů. Při každém novém vzorku se slabé klasifikátory všech selektorů aktualizují a selektor vybere se ten nejlepší. Jelikož aktualizace slabých klasifikátorů vyžadují mnoho výpočetního výkonu, je použita úprava tzv. globální sdružení slabých klasifikátorů, které je sdílené mezi všemi klasifikátory. Takže aktualizace se provede pouze jednou. Potom se selektory postupně přepnou na nejlepší slabý klasifikátor ohledem na aktuální odhadované skoré a váha důležitosti se přenese na další selektor. V každém kroku je také aktualizován silný klasifikátor. Ty nejhorší slabé klasifikátory jsou nahrazovány novými. [29]

4.2 MedianFlow tracker

Tento algoritmus je založen na Lucas-Kanade metodě. Algoritmus sleduje objekt v dopředném a zpětném směru a odhaduje tzv. Forward-Backward (FB) chyby těchto trajektorií, což mu umožnuje předpovídat další polohu objektu a detekovat případné selhání. [28]

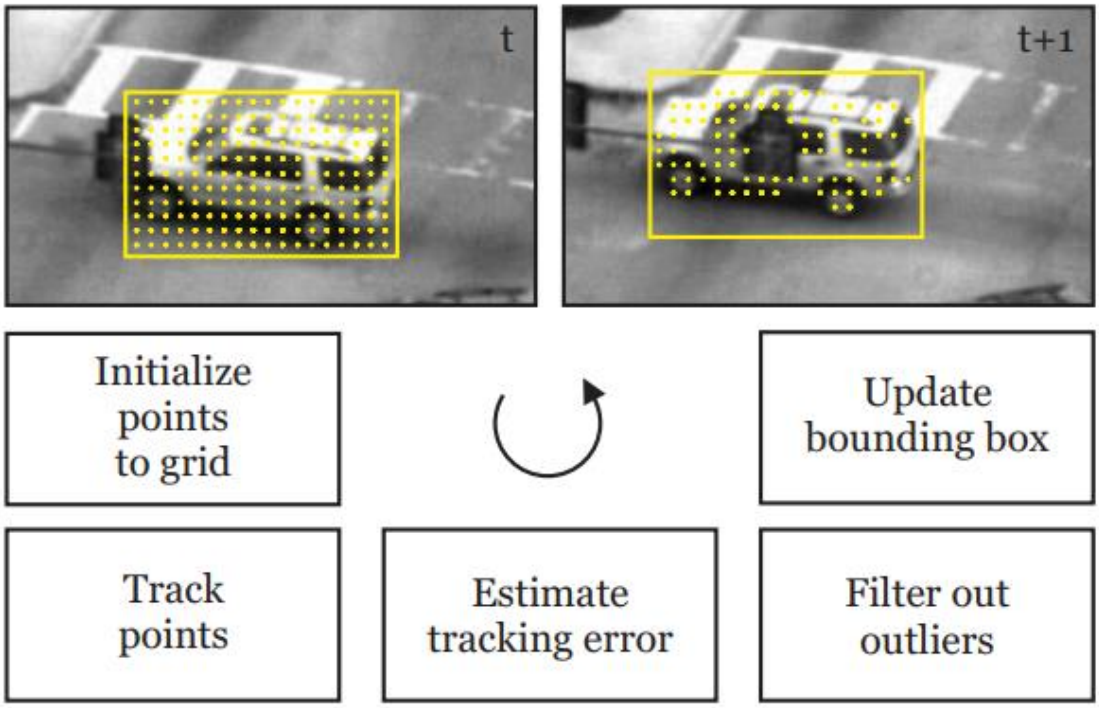
Výhodou FB chyby je snadná implementace. Sledovací algoritmy obvykle selžou, pokud data porušují předpoklady (pohyb objektu byl například rychlejší, než se očekávalo). Sledovací algoritmus vytváří trajektorií, která je do jisté míry náhodná a také zpětnou náhodnou trajektorii, které budou od sebe pravděpodobně odlišné. FB chyba je definována jako rozdíl těchto trajektorií. Dole na obrázku můžeme vidět znázornění, kde x_t je umístění bodu a I_t jako snímek. [30]



Obrázek 35 Dopředná a zpětná trajektorie [30]

Algoritmus přijme obrázek a ohraničující rámec s objektem. V rámci je inicializována sada bodů, tyto body pak sleduje Lucas-Kanade metoda, která generuje tok pohybu mezi přechozím obrázkem I_t a dalším T_{t+1}. Poté se odhaduje kvalita bodový predikcí a každému bodu je přiřazena chyba, polovina nejhorších odpovědí se vypustí. Zbývající předpovědi se využijí k odhadu posunutí ohraničujícího rámce. Odhad se vypočítá pomocí mediánu přes

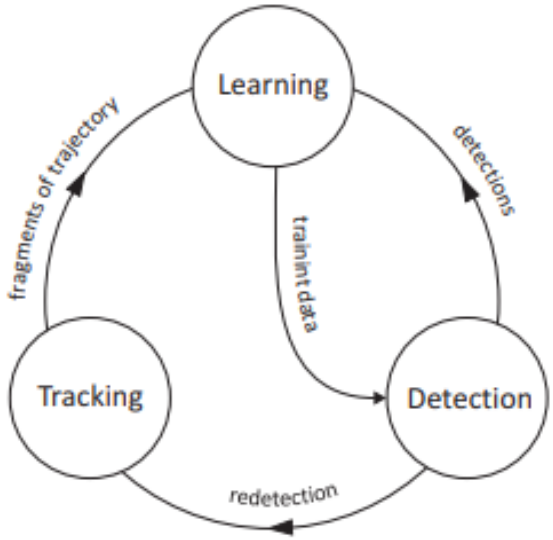
každý prostorový rozměr. Změna měřítka hraničního rámce je definována jako medián poměru vzdálenosti mezi aktuálním bodu a přechozího bodu. [30]



Obrázek 36 Proces MedianFlow [30]

4.3 TLD Tracker

TLD (Tracking, learning and detection – česky sledování, učení a detekce), jak jde poznat z názvu rozděluje sledování na tři procesy. Sledovací algoritmus je založený na MedianFlow algoritmu, který sleduje objekt. Detektor lokalizuje vyskytující se jevy a podle potřeby opraví algoritmus. Učící část odhaluje chyby, kterým bychom se budoucnu mohli vyhnout. Jelikož MedianFlow jsme si už vysvětlili, přeskočíme ho a vrhneme se rovnou na detektor. [28]



Obrázek 37 Proces TDL [31]

4.3.1 Detekce

Detektor skenuje vstupní obraz využitím metody Scanning-window grid a rozhoduje, jestli v rámci je přítomný nebo ne přítomný objekt. Ten vygenerujeme všechna možná měřítka a posuny počátečního ohraničovacího rámce. To vytvoří mnoho ohraničovacích rámců, počet závisí na rozlišení obrázku a poměru stran původního rámce. Jelikož je množství rámců vysoké, strukturujeme jej do tří fází patch variance (odchylka rámce), ensemble classifier (klasifikátor souboru), and nearest neighbor (nejbližší soused). [31]

Patch variance odmítne všechny rámce, které mají rozptyl šedé plochy menší než 50% rozptylu rámce, který právě sledujeme.

Ensemble classifier se skládá z n základních klasifikátorů, každý základní klasifikátor provede porovnání pixelů v rámci, což vede k výstupu binárnímu kódu, které jsou naindexovány do pole Posteriorů. Posteriory jsou poté zprůměrovány, pokud je průměr posterioru větší než 50 %, klasifikuje se jako objekt. [31]

Ke konci nám zbyde několik rámců, o kterých není rozhodnuto. K tomu použijeme NN (nearest neighbor) klasifikátor. Rámec je klasifikován jako rámec pokud $S^r(p, M) > 0.6$. Kde S^r je relativní podobnost (hodnoty 0-1, čím větší hodnota tím větší pravděpodobnost, že rámec obsahuje objekt), p jako rámec, M jako dynamická datová struktura, která představuje dosud pozorovaný objekt a jeho okolí Je to soubor pozitivních a negativních políček, $M = \{p^+_1, p^+_2, \ldots, p^+_m, p^-_1, p^-_2, \ldots, p^n\}$, kde p^+ a p^- reprezentují objekt a pozadí. [31]

4.3.2 Učení

Algoritmus inicializuje detektor a aktualizuje detektor pomocí hodnot P-expert a N-expert.

V prvním snímku tento komponent trénuje detektor pomocí označených příkladů. Příklady se vytváří tak, že pozitivní příklady jsou uměle vygenerovány z počátečního ohraničujícího rámce. Vybere se 10 rámců kolem počátečního rámce a každý projde dvaceti geometrickými transformacemi přidaným Gaussovým šumem, to vygeneruje dvacet pokřivených verzí. Dohromady se nám vygeneruje dvě stě pozitivních příkladů. Negativní příklady vybíráme z okolí počátečního rámce, žádné se neuměle negenerují. [31]

Cílem P-expert je objevit nové vzhledy objektu a zvýšit tak úspěšnost detektoru. P-expertů identifikuje spolehlivé části trajektorie, které používá ke generování pozitivních příkladů. V každém obrázku P-expert rozhoduje o spolehlivosti aktuální polohy, pokud je uzná za spolehlivé a vybere deset rámců a vygeneruje pro každý rámec deset pokřivených rámců upravených geometrickými transformacemi přidaným Gaussovým šumem. Dohromady se nám vygeneruje sto pozitivních příkladů. [31]

N-expert nám zase generuje negativní příklady, odhaluje nerovnosti v pozadí. Pokud je známa poloha objektu, okolí objektu je označeno jako negativní. Aplikuje se, tehdy kdy je trajektorie spolehlivá, což znamená že se aplikuje současně s P-expertem. [31]

4.4 MOSSE Tracker

MOSSE je zkratka pro Minimum Output Sum of Squared Error, používá adaptivní korelaci ve Fourierově prostoru pro sledování objektu. MOSSE je robustní a dokáže si poradit se změnou osvětlení, měřítka a deformacím. Filtr minimalizuje součet kvadratických chyb mezi výstupem konvoluce, tj nějaký vzorec obrázku a filtr. [28]

Algoritmus sleduje objekt od okamžiku, kdy mu předáme ohraničující rámec s objektem. Ten je pak sledován přes korelací filtrů ve vyhledávácím okně v dalším obrázku. Nová poloha objektu je reprezentována maximální hodnotou v korelačním výstupu. Před inicializací filtru a sledování objektu se vždy provede předběžné zpracování. Výsledek z předběžného zpracování se převede na Fourierovu doménu. Vygeneruje se tzv, syntetický neboli umělí výstup pro inicializaci a aktualizaci filtru. Ten se také převede do Fourierovy domény a v něm je vypočítán filtr. Výstup je převeden zpět do prostorové domény. [32]

V MOSSE se používá metoda Fast Fourier Transform (FFT), k výpočtu korelace ve Fourierově doméně. První se proved výpočet dvou dimenzionální Fourierovy transformace vstupního obrazu (F=F(f)) a filtru (H=F(h)).

$$G = F \odot \mathrm{H}^*$$

(6)

Kde *G* je jako výstupní odezva, *F* jako vstupní obrázek a *H* jako šablona filtru. Symbol $\odot$ představuje násobení prvku po prvku a * označuje komplexní konjugát. FFT je použita v horní rovnici. Proto se operace konvoluce stává operací násobení bodů, což značně snižuje množství výpočtů. V procesu sledování musíme ale vzít v úvahu vliv dalších faktorů jako je vzhled objektu. Vzorec je následující:

$$min_{\mathrm{H}^*} \sum_i |F \odot \mathrm{H}^* - G_i|^2$$

(7)

Po sérii transformací se získá vzorec:

$$\mathrm{H}^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*}$$

(8)

Algoritmus provede sledování přes korelací filtrů ve vyhledávacím okně v dalším obrázku a aktualizuje nové umístění objektu. Používá následující vzorec, kde η je rychlost učení:

$$H_i^* = \frac{A_i}{B_i}$$

(9)

$$A_i = \eta G_i \odot F_i^* + (1 - \eta) A_{i-1}$$

(10)

$$B_i = \eta G_i \odot F_i^* + (1 - \eta) B_{i-1}$$

(11)

K detekci chyby se měří tzv. maximální síla, jeho měření se nazývá Peak to Sidelobe Ratio (PSR). Pro výpočet PSR je vzorec :

$$PSR = \frac{g_{max} - \mu_{s1}}{\sigma_{s1}}$$

(12)

kde g_{max} je maximální hodnota a μ_{sl} a σ_{sl} jsou střední a standardní odchylka postranního laloku. [32]

II. PRAKTICKÁ ČÁST

5 POUŽITÝ HARDWARE A SOFTWARE

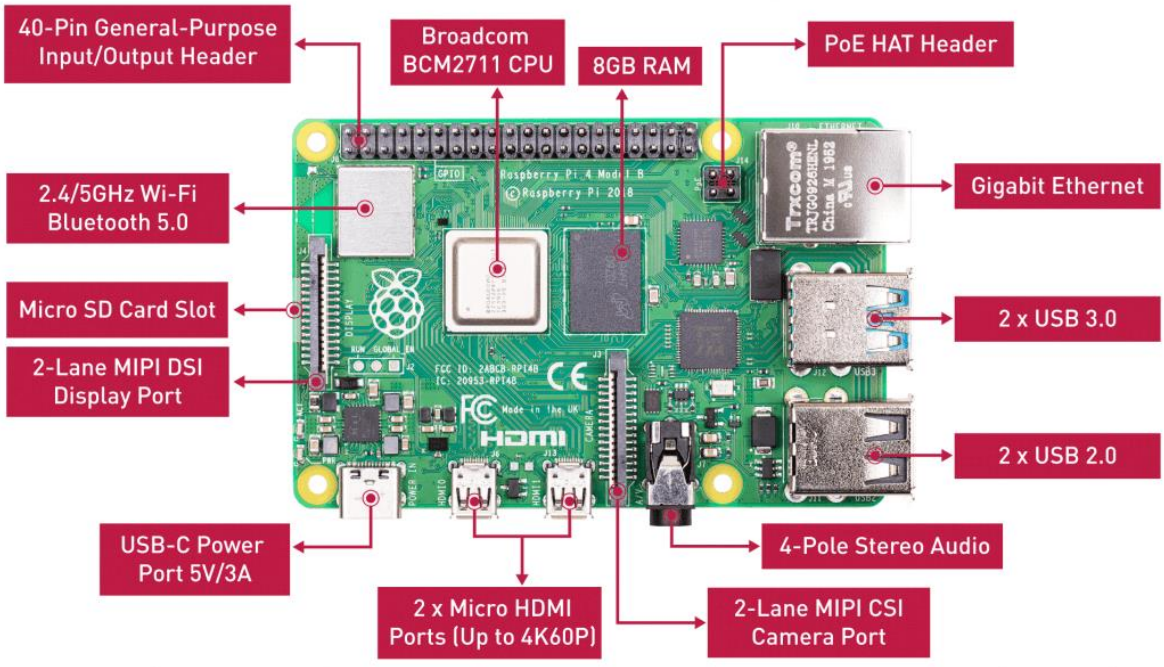
5.1 Raspberry Pi 4 Model B

Raspberry Pi je levný malý počítač, který jde dá připojit k standartním periferiím jako je normální počítač. Podle výrobce zařízení má umožnovat lidem všech věkových kategorii prozkoumat výpočetní techniku a naučit se programovat. Je schopné se integrovat se širokou řadou projektů od hudebních zařízení až po meteorologické stanice a mnoho další.

Tabulka 1 Specifikace Raspberry Pi 4 Model B

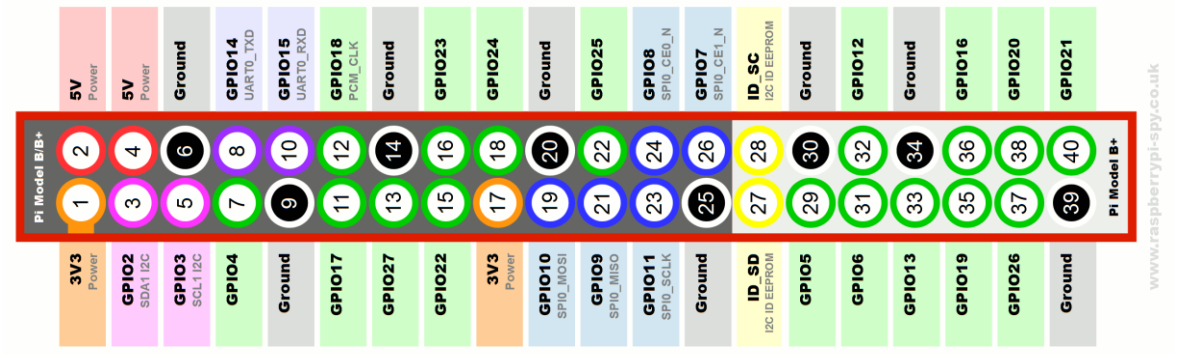
Specifikace	
Procesor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Paměť	8GB LPDDR4
Konektivita	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac Wi-Fi LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 konektor 2 × USB 2.0 konektor
GPIO	40pinový GPIO Header (zpětně kompatibilní s předchozími modely)
Video a zvuk	2 × micro HDMI porty (až 4Kp60) MIPI DSI display port MIPI CSI kamerový port čtyřpólový 3,5mm jack - výstup zvuku a kompozitního videa (PAL a NTSC)
Multimedia	H.265 (4Kp60 dekódování); H.264 (1080p60 dekódování, 1080p30 kódování); OpenGL ES, 3.0 grafika
SD	Slot pro Micro SD kartu pro operační systém

Napájení	5V DC přes USB-C konektor (minimálně 3A) 5V DC přes GPIO header (minimálně 3A) PoE - napájení přes ethernet (vyžaduje přídavný modul Raspberry Pi PoE HAT)



Obrázek 38 Popis Rapsberry Pi 4 Model B

Dole na obrázku můžeme vidět rozložení pinů, nás ale budou zajímat piny s 5 V výstupním napětím, 2 piny se zemí a 2 piny GPIO, které využijeme k ovládání servomotorů.



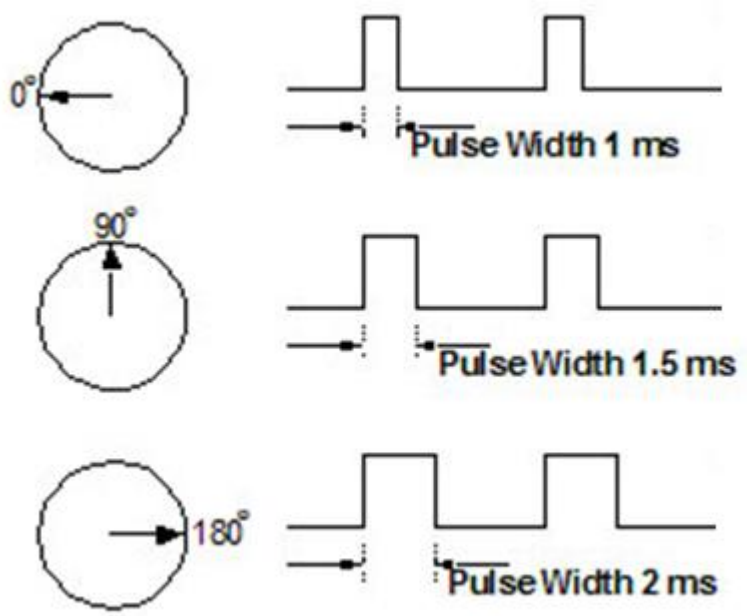
Obrázek 39 Popis PINů na Raspbery Pi

5.1.1 Raspberry Pi OS

Raspberry Pi OS (dříve Raspbian) je oficiální operační systém pro Raspberry Pi. Je založený na Debianu (jeden z mnoha distribucí Linuxu). Je k dispozici v 32bitové verzi nebo ve verzi 64bitové. OS umožnuje uživateli pracovat s grafickým uživatelským rozhraním. Přichází se zabudovaným prohlížečem chromium, s IDE Thorny Python IDE a další.

5.2 Servomotor SG90

Servomotory jsou zařízení, které se používají pro přesné natočení nějakého objektu a následné udržování jeho polohy. Používají se například u ovládání vztlakových klapek u menších letadlových modelů nebo u kormidel lodi. Na rozdíl od ostatních motorů, servomotory se dokáží nejčastěji otočit jen v rozsahu 0 až 180 stupňů. K ovládání se používá pulzní modulace šířky (PWM – Pulse Modulation Width).



Obrázek 40 PWM

Uvnitř servomotoru se nachází malý stejnosměrný motor, který otáčí s výstupní páčkou. Na výstupu se nachází potenciometr, který měří polohu páčky. Délku řídícího signálu s hodnotou potenciometru poté porovnává elektronika. Řekněme že, 1 ms délky řídícího odporu je rovná 0 Ω a 2 ms je roven 1000 Ω. Servomotor je zrovna v poloze s 0 Ω a příjde

řídící signál se hodnotou 2 ms, Elektronika postí kladný signál motorku a by se posunul na cílovou polohu. Pro posun zpět se pustí proud v záporné směru.

Obrázek 41 Tower Pro SG90 9g

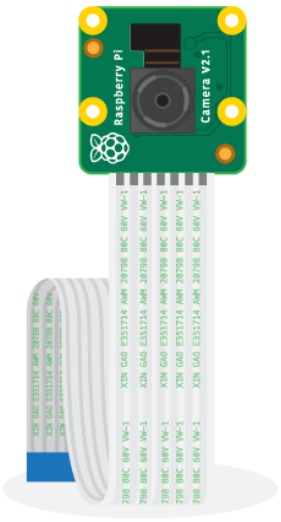
Pro ovládání kamery se zvolily 2 servomotory Tower Pro SG90 9g, jedno pro ovládání posunu a jeden pro ovládání náklonu. Je to poměrně jednoduchý motor a jeho zapojení není složíte, je malý a váží jen 9 gramů. Vychází z něho tři vodiče, červený vodič se připojuje na napájecí napětí, hnědý vodič se napojuje na zem a oranžový vodič je pro ovládací signál.

Tabulka 2 Specifikace Tower Pro SG90 9g

Specifikace	
Rozměry	21 x 12 x 29 mm
Váha	9 g
Rychlost	0.12 sec/60 (4.8 V) 0.09 sec/60 (6 V)
Tah	1.8 kg-cm
Napájení	3 V ~ 7,2 V
Rozsah otáčení	0-180°
Pulzní cyklus	cca. 20 ms, 0,5 ms - 0°, 2,4 ms - 180°
Šířka pulzu	500-2400 µs, 500 µs - 0°, 2400 µs - 180°

5.3 Raspberry Pi Camera Module V2

Kamera je oficiální modul přímo pro Raspberry Pi, stačí ji zapojit na kamerový port. Má již v ní také zabudovanou Python knihovnu, se kterou můžeme lehce ovládat kameru. Jak kamera snímá obraz viz 1.1.1.



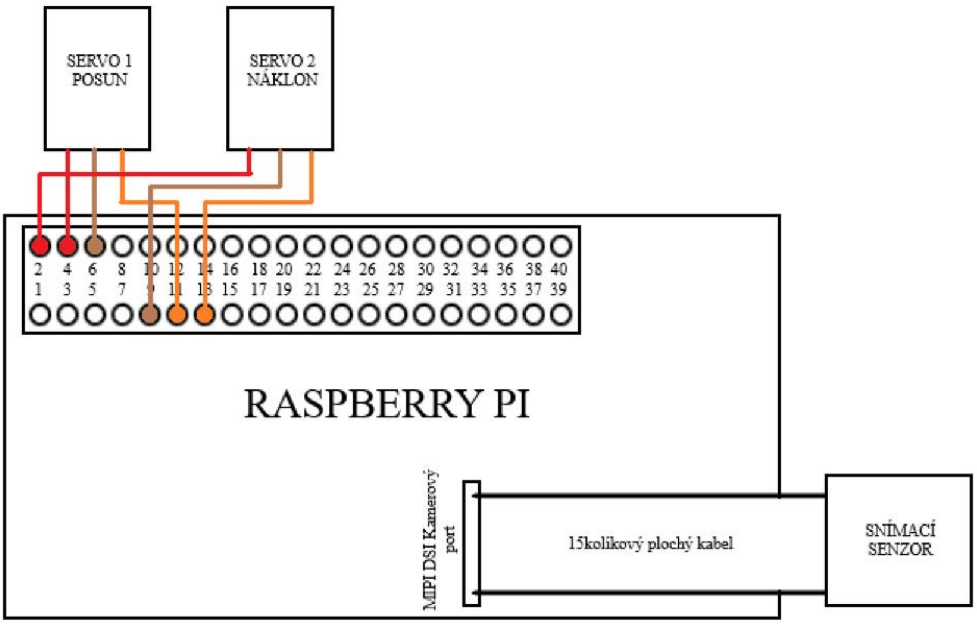
Obrázek 42 Raspberry Pi Camera Module V2

Tabulka 3 Specifikace Raspberry Pi Camera Module V2

Specifikace	
Senzor	Sony IMX 219 PQ CMOS obrazový snímač v modulu s pevným ohniskem.
Rozměry	23.86 x 25 x 9 mm
Váha	3 g
Velikost objektivu	6,35 mm
Video rozlišení	1080p30 720p60
Fotka rozlišení	3280x2464
Napojení na Raspberry Pi	15kolíkový plochý kabel

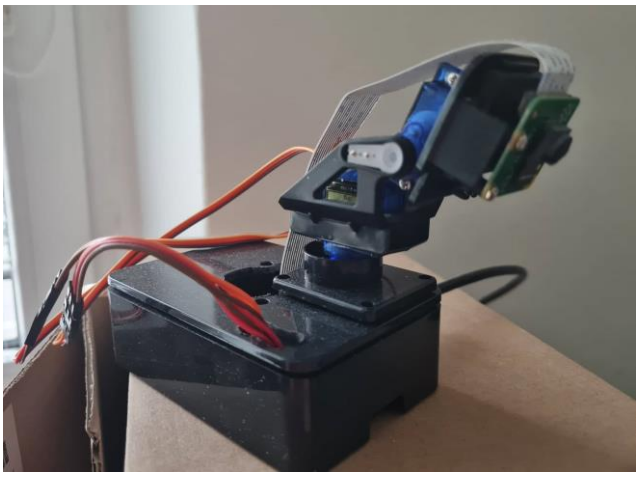
6 REALIZACE

6.1 Zapojení



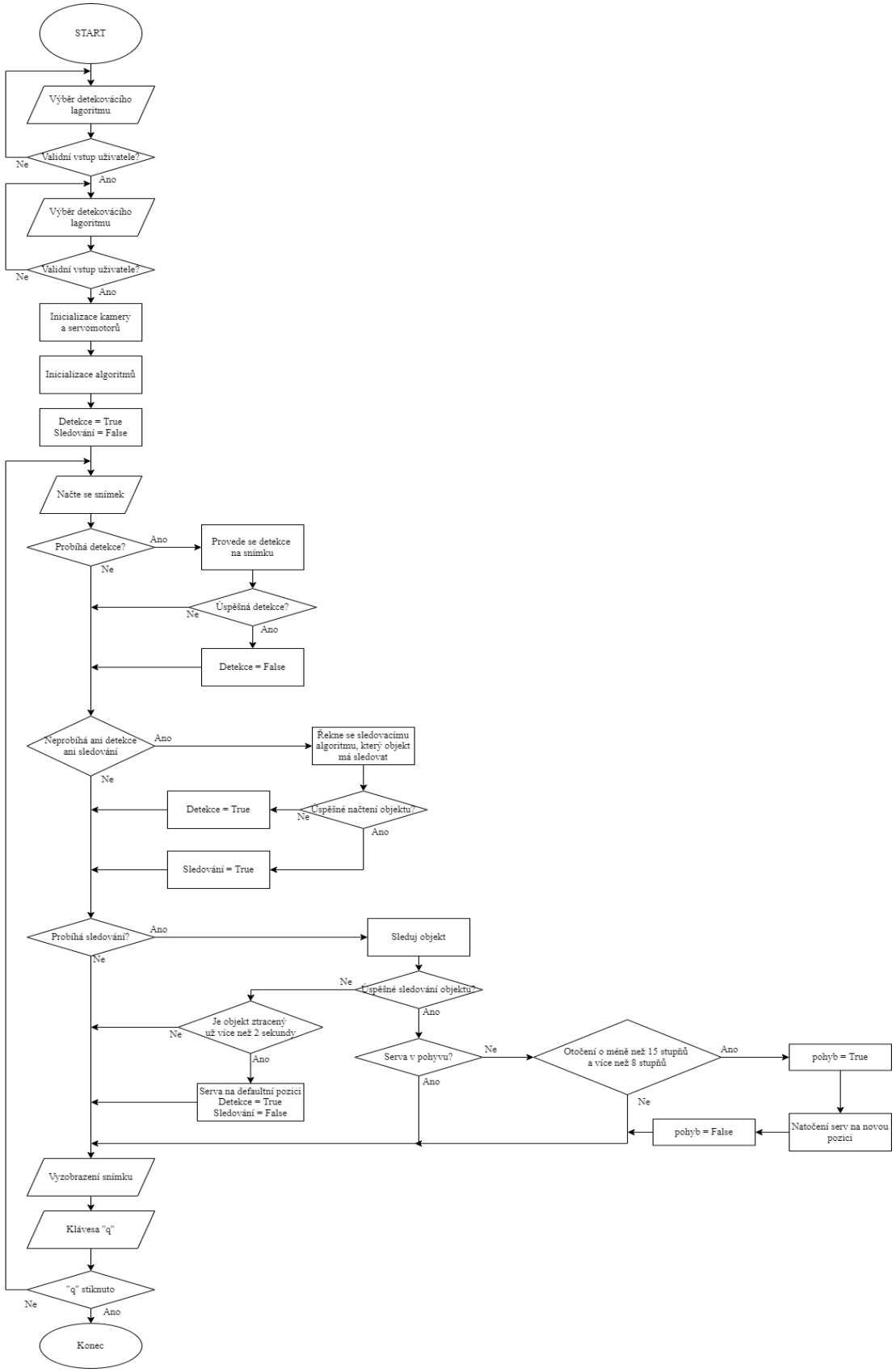
Obrázek 43 Zapojení

Jak jde vidět ze schématu, zapojení je jednoduché. Dalším krokem je sestavit nějakou základnu, na které bychom ovládali servomotory a otáčeli kamerou. Jako základnu použijeme levnou otočnou základnu bez serva, základna se dá lehce sehnat na internetu. Základna je kompatibilní se SG90 servomotorem, ale pro kameru se musel vyrobit speciální podklad. Podklad se poté přilepil na základnu. Pro zapojení se ještě použili propojovací kabely a obal pro Raspberry Pi.



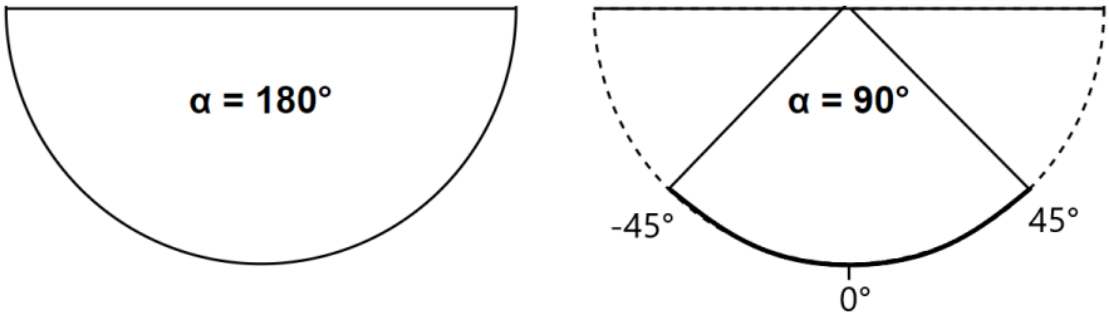
Obrázek 44 Konečné sestavení

6.2 Návrh programu



Obrázek 45 Diagram algoritmu

Ze začátku se nás program zeptá, který z algoritmů si chceme vybrat. Po výběru algoritmů program provede inicializaci algoritmů, nastaví potřebné parametry (načte modely algoritmů a jiné hodnoty). Jako další je inicializace snímacího zařízení, nastaví se rozlišení, tomto případě se nastaví na 640 x 480 a obnovovací frekvence na 24 Hz. Jelikož se jedná o oficiální modul Raspberry Pi OS, máme zde již implementovanou knihovnu na ovládání kamery nazývaná PiCamera. Pro nepřetržité snímání je využitá funkce **capture_continuous(output, format='bgr', use_video_port=True),** která vrací nekonečný iterátor snímků snímaných z kamery. Parametr output neboli česky výstup se nastaví na 3 dimenzionální RGB pole, k tomu využijeme funkci **PiRGBArray(camera, size=resolution)**. Pole získáváme pomocí atributu **array**, ovšem atribut musíme vyprazdňovat pomocí funkce **truncane(0)**, mezi snímáním. Kamera běží ve vlastním vlákně. K inicializaci motorů využijeme knihovnu gpiozero, které je také již implementovaná v Raspberry Pi OS. Knihovna je vybavená s rozhraním AngularServo, rozhraní rovnou pracuje s úhly. To nám velice ulehčuje programování, jelikož nemusíme pracovat s pulzní modulační šířkou, o to se už stará rozhraní. AngularServo umožnuje také maximální rozsah natočení, což je 90 stupňů na ose x a 50 stupňů na ose y. Je tu jedna věc, na kterou si musíme dát pozor a to je, že střed není 45 stupňů, ale 0 stupňů a nachází se v polovině maximálního rozsahu servomotoru.



Obrázek 46 Otáčení servomotoru

Servomotor vykazoval problémy u udržování klidové pozice, měl tendenci se trhat a to nebylo dobré, mělo to výrazné nežádoucí efekty na sledovací algoritmy. Bylo to způsobeno nepřesným řídícím signálem. Problém byl vyřešen rozhraním PiGPIO, který spoléhá na deamon běžící jako root, aby poskytoval přístup k pinům. PiGPIO komunikuje s deamonem přes šítový socket.

Aby mohl program sledovat osobu, musíme mu nejprve říct, jak vypadá a kde se nachází. Proto z počátku přichází detekce, pokud se úspěšně detekuje osoba program se přepne

z režimu detekce a snaží se inicializovat s pozicí ohraničujícího rámce sledovací algoritmus. Po úspěšné inicializaci se přepne do sledovacího režimu, nebo při selhání zpátky do režimu detekce. Při sledování osoby bude algoritmus aktualizovat informace o poslední pozici, podle kterých se bude základna hýbat. O kolik stupňů se mám servomotor nám říká následující vzorce:

$$posun = \frac{střed\ snímku\ x - střed\ ohraničujcího\ rámce\ x}{7}$$

$$náklon = \frac{střed\ snímku\ y - střed\ ohraničujcího\ rámce\ y}{8} \quad (13)$$

(13)

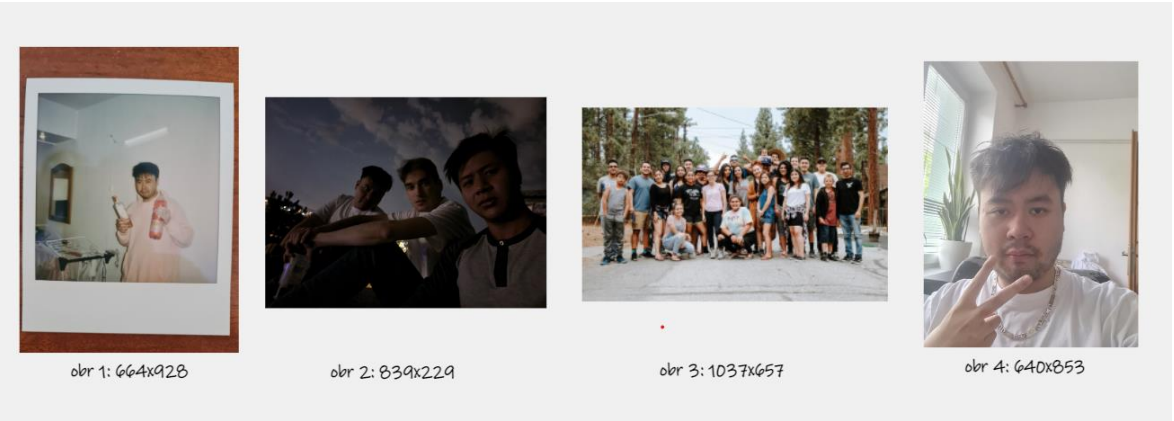
Servomotory nemají jemné ovládání pohybu a posouvají se velkou rychlostí což může rozhodit sledovací algoritmus. Řešením byla postupná iterace k cílové pozici. Posouvají se postupně s malými přestávkami mezi iterací. Nevýhodou je, ale zbytečné zabrání výpočetního výkonu a snížením počtu snímků za sekundu. Provádí se také kontrola citlivosti, aby se kamera neposouvala za osobou po každém malém pohybu, je nastaven minimální posun, který může provést (8 stupňů). To samé platí pro maximální posun (15 stupňů). Může ale nastat, že se osoba ztratí a už ji nenalezne. Pokud se po dvou sekundách osoba znovu nenalezne, program se zpátky přepne do režimu detekce a posune se do výchozí pozice. Snímek se nám následně zobrazí. Stisknutím q se nám program ukončí.

Detekovaní algoritmy jsme použili s přetrénované. Viola Jones je už ze samého začátku implementovaný, Mask R-CNN a YOLO jsou přetrénované na COCO Datasetu a SSD přetrénované na Pascal VOC datasetu. Jelikož některé algoritmy dokázaly detekovat více objektových tříd, museli jsme se ujistit, že vyřadíme objekty, které nevyžadujeme. Naštěstí algoritmy vraceli vektor s ohraničujícím rámce, s objektovou třídou a pravděpodobnost. Vyloučili jsme, proto všechny objekty, u který byla pravděpodobnost menší než 50 % a objekty s nežádoucí třídou. Zbývají zde jen osoby, ale chceme sledovat je jednu. Jednoduše podle Pythagorovy věty vypočítáme vzdálenost od středu snímku a vybereme tu nejbližší.

7 TESTOVÁNÍ

7.1 Detekce

Nejprve se testovali algoritmy na detekování. Použili se čtyři různé obrázky a každý obrázek jsme nechali 10x projít každým algoritmem. V průběhu jsme měřili, za jaký čas se daný algoritmus dokončí. Celé měření probíhalo na Raspberry Pi



Obrázek 47 Vstupní obrázky

Obrázky se lišili v měřítku, ale byla tu snaha je mít alespoň co nejpodobnější velikosti. Zvolili se obrázky v různých situacích, jiné světelné podmínky, v rozdílu v počtu osob nebo ve velikosti osob na obrázcích. Obrázek 3 je stáhnutý z internetu, jinak zbylé fotky jsou osobní.

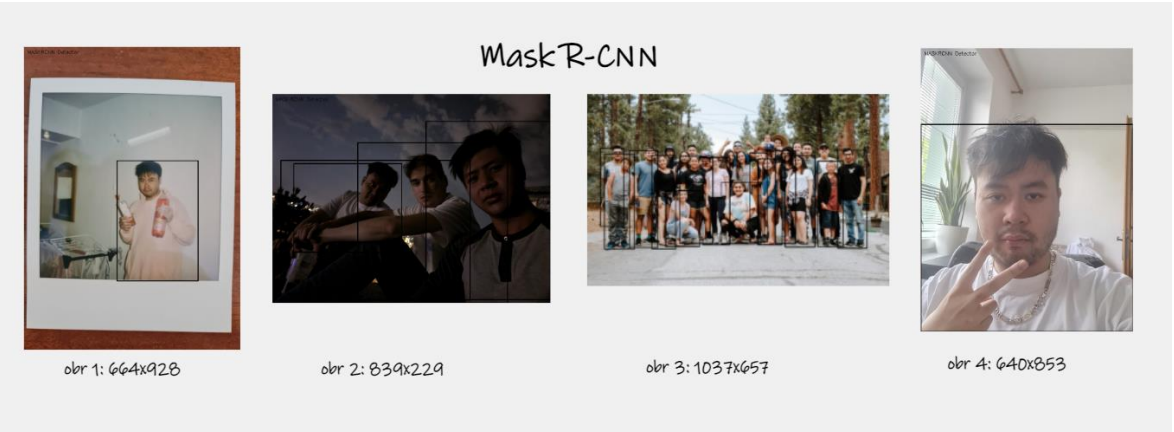
Obrázek 48 Výstupní obrázky Viola-Jones

Tabulka 4 Výsledky měření Viola-Jones

Viola Jones				
n	**obr1 [s]**	**obr2 [s]**	**obr3 [s]**	**obr4 [s]**
1	0.519	0.631	0.873	0.787

2	0.541	0.607	0.949	0.772
3	0.404	0.456	0.752	0.693
4	0.454	0.496	0.789	0.701
5	0.430	0.504	0.917	0.726
6	0.415	0.450	0.760	0.698
7	0.610	0.439	0.756	0.618
8	0.592	0.470	0.756	0.722
9	0.445	0.470	0.858	0.638
10	0.435	0.457	0.806	0.629
Průměr	**0.485**	**0.498**	**0.822**	**0.698**

Viola-Jones byl ze všech čtyř algoritmů nejrychlejší. Úspěšně zpracoval obrázku pod sekundu. Ale měl problém s detekováním více osob, našem případě to není problém, ale pokud bychom chtěli rozpoznat více osob, vyhnul bych se mu. I když Viola-Jones umí také detekovat celé osoby, spíše rozpoznával obličeje, je taky na to stavěný, v našem případě také zanedbatelný, jelikož obličej je také osoba.



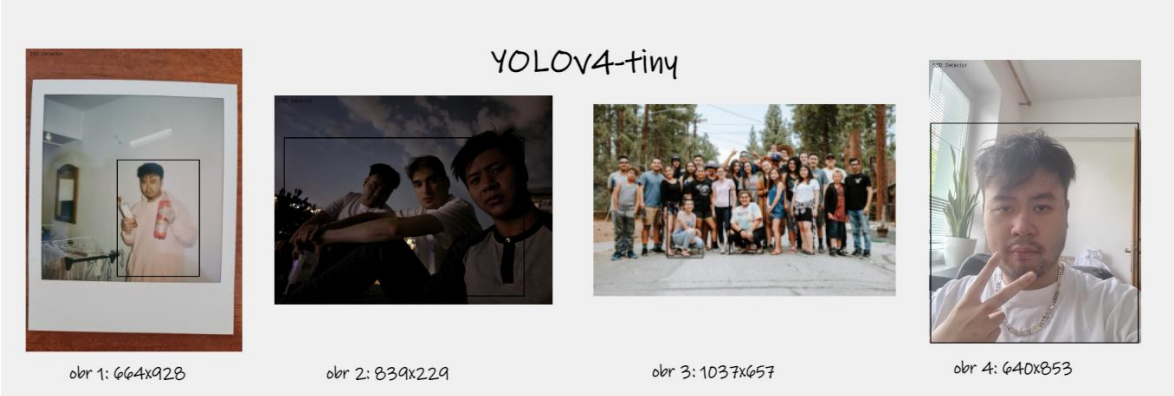
Obrázek 49 Výstupní obrázky Mask R-CNN

Tabulka 5 Výsledky měření Mask R-CNN

Mask R-CNN				
n	**obr1 [s]**	**obr2 [s]**	**obr3 [s]**	**obr4 [s]**

1	62.539	81.015	89.434	87.457
2	68.443	83.336	89.376	84.937
3	72.713	86.631	90.202	86.803
4	74.271	84.328	92.631	92.830
5	77.864	84.342	91.802	91.241
6	78.655	86.370	91.707	89.709
7	80.583	84.626	91.105	90.143
8	84.228	83.890	90.751	88.132
9	82.151	84.691	92.068	87.558
10	84.564	84.063	95.382	87.296
Průměr	**76.601**	**84.329**	**91.446**	**88.611**

Mask R-CNN byl ze všech nejpomalejší, dokázal rozpoznat nejvíc osob, úspěšně rozpoznal celou řadu na obrázku 3, jen u tmavšího obrázku 3 rozpoznal jednu osobu dvakrát. Vzhledem k časové náročnosti Mask R-CNN není vhodný na real-time detekování.



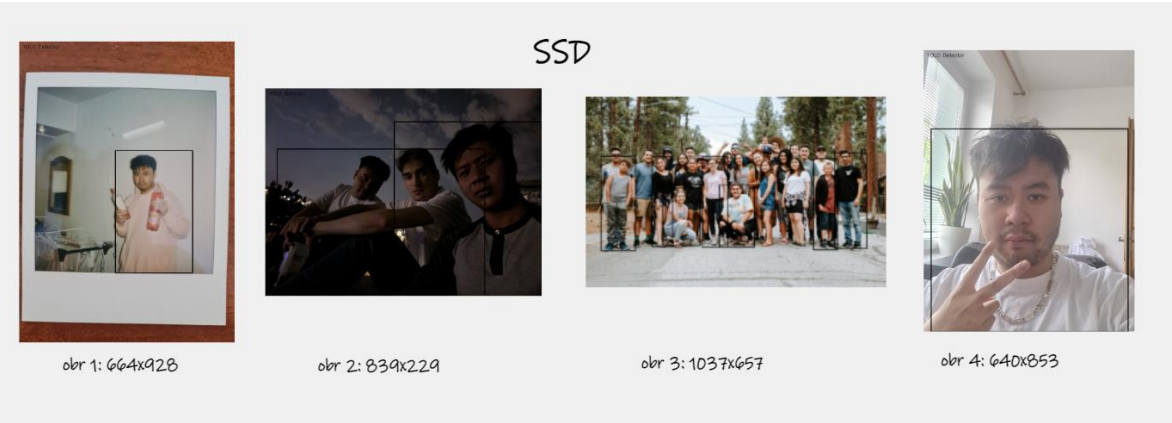
Obrázek 50 Výstupní obrázky YOLOv4-tiny

Tabulka 6 Výsledky měření YOLOv4-tiny

YOLOv4-tiny				
n	**obr1 [s]**	**obr2 [s]**	**obr3 [s]**	**obr4 [s]**
1	2.634	3.087	3.154	2.917

2	2.898	3.213	3.196	3.424
3	2.909	3.028	2.924	3.079
4	2.936	3.031	2.962	3.192
5	3.165	3.190	2.900	3.359
6	2.977	2.922	3.157	3.003
7	2.978	3.126	2.905	2.863
8	3.004	2.999	2.997	2.910
9	3.315	2.878	3.192	2.783
10	2.895	3.168	2.915	3.164
Průměr	**2.971**	**3.064**	**3.030**	**3.069**

YOLOv4-tiny si vedl v celku dobře, tři sekundy jsou přijatelné. Na výkonnějších čipech bychom se určitě dostali na milisekundy. Z obrázků je jasné vidět, že se nerovná Mask-RCNN, měl problém rozpoznávat osoby v menším měřítku a nelze přesně určit jakou osobu na obrázku 2 ohraničuje rámcem.



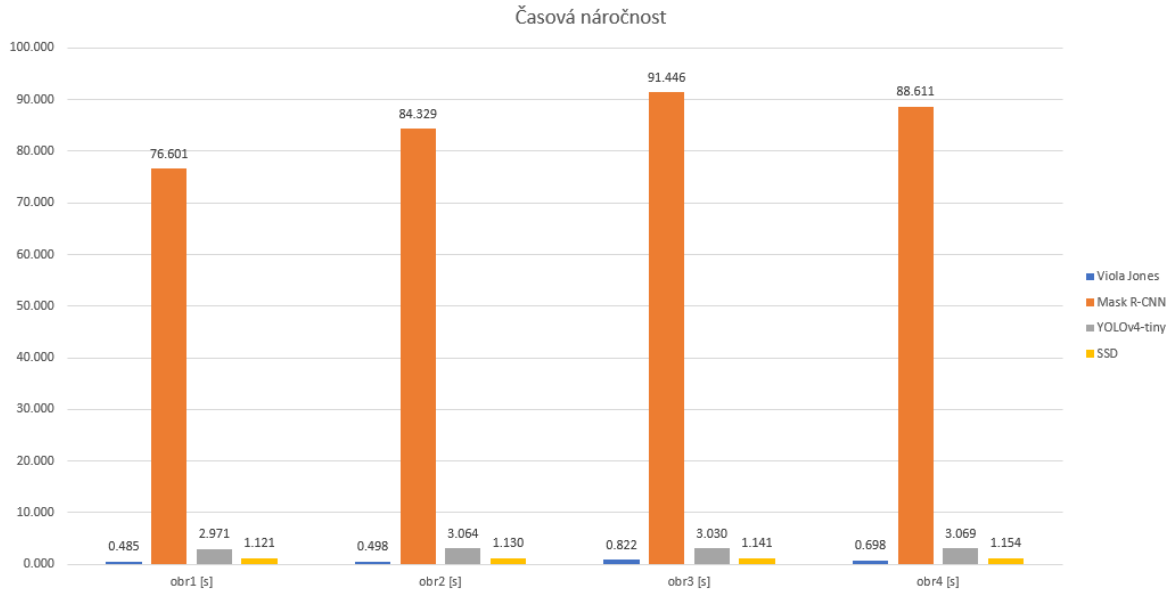
Obrázek 51 Výstupní obrázky SSD

Tabulka 7 Výsledky měření SSD

SSD				
n	**obr1 [s]**	**obr2 [s]**	**obr3 [s]**	**obr4 [s]**
1	1.039	1.141	1.197	1.270

2	1.291	1.453	1.285	1.326
3	1.092	1.150	0.971	1.178
4	0.881	0.971	1.213	1.059
5	1.153	1.025	1.290	1.153
6	1.369	1.127	1.021	0.999
7	1.090	1.100	1.059	0.982
8	1.039	1.211	0.918	1.106
9	0.998	1.135	1.361	1.192
10	1.257	0.990	1.090	1.274
Průměr	**1.121**	**1.130**	**1.141**	**1.154**

SSD byl v detekci na druhé pozici, nedokázal rozpoznat jednu osobu na obrázku 2 a obrázku 3 rozpoznal asi polovinu přední řady, ale v rychlosti si vedl velice dobře. Hodnoty se nacházely lehce nad sekundou.



Obrázek 52 Časová náročnost všech algoritmů

7.2 Sledování

Cílem bylo pozorovat chování algoritmu, za jistých podmínek videích o měřítku 640x480. V prvním videu byla jednoduchá chůze ze strany na stranu, ve druhém videu jsme se zaměřili

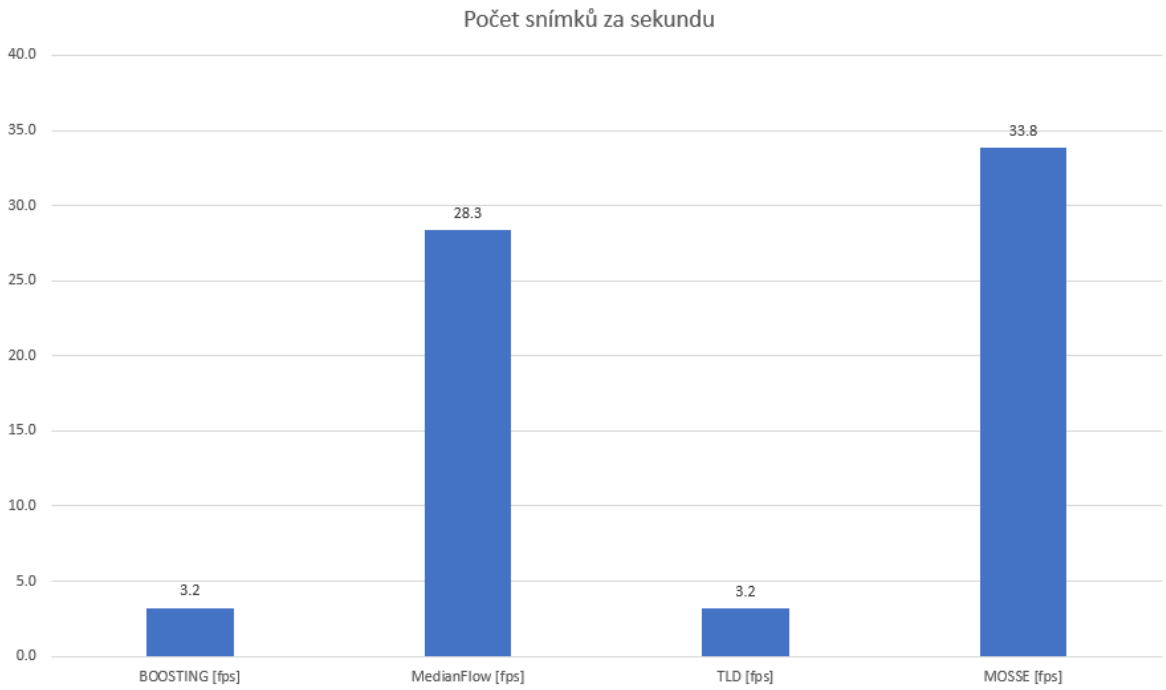
na chůzi ve kolečku neboli přibližování a oddalování osoby. Ve třetím videu jsme se zaměřili průchod za překážkou a poslední video byli náhodné pohyby.

BOOSTING měl v tendenci se zasekávat na jednou místě a už se nehýbat s osobou, museli jsme se postavit zpátky před ohraničující rámec, aby pokračoval v našem sledování. Ohraničující rámec, ale zůstával v konstantní velikosti. Samotný algoritmus byl jen průměrný a při sledování zpracovával obraz průměru jen tři snímky za sekundu (fps).

MedianFlow ztrácel osobu mnohem více než BOOSTING, nezvládal prudké pohyby a nesmyslně zvětšoval nebo zmenšoval ohraničující rámeček. Průměrná hodnota fps se rovnala 28 snímků za sekundu.

TDL jako jedí dokázal sledovat osobu, které prošla překážkou. Náhodně přeskakoval, ale nikdy osobu neztratil a zpátky se navedl. Pracoval rychlostí třech snímků za hodinu.

MOSSE je nejrychlejší. To už asi bylo všechno, ztrácel osobu, pokud dělala prudší pohyby. Hodnota fps se rovnala 33, což je nejvyšší ze všech čtyř.



Obrázek 53 Průměrná hodnota fps

7.3 Real-time sledování na enbedded zařízení

Kamera snímala v rozlišení 640x480. U algoritmů detekování až na Mask R-CNN nebyl problém, museli jsme stát asi minutu před kamerou, aby nás zpracoval a předal sledovacímu algoritmu.

V živém sledování si algoritmy dařili ještě lépe, jelikož jsme měli možnost se přizpůsobovat kameře. Tři algoritmy dosahovali ještě většího počtu snímků než u videí. Jediný MOSSE si vedl hůř a by o polovinu pomalejší. BOOSTING se choval stejně a zasekával se. Vlastnost náhodného přeskakování TDL znemožňovala plynulému sledování osoby. MediumFlow fungoval nejlépe a dosahoval rychlosti kolem 35 až 50 snímku za sekundu. Všechny algoritmy vyžadovali spíše menší a plynulejší pohyby. Nejsou proto vhodné pro náhlé a zběsilé pobíhání.

Stojí také za zmínku, že při pohybu kamera ztrácíme fps, je totiž potřebný výpočetní výkon pro plynulejší posun kamery.

ZÁVĚR

Můžeme zvážit, že ani jeden algoritmus na detekování není vhodný na kontinuální real-time detekci na Raspberry Pi. I když 1 sekunda je čas, který pro nás uběhne jen mrknutím oka, video záznamy ve většině pracují při 30 snímcích za sekundu (přibližně 33 milisekund každý snímek). To by spíše pro nás vznikla prezentace snímků. Jelikož se tento algoritmus v nejlepším případě spustí pouze jednou ze začátku, můžeme považovat tento proces jako inicializační, v tomto případě je časová prodleva přijatelná. Jako nejvhodnější algoritmus bych zvolil SSD. Viola-Jones a YOLOv4-tiny jsou také dobrou volbou, ale Viola-Jones je spíše na detekci obličejů a YOLOv4-tiny je výkonnostně a s přesností na tom hůře než SSD, raději bych se obrátil na plnou verzi YOLOv4, pokud by se využíval výkonnější počítač.

Ze sledovací algoritmů vyhrává MedianFlow. Je dobrý na sledování plynulých pohybů a předvídatelných pohybů, což bude ve většině případech používání tohoto zařízení. Jako jediný dosahuje akceptující výsledků. MOSSE oproti sledování video záznamu si vedl hůře než za živého sledování, měl je kolem 17 snímků za sekundu. TDL velice přeskakoval a měl veliké množství falešných pozitiv s kombinací nízkého fps. BOOSTING byl pomalý a nemohl fungovat spolehlivě.

SEZNAM POUŽITÉ LITERATURY

[1] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. Image processing, analysis, and machine vision. Fourth edition. Stamford, CT, USA: Cengage Learning, [2015]. ISBN 1133593607.

[2] KASAL, JAROSLAV. Cesta do hlubin digitální kamery - jak pracuje digitální kamera. Www.pcworld.cz [online]. Česká republika: pcworld, 2002 [cit. 2022-05-09]. Dostupné z: https://www.pcworld.cz/clanky/cesta-do-hlubin-digitalni-kamery-jak-pracuje-digitalni-kamera/

[3] Computer Vision: What is an image, and how are they stored? [online]. Spojené státy: Sundog Education with Frank Kane, 2018 [cit. 2022-05-09]. Dostupné z: https://www.youtube.com/watch?v=YsonLUQ-rTU&ab_channel=SundogEducationwithFrankKane

[4] The Computer Vision Pipeline, Part 3: image preprocessing [online]. Spojené státy: freecontent.manning.com, 2019 [cit. 2022-05-09]. Dostupné z: https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/

[5] Zelené a UV lasery [online]. Česká republika: Jiří Dušek, 2012 [cit. 2022-05-09]. Dostupné z: https://www.mega-blog.cz/lasery/zelene-a-uv-lasery/

[6] O barevných modelech RGB a CMYK [online]. Česká republika: Digineff, 2015 [cit. 2022-05-09]. Dostupné z: https://digineff.cz/o-barevnych-modelech-rgb-a-cmyk/

[7] Charge-coupled device [online]. San Francisco (CA): wikiwand, 2022 [cit. 2022-05-09]. Dostupné z: https://www.wikiwand.com/cs/Charge-coupled_device

[8] Color conversions [online]. Spojené státy: OpenCV, 2022 [cit. 2022-05-09]. Dostupné z: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

[9] HANTOUSH ALRUBAIE, Shymaa Akram a Ahmed Hasan HAMEED. Dynamic Weights Equations for Converting Grayscale Image to RGB Image. JOURNAL OF UNIVERSITY OF BABYLON for Pure and Applied Sciences [online]. 2018, 26(8), 122-129 [cit. 2022-05-09]. ISSN 2312-8135. Dostupné z: doi:10.29196/jubpas.v26i8.1677

[10] Python OpenCV: Converting an image to gray scale [online]. Spojené státy: techtutorialsx, 2019 [cit. 2022-05-09]. Dostupné z: https://techtutorialsx.com/2018/06/02/python-opencv-converting-an-image-to-gray-scale/

[11] Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1) [online]. Spojené státy: analyticsvidhya, 2019 [cit. 2022-05-09]. Dostupné z: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

[12] COMPUTER VISION AND IMAGE CLASSIFICATION -A STUDY [online]. Spojené státy: Dexlab, 2020 [cit. 2022-05-09]. Dostupné z: https://www.dexlabanalytics.com/blog/computer-vision-and-image-classification-a-study

[13] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2018 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[14] About [online]. Spojené státy: OpenCV, 2022 [cit. 2022-05-09]. Dostupné z: https://opencv.org/about/

[15] What is OpenCV | OpenCV Python Tutorial For Beginners | Updegree [online]. Indie: Up Degree, 2020 [cit. 2022-05-09]. Dostupné z: https://www.youtube.com/watch?v=_aBFA4g8l8E&ab_channel=UpDegree

[16] Object Detection in 2022: The Definitive Guide [online]. Spojené státy: Gaudenz Boesch, 2022 [cit. 2022-05-09]. Dostupné z: https://viso.ai/deep-learning/object-detection/

[17] GUPTA, Rohan Gupta. Breaking Down Facial Recognition: The Viola-Jones Algorithm. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2019 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999

[18] LEE, Socret. Understanding Face Detection with the Viola-Jones Object Detection Framework. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2020 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14

[19] TYAGI, Mrinal. Viola Jones Algorithm and Haar Cascade Classifier. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2021 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8

[20] GANDHI, Rohith. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2018 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[21] VERMA, YUGESH. R-CNN vs Fast R-CNN vs Faster R-CNN – A Comparative Guide. Analyticsindiamag [online]. Česká republika: DEVELOPERS CORNER, 2021 [cit. 2022-05-09]. Dostupné z: https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/

[22] KHANDELWAL, Renu. Computer Vision: Instance Segmentation with Mask R-CNN. Towardsdatascience.com [online]. Indie: towardsdatascience, 2019 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1

[23] Manishgupta. YOLO — You Only Look Once. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2020 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4

[24] KARIMI, Grace. Introduction to YOLO Algorithm for Object Detection. Section.io [online]. Kenya: section, 2021 [cit. 2022-05-09]. Dostupné z: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

[25] MORGUNOV, Anton. Object Detection with YOLO: Hands-on Tutorial. Neptune.ai [online]. Spojené státy: neptune, 2021 [cit. 2022-05-09]. Dostupné z: https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial

[26] ANKA, Andrej. YOLO v4: Optimal Speed & Accuracy for object detection. Towardsdatascience.com [online]. Spojené státy: towardsdatascience, 2020 [cit. 2022-05-09]. Dostupné z: https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50

[27] SSD [online]. Spojené státy: leonardoaraujosantos, 2020 [cit. 2022-05-09]. Dostupné z: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/single-shot-detectors/ssd

[28] A Complete Review of the OpenCV Object Tracking Algorithms [online]. Spojené státy: broutonlab, 2020 [cit. 2022-05-09]. Dostupné z: https://broutonlab.com/blog/opencv-object-tracking

[29] GRABNER, H., M. GRABNER a H. BISCHOF. Real-Time Tracking via On-line Boosting. In: Procedings of the British Machine Vision Conference 2006 [online]. British Machine Vision Association, 2006, 2006, 6.1-6.10 [cit. 2022-05-09]. ISBN 1-901725-32-4. Dostupné z: doi:10.5244/C.20.6

[30] KALAL, Zdenek, Krystian MIKOLAJCZYK a Jiri MATAS. Forward-Backward Error: Automatic Detection of Tracking Failures. In: 2010 20th International Conference on Pattern Recognition [online]. IEEE, 2010, 2010, s. 2756-2759 [cit. 2022-05-09]. ISBN 978-1-4244-7542-1. Dostupné z: doi:10.1109/ICPR.2010.675

[31] KALAL, Zdenek, Krystian MIKOLAJCZYK a Jiri MATAS. Tracking-Learning-Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2012, 34(7), 1409-1422 [cit. 2022-05-09]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2011.239

[32] BOLME, Dav, J. Ross BEVERIDGE, Bruce A. DRAPER a Yui Man LUI. Visual object tracking using adaptive correlation filters. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition [online]. IEEE, 2010, 2010, s. 2544-2550 [cit. 2022-05-09]. ISBN 978-1-4244-6984-0. Dostupné z: doi:10.1109/CVPR.2010.5539960

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CCD	Význam první zkratky
CMOS	Význam druhé zkratky
RGB	Význam třetí zkratky
SVM	Support vector machines
CNN	Convulutional Neural Networks
OpenCV	Open Source Computer Vision Library
R-CNN	Region Based Convolutional Neural Networks
YOLO	You Only Look Once
SSD	Single Shot Multibox Detektor
ROI	Region of Interest
NMS	Non-max suppression
VGG	Visual Geometry Group
IoU	Intersection over Union
FPN	Feature Pyramid Network
BoF	Bag of specials
BoF	Bag of specials
SE	Squeeze-and-Excitation
SAM	Spatial Attention Module
SPP	Spatial Piramid Pooling
ASPP	Atrous Spatial Piramid Pooling
RFB	Receptive Field Block
PANet	Path Aggregation Network
PAN	Path Aggregation Network
FB	Forward-Backward
NN	nearest neighbor

TLD	Tracking, learning and detection
MOSSE	Minimum Output Sum of Squared Error
FFT	Fast Fourier Transform
PSR	Peak to Sidelobe Ratio
GPIO	General Purpose Input/Output
LAN	Local Area Network
MIPI	Mobile Industry Processor Interface
DSI	Display Serial Interface
CSI	Camera Serial Interface
PWM	Pulse Modulation Width
FPS	Frames per second

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY