VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE PALNÝCH ZBRANÍ V OBRAZU

WEAPON DETECTION IN AN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVOL DEBNÁR**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL DVOŘÁK**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Debnár Pavol**
Program: Informační technologie
Název: **Detekce palných zbraní v obrazu**
Weapon Detection in an Image
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte literaturu týkající se detekce objektů v obrazu a seznamte se s nejčastěji využívanými detekčními algoritmy.
2. Připravte anotovanou databázi snímků zbraní.
3. Navrhněte algoritmus pro detekci zbraní ve 2D obrazu.
4. Postup navržený v předchozím bodu implementujte. Otestujte Vaše řešení na databázi vytvořené v bodě 2.
5. Shrňte a zhodnoťte dosažené výsledky. Diskutujte možnosti budoucího vývoje.

Literatura:

- RUSS, J.C. The Image Processing Handbook. Boca Raton : CRC Press, 1995. 674 p. ISBN 0-8493-2516-1.
- SONKA, Milan, HLAVAC, Vaclav, BOYLE, Roger. Image Processing, Analysis and Machine Vision. 3rd edition. Toronto : Thomson, 2008. 829 p. ISBN 978-0-495-08252-1

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a 3.

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

Vedoucí práce: **Dvořák Michal, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Táto práca sa zaoberá detekciou palných zbraní v obraze. V teoretickej časti je najprv definovaný pojem palná zbraň a potom sú uvedené typy palných zbraní. Nasleduje vysvetlenie obrazového šumu, ktorý môže do značnej miery ovplyvniť výsledok detekcie, a sú uvedené spôsoby, ako ho redukovať. Ďalej sú definované algoritmy obrazovej detekcie, ktoré pracujú na báze neurónových sietí - konvolučné neurónové siete (CNN) a Single Shot Multibox Detector (SSD). Vysvetlené sú aj klasické algoritmy pre detekciu v obraze ako HOG+SVM a SURF. V ďalšej časti sú uvedené použité knižnice a softvér. Nasleduje experimentálna časť, kde uvedieme navrhnutý postup riešenia a databázu. Pre detekciu boli použité algoritmy HOG+SVM, SURF a SSD. Následne sú uvedené výsledky testov na databáze a videu, čomu nasleduje zhrnutie a možnosti rozšírenia do budúcnosti.

Abstract

This thesis is focused on the topic of firearms detection in images. In the theoretic section, the explanation of the term firearm is covered, along with the definition of the most prevalent firearm categories. Then the concept of image noise and the ways it can hinder image detection is covered, along with ways of reducing it. Next, algorithms of image detection are introduced - first those which operate on the basis of neural nets - such as Convolutional Neural Nets and Single Shot Multibox Detection. The next section discusses classic algorithms of object detection such as HOG+SVM and SURF. After that, information on the used libraries and software is provided. The experimental part covers the designed algorithm and database. For detection, the HOG+SVM, SURF and SSD algorithms were used. All the algorithms are tested on the database and, if possible, on video. A final evaluation is provided, along with possible future development options.

Kľúčové slová

strelné zbrane, detekcia objektov, SURF, SSD, HOG, SVM, Speeded-Up Robust Features, Single Shot Multibox Detector, Histogram of Oriented Gradients, Support Vector Machines, detekcia strelných zbraní, neurónové siete, konvolučné neurónové siete, CNN

Keywords

firearms, object detection, SURF, SSD, HOG, SVM, Speeded-Up Robust Features, Single Shot Multibox Detector, Histogram of Oriented Gradients, Support Vector Machines, firearm detection, neural nets, convolutional neural nets, CNN

Citácia

DEBNÁR, Pavol. *Detekce palných zbraní v obrazu.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Dvořák

Detekce palných zbraní v obrazu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Dvořáka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Pavol Debnár
14. mája 2019

Poďakovanie

Chcel by som poďakovať Ing. Michalovi Dvořákovi za pomoc a odborné rady pri riešení tejto bakalárskej práce. Zároveň by som chcel poďakovať mojej rodine, ktorá ma podporovala počas celého štúdia.

Obsah

Kapitola 1

Úvod

V súčasnej dobe sa stále vykonáva značné množstvo zločinov pomocou strelných zbraní. Kým na Slovensku a v Česku počty týchto zločinov nie sú až také vysoké, v iných končinách sveta je to, žiaľ, opačne. To však nemení nič na fakte, že na prevenciu týchto zločinov sa míňa každoročne veľa peňazí a ľudskej sily. Zároveň sa rapídne modernizuje technika a zvyšuje sa počet kamier, či už bezpečnostných alebo mobilných. Tým pádom máme množstvo nahraných záznamov daných zločinov. Tieto záznamy však často bývajú dlhé niekoľko hodín a náročné na všímanie si všetkých detailov. Práve preto by bol potrebný program, ktorý by túto prácu používateľovi zjednodušil, alebo priamo vyriešil za neho.

Vo svete sa čoraz častejšie na podobné problémy detekcie využíva strojové učenie. Príkladom efektívneho nástroja na detekciu tvárí, alebo chodcov je algoritmus Histogram of Oriented Gradients spolu so Support Vector Machines (HoG+SVM). Ďalším efektívnym spôsobom detekcie sú konvolučné neurónové siete, ktorých efektivita sa každoročne potvrdzuje napríklad na ImageNet Large Scale Visual Recognition Challenge, kde je cieľom klasifikovať veľké množstvo fotografií až do 1000 kategórií.

Preto sa v tejto práci pokúšame vytvoriť detektor zbraní v obraze pomocou súčasných algoritmov detekcie. Na riešenie tejto práce sme si vybrali algoritmy HoG+SVM, ďalej Speeded-Up Robust Features (SURF) a Single Shot Multibox Detection (SSD), ktorý je založený na princípoch konvolučných neurónových sietí. Keďže každý z týchto algoritmov je zložitý, v rámci riešenia tejto práce do nich nebudeme zasahovať, ale budeme sa zaoberať iba hľadaním parametrov, s ktorými by dané algoritmy fungovali na detekciu zbraní. Súčasťou riešenia je aj vytvorená anotovaná databáza, ktorá slúži na natrénovanie jednotlivých algoritmov.

Ak pomocou zvolených algoritmov dokážeme zostrojiť funkčný detektor zbraní, tak by záujemcami o výsledok tejto práce mohli byť napríklad policajné, alebo iné bezpečnostné zložky, ktoré často musia prechádzať záznamy ozbrojencov a vyhľadávať ich vybavenie.

Kapitola 2

Teoretická časť

Táto kapitola slúži na oboznámenie čítateľa s teoretickými prvkami tejto práce. Najprv je definovaný pojem palná zbraň, spolu s rozdelením do skupín, ktoré sa používa ako v reálnom svete, tak aj v tejto práci. Ďalej bude vysvetlený obrazový šum, ako jav, ktorý môže do značnej miery negatívne ovplyvniť detekciu. Následne bude venovaná pozornosť jednotlivým detekčným algoritmom. Prvá časť týchto algoritmov je založená na neurónových sieťach a následne sú uvedené klasické algoritmy detekcie pre detekciu v obraze.

2.1 Palné zbrane

Pre účel jednoduchšieho pochopenia a zasvätenia čitateľa do tématiky palných zbraní si v tejto kapitole uvedieme základné vysvetlenie pojmu palná zbraň, delenie palných zbraní a ich základné časti.

Zákon definuje palnú zbraň ako zbraň, ktorej funkcia je založená na okamžitom uvoľnení energie pri chemickej reakcii [3]. Ďalej môžeme podľa [14] palné zbrane klasifikovať podľa druhu, spôsobu držania pri streľbe, ráže, alebo stupňa automatizácie. Ku každej kategórií si niečo povieme.

2.1.1 Klasifikácia palných zbraní podľa druhu

Medzi hlavné druhy palných zbraní podľa [14] patria:

- **Pištole** - krátke zbrane s pevnou, alebo pohyblivou hlavňou, ktorá obsahuje nábojovú komoru. Môžu mať jednu, alebo niekoľko hlavní, môžu byť jednoranové, opakovacie, samonabíjacie alebo automatické.
- **Revolvery** - krátke zbrane s pevnou hlavňou a otáčajúcim sa valcom, ktorý obsahuje nábojové komory, ktoré sa jednotlivými výstrelmi postupne striedajú v hlavni.
- **Pušky** - dlhé zbrane s drážkovaným vývrtom. Z hľadiska automatizácie môžu byť jednoranové, opakovacie, samonabíjacie, alebo automatické, preto do tejto kategórie spadajú ako guľovnice a ostreľovacie pušky, tak i karabíny a útočné pušky.
- **Brokovnice** - dlhé aj krátke zbrane s hladkým vývrtom a jednou, alebo viacerými brokovými hlavňami.

- **Kombinované zbrane** - typicky dlhé lovecké zbrane s minimálne dvoma hlavňami s rôznym kalibrom.
- **Samopaly** - ručné automatické zbrane používajúce náboje pištoľového kalibru.
- **Guľomety** - automatické zbrane, konštruované na náboj puškového, alebo väčšieho kalibru. Z hľadiska mobility môžu byť ručné alebo upevnené.
- **Granátomety** - zbrane určené na streľbu špeciálnych granátov. Podľa konštrukcie môžu byť ručné, podvesné, alebo lafetované.
- **Mínomety** - špeciálne vojenské zbrane určené pre vystreľovanie delostreleckých mín.

2.1.2 Klasifikácia zbraní podľa ostatných kritérií

Podľa spôsobu držania pri streľbe rozdeľujeme zbrane na krátke a dlhé. Toto delenie platí pre ručne držané zbrane [14].

- **Krátke zbrane** - ich konštrukcia predpokladá streľbu s použitím jednej ruky. Do tejto kategórie spadajú pištole a revolvery.
- **Dlhé zbrane** - sú konštruované tak, aby strelec pri streľbe používal obidve ruky a mal zbraň opretú o rameno. Pušky, brokovncie, samopaly, kombinované zbrane, PDW a prenosné guľomety spadajú do tejto kategórie.

Ďalej môžeme zbrane deliť podľa **ráže**. Toto delenie je však závislé od kontextu a je diskutovateľné. V armádach sa používa napríklad delenie na strelecké a delostrelecké zbrane, kde sa za hranicu považuje ráž 20 mm. Toto delenie môže byť aplikované aj na samotné druhy zbraní. Napríklad veľkorážový guľomet je v praxi guľomet ráže 10 mm až 20 mm, v športovnej streľbe je za veľkorážovú pištoľ považovaná pištoľ o ráži 7.62 mm až 12 mm [14].

Posledná klasifikácia je podľa stupňa automatizácie, kde rozlišujeme:

- **Jednoranové a opakovacie** - kde strelec zasúva náboj zo zásobníka do nábojovej komory a ručne uzamyká záver.
- **Čiastočne automatizované** - u týchto zbraní sú privedenou energiou ovládané iba niektoré úkony - vyhodenie nábojnice, alebo otváranie záveru.
- **Samonabíjacie** - pri týchto zbraniach je automaticky vykonaný celý funkčný cyklus, okrem spustenia. Do tejto kategórie spadajú pušky a pištole, u ktorých je potreba zbraň po každom výstrele znovu zamieriť.
- **Automatické** - majú celý funkčný cyklus automatizovaný a zbraň strieľa, pokým strelec neuvoľní spúšť. Medzi tieto zbrane patria útočné pušky, samopaly a guľomety.

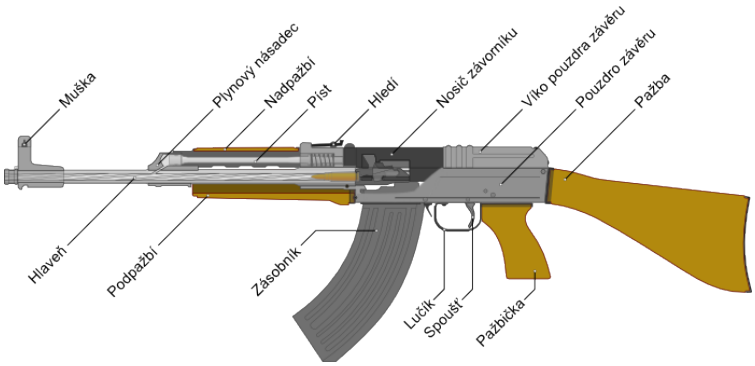
2.1.3 Časti zbraní

V tejto kapitole sa budeme venovať popisu častí jednotlivých druhov zbraní. Nebudeme zachádzať do vnútorných častí kvôli faktu, že táto práca nerieši vnútra jednotlivých zbraní, a je dôležité vedieť iba o externých častiach zbrane, keďže tieto pojmy sa budú vyskytovať v praktickej časti práce.

Obr. 2.1: Pištoľ typu CZ 75 s vyznačenými časťami. Prevzaté z [5].

Podľa [40] majú zbrane nasledujúce časti (obr. 2.1, 2.2):

- **Záver** - slúži najmä pre uzamknutie komory pri výstrele, navyše slúži pre strelca ako plocha za ktorú drží zbraň pri naťahovaní záveru - táto činnosť vkladá náboj do nábojovej komory.
- **Hlaveň** - vedie strelu až kým neopustí zbraň, obsahuje drážky, ktoré pridelia náboju rotáciu na zvýšenie presnosti.
- **Muška a cieľnik** - jedným slovom ich označujeme ako mieridlá, slúžia na zamierenie zbrane.
- **Spúšť** - spúšťa spúšťový mechanizmus, ktorý ďalej spúšťa bicí mechanizmus, ktorý má za následok vystrelenie náboja zo zbrane.
- **Zásobník** - jeho funkciou je zásobovať zbraň nábojmi.
- **Kohútik** - po stlačení **spúšte** dopadá na zápalník, ktorý je narazený do zápalky náboja, čím odpáli náboj.
- **Nadpažbie, podpažbie, pažba a pažbička** tvoria úchopové časti zbrane.



Obr. 2.2: Česká útočná puška SA vz. 58 s vyznačenými časťami. Prevzaté z [40].

2.2 Obrazový šum a jeho filtrácia

Jedným z faktorov, ktorý do značnej miery môže ovplyvniť úspešnosť detekcie objektu je obrazový šum. V tejto sekcii si preto prejdeme jeho definíciu, najznámejšie druhy a spôsoby ako ho odstraňovať.

Šum

Obrazovým šumom rozumieme náhodné rozdiely vo farbe, alebo jase obrazu, ktoré nie sú prítomné na zobrazovanom objekte. Môže byť produkovaný senzorom alebo obvodmi daných optických zariadení [8]. Je to všeobecne nežiadaný jav, keďže v súčasnej dobe sa čím ďalej, tým viac odvetví vedy, medicíny, alebo priemyslu zakladá na snímkach, ktoré musia čo najviac odpovedať realite [33]. Z hľadiska detekcie objektov je šum nežiadúci preto, lebo rozrušuje hrany a textúry, podľa ktorých hľadáme daný objekt.

Druhy šumu

Gaussovský šum

Tento druh šumu je zvyčajne vytváraný pri zachytávaní obrazu a je zapríčinený napríklad zlým osvetlením, vysokou teplotou, alebo šumom spôsobeným elektrickými obvodmi [8]. Porovnanie obrazu bez šumu a s gaussovským šumom sa nachádza na obrázku 2.3.

Obr. 2.3: Porovnanie obrazu bez šumu (vľavo) a obrazu s gaussovským šumom (vpravo). Prevzaté z [4].

Hustota pravdepodobnosti je určená podľa normálneho (Gaussového) rozdelenia, teda: $P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$, kde x je jas pixelu, μ je stredná hodnota a σ je štandartná odchýlka [6].

Impulzný šum (salt and pepper noise)

Impulzný šum klasifikujeme ako korupciu obrazu náhodnými čiernymi a bielymi pixelmi. Vzniká napríklad pri chybách A/D konvertéra, alebo bitových chybách pri prenose [8].

Metódy odstraňovania šumu

Lineárne filtre

Lineárne filtre úspešne odstraňujú šum, ale rozmazávajú hrany obrazu. Medzi najznámejšie lineárne filtre patria priemerový (mean) a Gaussov filter. Priemerový filter vezme hodnoty konkrétneho pixelu a jeho okolia a spriemeruje ich. Hodnoty masky Gaussovho filtra sú určené pomocou vzorca $G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$. Táto maska sa následne použije ako maska konvolúcie so zdrojovým obrazom [32]. Ukážka priemerového a gaussovho filtra je v tabuľke 2.1. Aplikácia týchto filtrov sa nachádza v obrázku 2.4.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Tabuľka 2.1: Priemerový filter o veľkosti 3x3 (vľavo) a Gaussový filter o veľkosti 5x5 (vpravo).

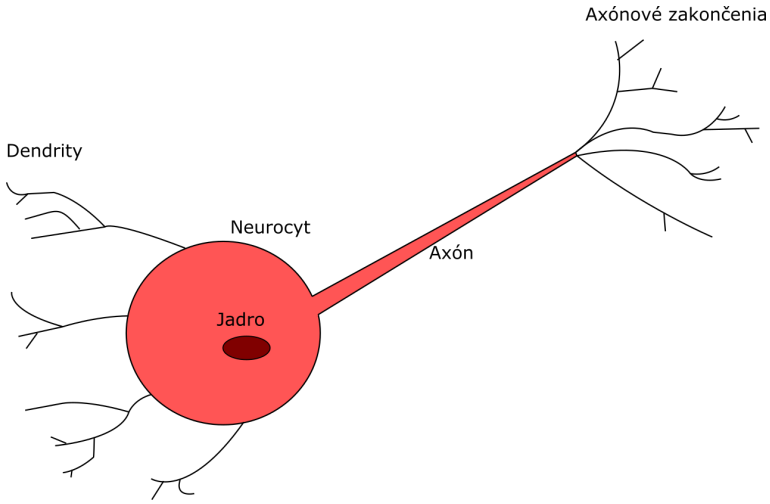
Nelineárne filtre

Na rozdiel od lineárnych filtrov nelineárne filtre pri odstraňovaní šumu nerozmazávajú obraz. Najjednoduchším a najznámejším nelineárnym filtrom je mediánový filter. Pri tomto filtri sa výsledná hodnota pixelu rovná mediánu pixelu zo zdroja a jeho okolia [32]. Aplikácia mediánového filtra sa nachádza na obrázku 2.4.

Obr. 2.4: Obraz s Gaussovským šumom (vľavo hore). Obraz po použití 5x5 Gaussovho filtru (vpravo hore), 5x5 priemerového filtru (vľavo dole), 5x5 mediánového filtru (vpravo dole). Prevzaté z [4].

2.3 Neurónové siete

Neurónové siete sú v súčasnosti veľmi populárnou metódou na riešenie najrozmanitejších problémov, ako napríklad autonómne riadenie motorových vozidiel, predpovede počasia, hranie hier, alebo predpovedanie vývoja akciového trhu. Ďalším takýmto problémom je detekcia objektov v obraze, na ktoré vznikli špecializovanejšie algoritmy, ako napríklad konvolučné neurónové siete alebo Single Shot Multibox Detector [22]. V tejto sekcii si preto vysvetlíme základy funkcie a stavby neurónových sietí a potom sa presunieme do týchto špecializovaných algoritmov na detekciu objektov.

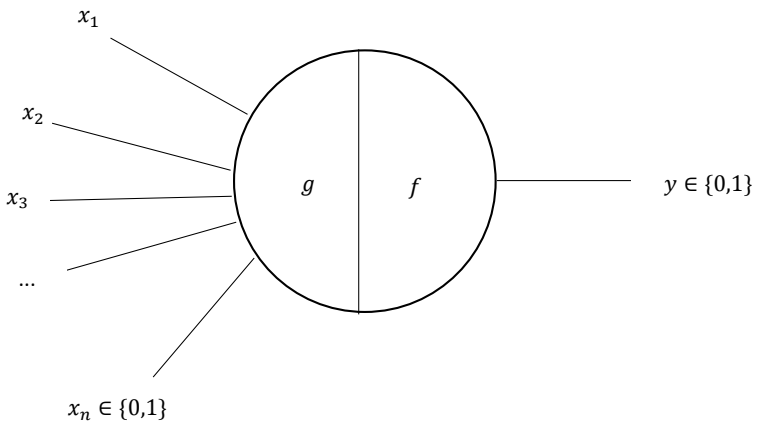


Obr. 2.5: Model biologického neurónu - upravené z [30].

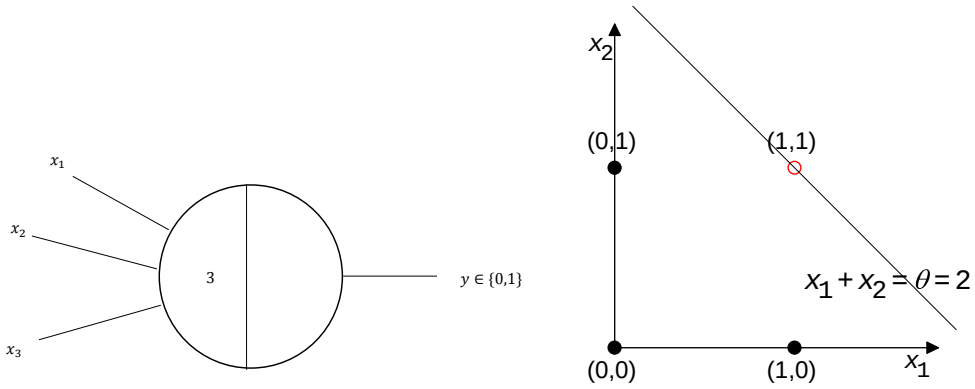
Neurónová sieť je výpočetný model, ktorý bol inšpirovaný ľudským mozgom. Základnou bunkou je neurón. Neurón v ľudskom mozgu sa skladá z dendritov, neurocytu - tela bunky, jadra, axónu a jeho zakončení (obr. 2.5). Dendrity získavajú signály z iných neurónov, neurocyt tieto informácie spracuje, axón prenáša výstup z neurocytu a axónové zakončenia slúžia ako vstup do ďalších neurónov. V roku 1943 sa túto funkcionalitu neurónov podarilo napodobniť vedcom McCullochovi a Pittsovi a vznikol McCulloch-Pittsov neurónový model [30, 18, 32]. McCulloch-Pittsov neurón je zobrazený na obrázku 2.6 a matematicky popísaný rovnicou 2.1:

$$\begin{aligned} g(x) &= \sum_{i=1}^{n} x_i, \\ y &= f(g(x)) = 1 \quad if \quad g(x) \geq \theta \;\; AND \;\; NO \;\; inhibitors, \\ y &= f(g(x)) = 0 \quad if \quad g(x) < \theta, \end{aligned} \tag{2.1}$$

kde $x_1 ... x_n$ sú vstupy, ktoré majú hodnoty 0 alebo 1. Vstupy môžu byť inhibitory alebo excitátory. Každý vstup má rovnakú váhu - rovnakú mieru dôležitosti. g je súčtom všetkých excitátorov. Ak má niektorý inhibítor hodnotu 1, výstup je automaticky 0. θ je prahová hodnota, ktorú musí $g(x)$ dosiahnuť na to, aby výstup f bol 1 [18, 32].

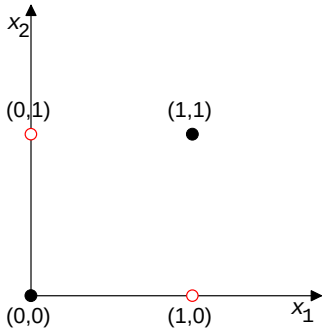


Obr. 2.6: McCulloch-Pittsov neurónový model - upravené z [18].



Obr. 2.7: Logická funkcia AND je lineárne separovateľná - upravené z [18].

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Obr. 2.8: Logická funkcia XOR je lineárne neseparovateľná - upravené z [18].

Pomocou týchto neurónov sa dajú riešiť niektoré základné logické funkcie, ako napríklad AND, OR, NOR, NOT, ale musia byť lineárne separovateľné [18, 32]. Funkcia AND je zobrazená na obrázku 2.7 a lineárne neseparovateľná funkcia XOR je na obrázku 2.8.

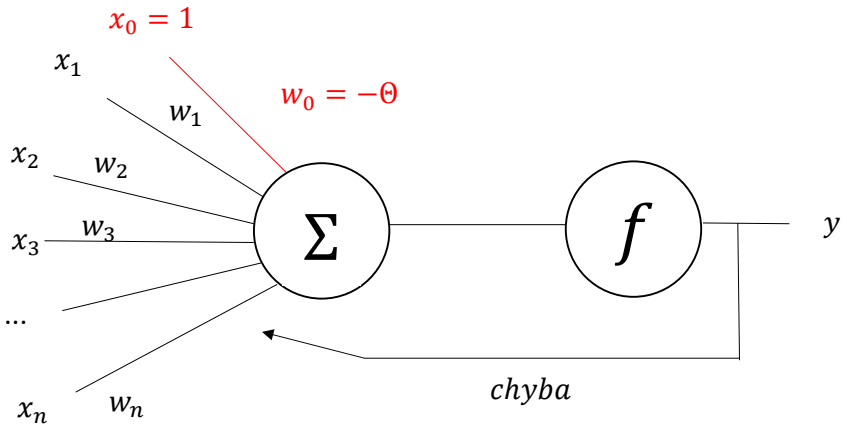
Tento model neurónu je obmedzený v nasledujúcich veciach:

- Dokáže pracovať iba s boolean hodnotami
- θ musí byť vždy určená
- Každý vstup má rovnakú váhu
- Nedokáže spracovať lineárne neseparovateľné funkcie

Kvôli týmto nedostatkom vznikol vylepšený model s názvom *Perceptrón*.

Perceptrón

V roku 1957 prišiel Frank Rosenblatt s vylepšením na McCulloch-Pittsovým neurónom. Perceptrón je všeobecnejší výpočtový model, ktorý odstránil niektoré obmedzenia M-P neurónu. Zaviedli sa váhy (w) ako miery dôležitosti, ktorých hodnoty sa daný perceptrón môže naučiť. Jednotlivé vstupy (x) už nie sú obmedzené na boolovské hodnoty, ale môžu sa použiť reálne čísla [2, 19]. Schéma tohoto modelu je na obrázku 2.9.



Obr. 2.9: Model perceptrónu, upravené z [2, 19].

Jednotlivé vstupy sú vynásobené svojou váhou a následne sú sčítané. Váha w_0 je predsudok, ktorý určuje základnú hodnotu, ktorá musí byť prekonaná na spustenie neurónu, kde θ je prahová hodnota. Jednoducho to môžeme vysvetliť na analógií futbalového fanúšika, ktorý sa rozhoduje nad tým, ktoré zápasy pozerať: Zapálený fanúšik chce pozerať čo najviac zápasov, a preto bude menej ovplyvnený inými vstupmi, ako napríklad či hrá jeho obľúbený klub, alebo či sa hrá liga majstrov, tak jeho prahová hodnota θ bude nízka alebo nulová. Naopak, výberavý fanúšik bude chcieť pozerať iba svoj obľúbený klub a iba významné zápasy, takže jeho prahová hodnota bude vysoká ($\theta = 2$). Následne tento súčet vstupuje do aktivačnej funkcie, ktorá určí konečnú hodnotu neurónu [2, 19].

Aktivačná funkcia určuje výstup perceptrónu. Medzi najvýznamnejšie funkcie patria kroková funkcia (step function), signum (sign function), sigmoid a ReLU. Pri krokovej

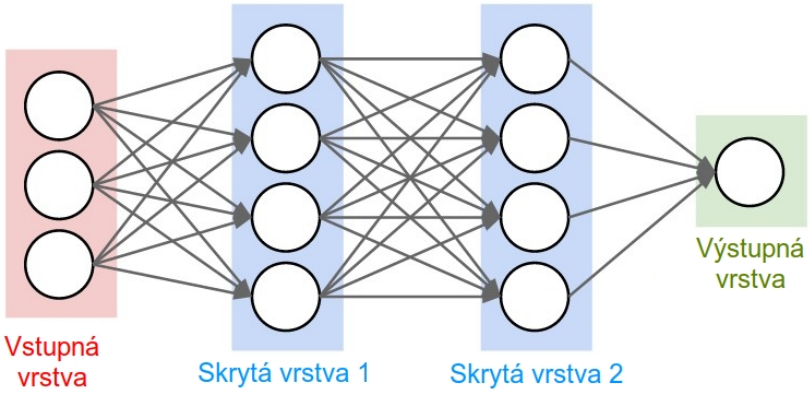
funkcii musí byť vstup väčší ako daný krok (t), inak je záporný. Signum je záporné pre záporný vstup, 0 pre 0, a kladné pre kladný vstup. Výhodou sigmoidu je, že výstupom je reálne číslo v rozpätí $< 0,1 >$, tak je môžné ho interpretovať ako pravdepodobnosť daného javu. V súčasnosti je ale najviac používaná funkcia ReLU, ktorá je určená ako $y = max(0, x)$. Podľa toho, či je výstup správny, alebo nesprávny sa následne vypočíta chyba, pomocou ktorej sa aktualizuje hodnota jednotlivých váh na vylpšienie klasifikácie [2, 19].

Samotný perceptrón však stále nedokáže vyriešiť problém, ktorý nie je lineárne separovateľný. Tento problém sa vyriešil zapojením viacerých neurónov za sebou a algoritmom spätnej propagácie (backpropagation).

2.3.1 Štruktúra neurónovej siete

Moderné neurónové siete sa skladajú z viacerých vrstiev. Na začiatku je vstupná vrstva, ktorá reprezentuje informácie, ktoré vstupujú do neurónovej siete. Nasleduje určitý počet skrytých vrstiev, kde sa dané informácie generalizujú. V praxi sa využívajú najčastejšie siete s jednou, alebo dvomi skrytými vrstvami. Na základe dát zo skrytých vrstiev je určený výstup vo výstupnej vrstve [31, 17]. Model neurónovej siete sa nachádza na obrázku 2.10.

Ďalej môžeme neurónové siete deliť podľa zapojenia neurónov. V priamoväzbových neurónových sieťach tečú informácie z jednej vrstvy do nasledujúcej s tým, že výstup jednej vrstvy nemôže ovplyvniť tú istú, alebo predchádzajúcu vrstvu. Naopak v rekurentných (cyklických) neurónových sieťach sa vyskytujú vrstvy so spätnými väzbami do inej vrstvy, alebo neuróny, ktoré majú spätnú väzbu samé so sebou [17].



Obr. 2.10: Štruktúra neurónovej siete. Upravené z [1].

2.3.2 Učenie neurónovej siete

Neurónová sieť je adaptívny systém, to znamená, že môže meniť svoju vnútornú štruktúru podľa informácií, ktoré cez daný systém prebiehajú. Toto je dosiahnuté menením hodnôt váh, spomínaných v predchádzajúcej kapitole. K zmene váh dochádza vždy pri dosiahnutí nesprávneho výsledku. Podľa prístupu poznáme 3 techniky učenia - učenie s učiteľom, učenie bez učiteľa a posilňované učenie [31].

Učenie s učiteľom

Tento prístup využíva prístup učiteľa, ktorý je múdrejší ako sieť. Učiteľ definuje trénovaciu sadu - sadu vstupov a k nim príslušných výstupov. Pre dané vstupy sú následne vypočítané výstupy neurónovou sieťou. Pomocou rozdielu vypočítaných výstupov s výstupmi určených učiteľom môžeme vypočítať chybu, ktorá sa následne využije na upravenie váh neurónovej siete [31, 17].

Učenie bez učiteľa

Pri učení bez učiteľa nepoznáme hodnotu výstupov pre konkrétne vstupy. Neurónová sieť musí sama objaviť jednotlivé príznaky, podľa ktorých by mohla vznikať nejaká korelácia medzi dátami [31, 17].

Posilňované učenie

Pri posilňovanom učení máme pre každú vstupnú hodnotu definovanú hodnotu výstupnú. Častokrát je ale pre veľa sekvencií vstupov definované iba, či je výsledný výstup správny alebo nesprávny. Preto je tento prístup založený na pokusoch a omyloch. Príklad - robot má niekoľko ťahov na to, aby sa dostal do cieľa cez prekážkovú dráhu. Po náhodnej sekvencii pohybu do určitých smerov buď nabúral, alebo sa dostal do cieľa [31, 17].

Algoritmus spätnej propagácie (backpropagation)

Tento algoritmus slúži na trénovanie dopredných neurónových sietí. Na začiatku trénovania sú váhy jednotlivých neurónov určené náhodne. Následne zoberieme sadu vstupov a vypočítame výstupy. Na základe vypočítaných a očakávaných výstupov môžeme pomocou stratovej funkcie vypočítať stratu (chybu). Vypočítaná chyba sa následne spätne propaguje, teda začína na výstupnej vrstve a propaguje sa na všetky neuróny v skrytých vrstvách, pomocou ktorých bol výstup vypočítaný. Chyba sa rozloží pre jednotlivé neuróny v pomere, v akom neuróny prispeli ku výstupu. Nasleduje gradientný zostup - výpočet derivácie stratovej funkcie, pomocou ktorej vieme ako znížiť chybovosť pre každý neurón [35, 32].

2.3.3 Konvolučné neurónové siete (CNN)

Konvolučné neurónové siete sú algoritmus strojového učenia, ktorý je v praxi používaný na klasifikáciu obrazu. Klasickými neurónovými sieťami by sa obraz síce dal klasifikovať, ale to by viedlo k mnohým vstupným neurónom a spojeniami medzi nimi, čo by sťažovalo tréning siete [16]. Oproti štandartným neurónovým sietiam majú konvolučné neurónové siete predspracovanie, ktoré sa tvorí z konvolučných, pooling, a rektifikačných vrstiev, ktoré znižujú počet vstupov do plne prepojených vrstiev, a teda celkovú zložitosť [1].

Konvolučné vrstvy

Konvolučná vrstva obsahuje sadu filtrov, ktorých konkrétne hodnoty sa daná konvolučná sieť naučí. Programátor musí ale definovať šírku, výšku a hĺbku filtrov. Hĺbkou označujeme počet filtrov. Vstupom do konvolučnej vrstvy je obraz určitých rozmerov. Počiatočný vstup má hĺbku 3 - kanály RGB. Zo vstupného obrazu sa pomocou operácie konvolúcie s filtrami vytvoria jednotlivé aktivačné mapy, ktoré tvoria výsledok tejto vrstvy. Pre lepšie výsledky určujeme ešte hodnoty striedy a zero-paddingu. Striedou myslíme počet pixelov, o ktorý

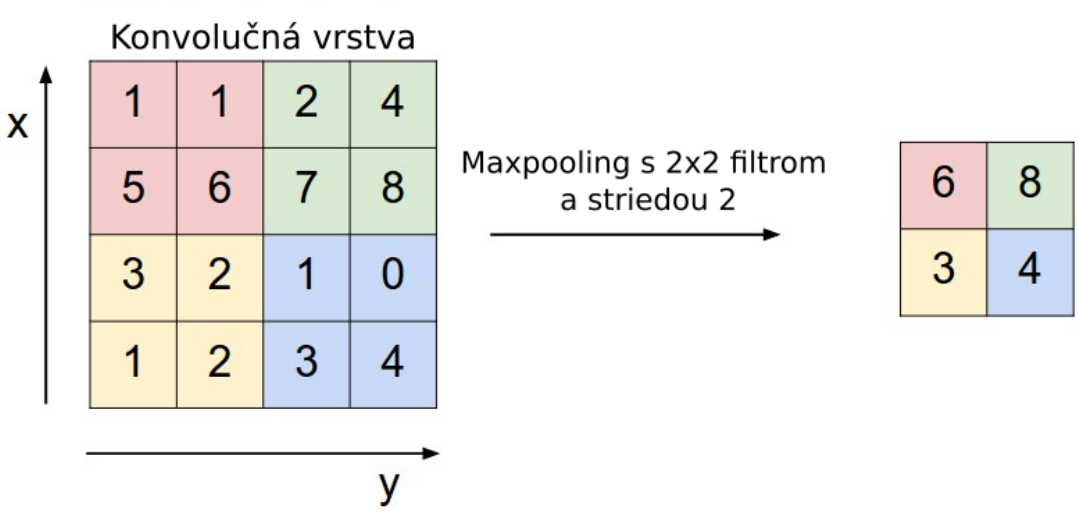
sa posunie filter pri prechádzaní po obraze. Zero-padding je hodnota, ktorá určuje koľko pixelov s nulovou hodnotou je umiestnených okolo obrazu. Výslednú veľkosť aktivačnej mapy môžeme zistiť podľa nasledujúcej rovnice:

$$(W - F + 2P)/S + 1, \tag{2.2}$$

kde W je dĺžka vstupného obrazu (predpokladáme štvorcový vstup a filtre), F je dĺžka filtru, P je počet pridaných 0, a S je strieda pri pohybe filtra [16, 1].

Pooling vrstvy

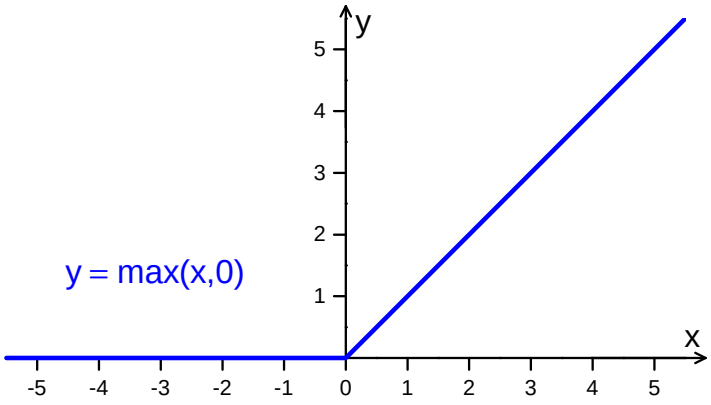
Pooling vrstva sa zvyčajne nachádza za konvolučnou vrstvou, prípadne za sériou konvolučných vrstiev. Cieľom pooling vrstvy je znížiť celkový rozmer vstupu. Pooling sa aplikuje na všetky vrstvy vstupu. Najbežnejší druh tejto vrstvy je maxpooling s veľkosťou a striedou 2, kvôli jeho efektívnosti. Poznáme aj iné prístupy, ako napríklad priemerový pooling. Operácia maxpooling vybere maximum z oblasti (2x2) vstupu a uloží ho ako nový pixel, potom sa posunie o striedu. Výsledkom je teda zmenšený obraz [1]. Príklad operácie maxpooling sa nachádza na obrázku 2.11.



Obr. 2.11: Operácia maxpooling. Upravené z [1].

Rektifikačné vrstvy

Rektifikačné vrstvy aplikjú na daný vstup funkciu ReLU (obr. 2.12). Táto funkcia odstraňuje zo vstupu negatívne hodnoty a bolo dokázané, že urýchľuje učenie [1, 16].



Obr. 2.12: Funkcia ReLU

Celková architektúra

Jednotlivé vrstvy sa skladajú postupne za sebou. Poznáme však určité konvencie, pri ktorých bolo dokázané, že fungujú najlepšie. Preto má väčšina konvolučných neurónových sietí vrstvy rozložené nasledovne:

$$INPUT \rightarrow [[CONV \rightarrow RELU] * N \rightarrow POOL?] * M \rightarrow [FC \rightarrow RELU] * K \rightarrow FC, \quad (2.3)$$

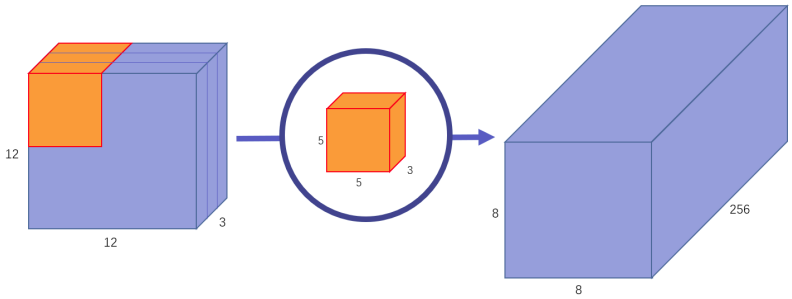
kde INPUT je vstupná vrstva, CONV je konvolučná vrstva, RELU je rektifikačná vrstva, POOL je pooling vrstva, FC je plne spojená vrstva. $N, M, K \geq 0$ a zvyčajne $N \leq 3$ a $K < 3$. Tieto čísla označujú opakovanie daných vrstiev [1].

Ako príklad si môžeme uviesť konvolučnú neurónovú sieť LeNet, ktorá bola vytvorená v roku 1998 za cieľom klasifikácie napísaných číslic. Vstupom je šedotónový obraz s rozmermi 32x32x1, ktorý je na prvej konvolučnej vrstve filtrovaný pomocou šiestich 5x5 filtrov so striedou 1, čo zmení rozmer obrazu na 28x28x6. Na druhej vrstve sa vykoná operácia priemerového poolingu s filtrom 2x2 a striedou 2. Obraz je teda zmenený na veľkosť 14x14x6. Na tretej vrstve sa vykonáva konvolúcia s šestnástimi filtrami 5x5 a striedou 1, výsledkom ktorej je obraz 10x10x16. Vrstva číslo štyri je rovnaká ako druhá vrstva - priemerový pooling s filtrom veľkosti 2x2 a striedou 2, čoho výsledkom je obraz 5x5x16. Piata vrstva je plne prepojená so štvrtou so 120 neurómni, šiesta obsahuje 84 neurónov. Výstupná vrstva má 10 rôznych hodnôt - ktoré reprezentujú čísla 0 až 9 [20].

2.3.4 Mobilenet

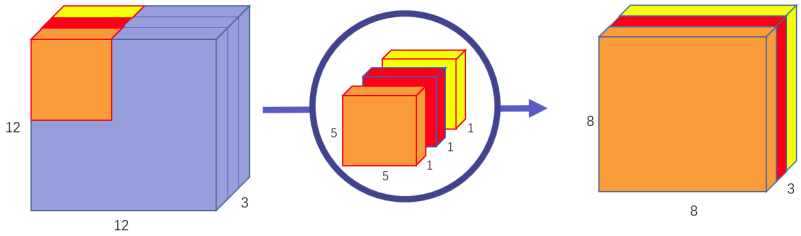
Keďže sa v tejto práci používa konvolučná neurónová sieť Mobilenet (alebo aspoň časť z nej), je vhodné, aby bola popísaná. Sieť Mobilenet bola predstavená v roku 2017 a patrí medzí nové a inovatívne CNN. Zároveň dosahuje dobré výsledky - napríklad na datasete ImageNet má presnosť 70.6% Jej cieľom je zníženie celkovej výpočtovej náročnosti, kvôli tomu, aby bola použiteľná aj na mobilných zariadeniach. Zníženie výpočtovej náročnosti sa dosahuje pomocou rozdelenia operácie konvolúcie na dve: HÍbková konvolúcia (Depthwise convolution) a Bodová konvolúcia (Pointwise convolution) [11]. Tieto operácie si porovnáme s klasickou konvolúciou na príklade:

Máme vstupný obraz veľkosti 12x12x3. Veľkosť konvolučného filtra je 5x5x3 so striedou 1, po konvolúcií je výstup 8x8x1. Pokiaľ chceme výsledok o veľkosti 8x8x256, použijeme 256 rôznych konvolučných jadier veľkosti 5x5x3. Ilustrované na obrázku 2.13.



Obr. 2.13: Klasická operácia konvolúcie, prevzaté z [37].

Pri Mobilenete je prvým krokom hĺbková konvolúcia(obr. 2.14). Pri tejto operácii použijeme počet konvolučných jadier, ktorý je identický s hĺbkou vstupového obrazu. Každý filter prejde jednu vrstvu hĺbky vstupového obrazu. Pre tento príklad to sú tri 5x5 jadrá. Výsledkom je teda výstup o veľkosti 8x8 s hĺbkou 3.

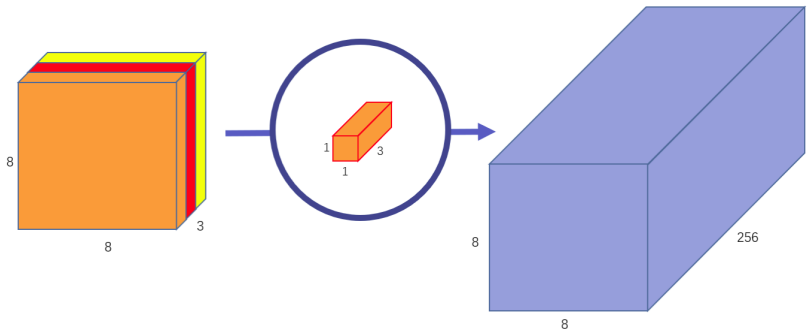


Obr. 2.14: Hĺbková konvolúcia, prevzaté z [37].

Nasleduje bodová konvolúcia. Tá prechádza výstupom z predchádzajúcej operácie a používa konvolučné jadro s veľkosťou 1x1 a rovnakou hĺbkou ako výstup predošlej operácie. V tomto príklade to je 1x1x3, pričom výsledkom je výstup o veľkosti 8x8x1. Následne sa použije toľko konvolučných jadier tejto veľkosti, aby sme dosiahli požadovanú hĺbku - v tomto príklade to je 256 jadier. Bodová konvolúcia pre tento príklad sa nachádza na obrázku 2.15.

Pri tradičnej konvolúcii použijeme v príklade 256 5x5x3 jadier, ktoré sa musia pohnúť 8x8 krát. Spolu to je 1 228 800 operácií násobenia. Operácia násobenia je výpočetne náročná, preto je najlepšie ju používať čo najmenej. Pri hĺbkovej konvolúcii máme tri 5x5x1 jadrá, ktoré sa použijú 8x8 krát. Výsledkom je 4 800 operácií násobenia. Bodová konvolúcia využíva 256 1x1x3 jadier, ktoré sa pohnú 8x8 krát, čo robí 49 152 násobení. Spolu sa teda použije pomocou týchto dvoch konvolúcií iba 53 952 násobení [37].

Táto redukcia výpočtov tvorí silnú stránku siete Mobilenet. Avšak, tým sa znižuje celkový počet naučiteľných parametrov, takže sieť nemusí byť vhodná na riešenie rozsiahlych problémov [37]. V tabuľke na obrázku 2.16 je uvedená architektúra tejto siete.



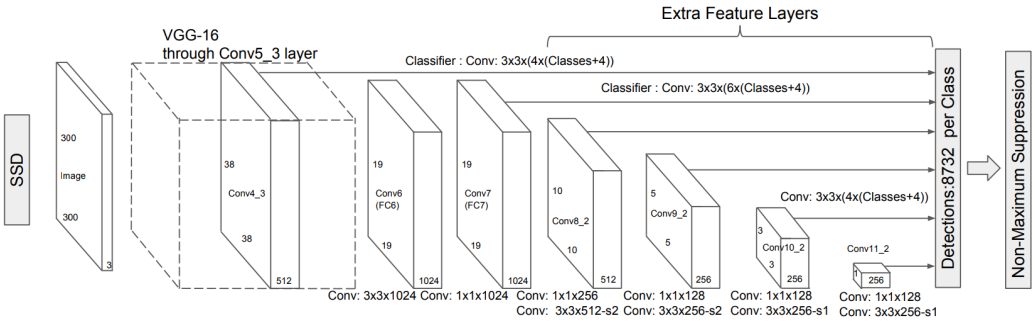
Obr. 2.15: Bodová konvolúcia, používa sa 256 jadier o veľkosti 1x1 s hĺbkou 3. Prevzaté z [37].

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Obr. 2.16: Architektúra Mobilenet. Značka dw označuje hĺbkovú konvolúciu. Prevzaté z [11].

2.3.5 Single Shot Multibox Detection (SSD)

SSD je jedným z modelov pre detekciu objektov v obraze, ktoré si momentálne užívajú najväčšiu popularitu a sú stále rozvíjané. Je založený na doprednej neurónovej sieti, ktorej výsledkom je kolekcia pevnej veľkosti, ktorá obsahuje ohraničené priestory (bounding boxes) a ich ohodnotenie pre hľadané triedy. Finálnym krokom je funkcia Non-Max Supression, ktorá pre viacero prekrývajúcich sa okien (ktoré majú spoločnú oblasť väčšiu ako 50%), vybere okno s najväčším skóre pre danú triedu [22].

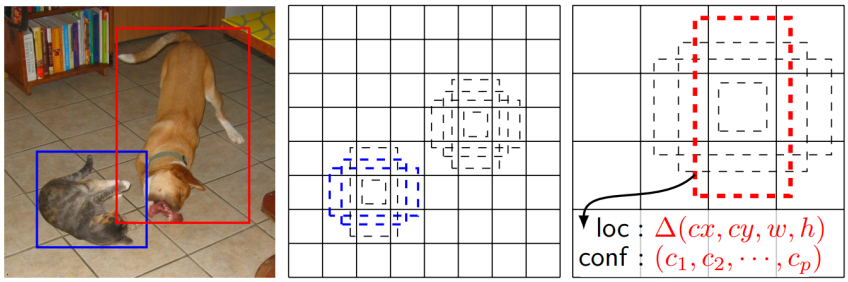


Obr. 2.17: Architektúra Single Shot Detection. Prevzaté z [22].

Prvé vrstvy siete (obr. 2.17) sa nazývajú Base network a typicky sú založené na architektúre, ktorá je určená klasifikáciu obrazov - napríklad sa používajú konvolučné vrstvy zo siete VGG16. Ďalej na base network naväzuje podporná štruktúra, ktorá obsahuje postupne sa zmenšujúce konvolučné vrstvy a tým umožňuje detekciu v rôznych mierkach.

Každá vrstva príznakov môže produkovať sadu predikcií pre detekciu pomocou použitia konvolučných filtrov. Pre vrstvu príznakov o veľkosti $m \times n$ o šírke p sa používa konvolúcia s jadrom o veľkosti $3 \times 3 \times p$, ktorá produkuje buď skóre pre danú triedu, alebo offset pre koordináty predvolenej ohraničenej oblasti (bounding box).

Ďalej sa každá mapa príznakov rozdelí na bunky (obr. 2.18), ktorých veľkosť závisí od mierky, v ktorej sa nachádza, a každej bunke je priradená sada predvolených ohraničených oblastí. Pre každú bunku je následne vypočítané skóre pre každú triedu a zároveň je predikovaný posun voči pôvodným ohraničeným oblastiam, zaručujúci lepšiu detekciu. [22]



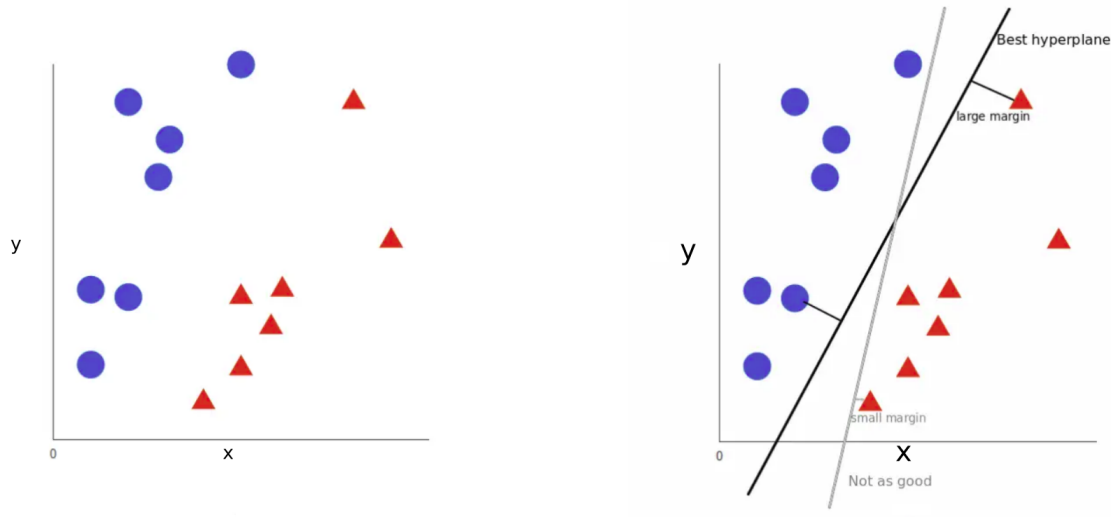
Obr. 2.18: Princíp SSD. (Vľavo) SSD na tréning potrebuje iba vstupný súbor, v ktorom sú anotované objekty. Každá mapa príznakov (stred a vpravo) je rozdelená na bunky, pričom počet buniek je daný mierkou. Pre každú bunku sa používajú 4 rôzne bounding boxy. Pre každú predikciu je zároveň odhadnutý posun ohraničenej oblasti a skóre pre danú triedu. Na obrázku v strede je zhoda pre 2 predvolené ohraničené oblasti a na obrázku vpravo je zhoda pre jednu ohraničenú oblasť. Prevzaté z [22].

2.4 Histogram orientovaných gradientov (HoG)

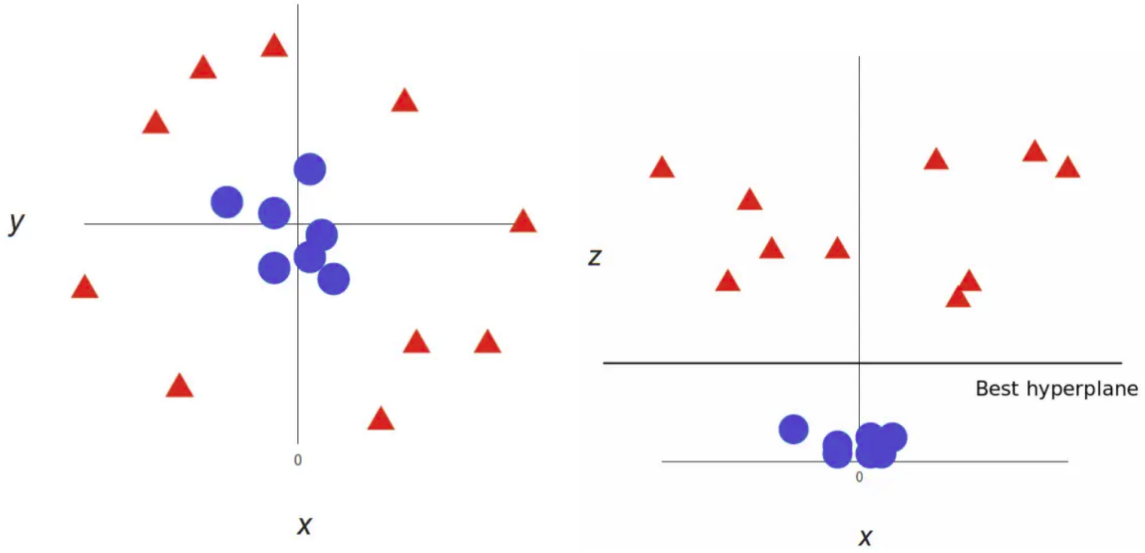
Histogram orientovaných gradientov je algoritmus pre extrakciu príznakov z obrazu. Bol predstavený v roku 2005 ako časť algoritmu pre detekciu chodcov. Zakladá sa na výpočte gradientov, čo sú vektory určujúce smer a veľkosť zmeny v obraze. Pre každý pixel sa vypočíta horizontálny (g_x) a vertikálny (g_y) gradient. Na základe týchto hodnôt môžeme vypočítať veľkosť výsledného gradientu pomocou vzorca $g = \sqrt{g_x^2 + g_y^2}$ a jeho orientáciu $\theta = \arctan(\frac{g_y}{g_x})$. Následne sa vstupný obraz rozdelí do buniek určitej veľkosti a pre každú bunku je vypočítaný histogram gradientov. Histogram je popísaný pre jednotlivé veľkosti uhlov napr. 0-20, 21-40, atď. a jednotlivé hodnoty sú určené veľkosťami gradientov. Ďalším krokom je normalizácia histogramov buniek so svojim okolím. Napríklad pri bunkách 8x8 sa vykoná normalizácia o veľkosti štyroch takýchto buniek, teda bunky o veľkosti 16x16. Touto operáciou dosiahneme invariantnosť voči jasu. Výstupom je konkatenácia hodnôt histogramov zo všetkých buniek obrazu. Tento algoritmus nie je invariantný voči rotácií [23, 32].

2.5 Support Vector Machines (SVM)

Support Vector Machine je algoritmus strojového učenia s učiteľom, ktorý sa najčastejšie používa pre klasifikáciu prvkov. Cieľom tohoto algoritmu je nájsť vhodnú nadrovinu pre separáciu prvkov odlišných tried v n-dimenzionálnom priestore (obr. 2.19). Optimálna nadrovina má čo najväčší priestor (margin) medzi bodmi jednotlivých tried. Pre nájdenie takejto nadroviny používame hraničné body daných tried - pomocné vektory. SVM môžeme použiť pre lineárne separovateľné ako aj lineárne neseparovateľné problémy. Pre prípady kedy nemôžeme použiť lineárny separátor, používame jadrovú transformáciu (kernel trick), čo je funkcia, ktorá zvýši počet dimenzií daného priestoru takým spôsobom, aby boli jednotlivé triedy separovateľné [32, 29]. Kernel trick je ilustrovaný na obrázku 2.20.



Obr. 2.19: (Vľavo) Máme lineárne separovateľný priestor dvoch tried, medzi ktorými potrebujeme nájsť optimálnu nadrovinu. (Vpravo) SVM zvolí ideálnu nadrovinu tak, aby bola vzdialenosť medzi prvkami jednotlivých tried (margin) čo najväčšia. Šedou je zároveň vyznačená nadrovina, ktorá nie je ideálna. Prevzané z [34].



Obr. 2.20: (Vľavo) Máme dve lineárne neseparovateľné triedy. (Vpravo) Môžeme použiť kernel trick, kde nahradíme os y osou z, ktorá je definovaná ako $z = x^2 + y^2$. Po jeho použití sú dané triedy už lineárne separovateľné. Prevzané z [34].

2.6 Speeded-Up Robust Features (SURF)

V roku 2006 bol predstavený na Európskej konferencií počítačového videnia (European Conference on Computer Vision - ECCV). Jeho cieľom bolo vylepšiť algoritmus Scale-Invariant Fearute Transform (SIFT). Autori sa domnievali, že hlavnou slabinou SIFTu bola rýchlosť. Algoritmus SURF je taktiež invariantný voči vzdialenosti kamery od objektu a voči rotáciam. Existuje však varianta U-SURF, ktorá nedetekuje rotáciu jednotlivých kľúčových bodov, a teda je ešte rýchlejšia a vhodná v prípadoch, kedy rotácia kamery nie je ovplyvňujúci faktor [13].

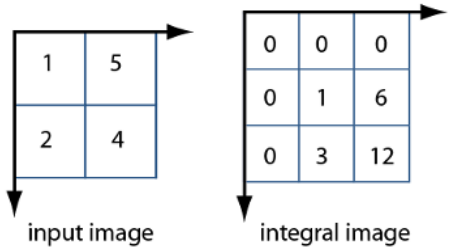
Jedným z urýchľujúcich faktorov je použitie integrálnych obrazov, ktorých efektivita bola potvrdená v detektore Viola-Jones. Integrálny obraz (obr. 2.21), je obraz, kde sa každá hodnota bodu rovná súčtu hodnôt, ktoré sú v pôvodnom obraze naľavo a hore od pôvodného bodu, podľa rovnice 2.4.

$$I(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x',y'), \tag{2.4}$$

kde $I(x,y)$ je bod na pozícií x,y v integrálnom obraze, $i(x,y)$ je bod na pozícií x,y v pôvodnom obraze.

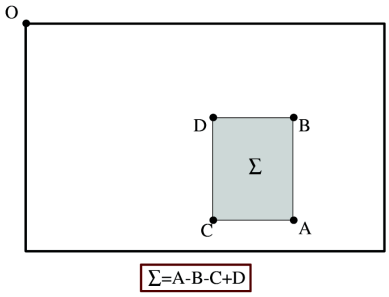
Výhodou tohoto obrazu je fakt, že súčet hodnôt intenzít v ľubovoľnej obdĺžnikovej časti obrazu sa dá vypočítať pomocou troch operácií sčítania. Celkovo to teda sú 3 operácie sčítania a 4 krát načítanie hodnôt z pamäti, čo je veľmi malý počet operácií v porovnaní s klasickým súčtom hodnôt. Rovnicou pre tento súčet je rovnica 2.5.

$$\sum_{\substack{x_0 < x \leq x_1 \\ y_0 < y \leq y_1}} i(x',y')i(x,y) = I(D) + I(A) - I(B) - I(C) \tag{2.5}$$



Obr. 2.21: Reprezentácia klasického obrazu (vľavo) a integrálneho obrazu (vpravo). Prevzaté z [25].

Pričom $i(x, y)$ je bod na pozícií x, y v pôvodnom obraze a body A,B,C,D majú súradnice $A = (x_0, y_0), B = (x_1, y_0), C = (x_0, y_1), D = (x_1, y_1)$. Na znázornenie slúži obrázok 2.22.



Obr. 2.22: Znázornenie súčtu oblasti pomocou integrálnych obrazov. Na dosiahnutie výsledku sú potrebné tri operácie súčtu a štyri prístupy do pamäte. Prevzaté z [13].

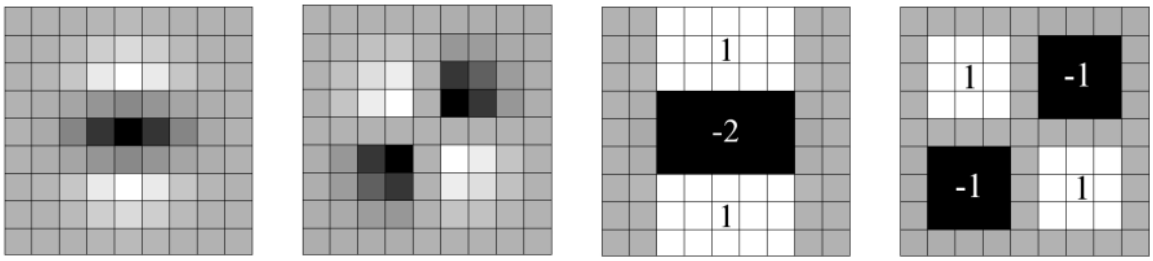
2.6.1 Detekcia bodov záujmu

Na detekciu bodov záujmu sa využíva Hessova matica, kvôli priaznivej presnosti na nej založeného detektoru. Má tvar:

$$\mathcal{H}(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}, \tag{2.6}$$

kde x je bod v obraze, $L_{xx}(x, \sigma)$ je konvolúcia druhej derivácie Gaussovej funckie v bode x a σ značí mierku.

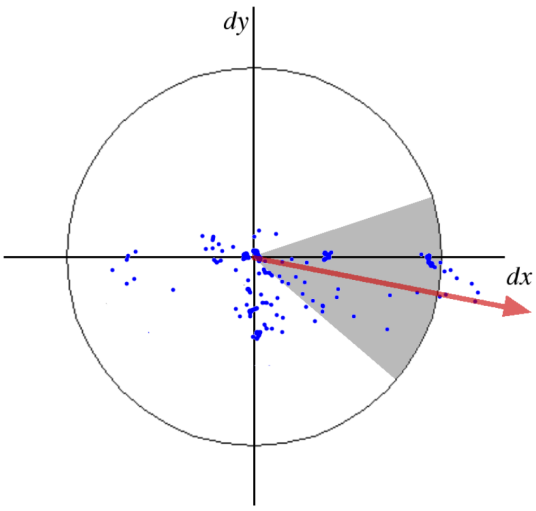
Detekcia však musí prebiehať vo viacerých mierkach. Tradične sa na to využíval scale-space s postupnými konvolúciami obrazu s Gaussovým filterom, no v algoritme SURF sa používajú postupné konvolúcie obrazu s aproximovanými druhými deriváciami Gaussovho filtra (obr. 2.23). Ďalším rozdielom je, že zvyčajne býva scale-space generovaný postupným zmenšovaním obrazu a konvolúciou s filtrom konštantnej veľkosti. V algoritme SURF sa však postupne zväčšuje filter. Obe tieto zmeny prinášajú zníženie celkovej výpočetnej zložitosti, a teda urýchlenie algoritmu. Bod je označený ako bod záujmu, pokiaľ je determinant jeho Hessovej matice maximom v okolí 3x3x3. Najmenším filtrom je filter 9x9, pri $\sigma = 1.2$, zväčšujúci sa postupne na veľkosti 15x15, 21x21 atď.



Obr. 2.23: Z ľava do prava: Parciálna derivácia Gaussiánu v y-smere (L_{yy}) a v xy-smere (L_{xy}) a ich respektívne použité aproximácie. Šedé oblasti sa rovnajú nule. Prevzaté z [13].

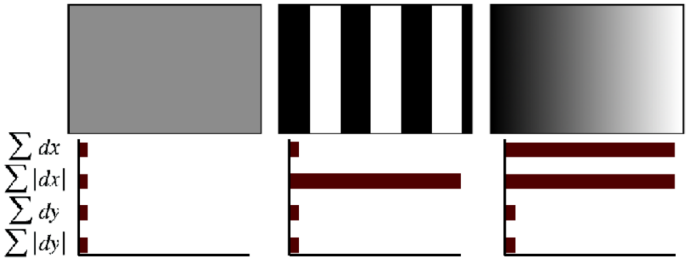
2.6.2 Vytvorenie deskriptoru

Na dosiahnutie odolnosti algoritmu voči rotácii, musí každý bod záujmu obsahovať informácie o svojej orientácii. Pre detekciu natočenia sa najprv vypočítajú odozvy na Haarove vlny v x a y-smere v okolí $6s$ okolo bodu záujmu, kde s je mierka, pri ktorej sa daný bod našiel. Závislá od mierky je aj veľkosť Haarovych vĺn, ktoré majú dĺžku strany $4s$. Opäť sa využívajú integrálne obrazy pre rýchle filtrovanie. Následne sú odozvy na Haarove vlny vážené Gaussianom so $\sigma = 2s$ a sú reprezentované ako body v priestore, kde hodnota odozvy na vlnu v x-smere určuje hodnotu na x-osi a hodnota odozvy na vlnu v y-smere určuje hodnotu bodu na y-osi. Orientácia sa určuje pomocou sliding window o veľkosti $\frac{\pi}{3}$ (obr. 2.24), v ktorom sa vykonajú súčty jednotlivých odoziev. Každý takýto súčet vytvára lokálny vektor orientácie a najväčší z týchto vektorov určuje celkovú orientáciu [13].



Obr. 2.24: Priradenie orientácie: Detekcia dominantnej orientácie pomocou sliding window o veľkosti $\frac{\pi}{3}$ na odozvách z Haarových vĺn, vážených Gaussianom v kruhovom okolí bodu záujmu. Prevzaté z [13].

Po získaní orietnácie sa vytvorí štvorcová oblasť s dĺžkou strany $20s$ a odpovedajúcou orientáciou. Táto oblasť sa následne rozdelí na 4x4 štvorcové podoblasti. Z týchto podoblastí je vždy vybraných 5x5 pravidelne rozmiestnených bodov, na ktoré znovu aplikujeme Haarove vlny. Odozvy v horizontálom smere nazveme d_x a vo vertikálnom d_y. Tieto odozvy sú vážené Gaussianom, kde $\sigma = 3.3s$. Popisovací vektor pre jednu podoblasť je teda $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Príklad hodnôt tohoto vektoru sa nachádza na obrázku 2.25. Výsledný popisovací vektor vznikne konkatenáciou popisovacích vektorov z týchto štyroch 4x4 podoblastí s výslednou dĺžkou 64.



Obr. 2.25: Príklad hodnôt popisného vektora pre rôzne obrazy. Pre homogénny obraz sú všetky hodnoty relatívne nízke. Pri periodických hodnotách na osi x sa zvýši hodnota $\sum |d_x|$, ale všetky ostatné ostanú nízke. Ak sa hodnota na osi x postupne mení, tak sa okrem $\sum |d_x|$ zvýši aj hodnota $\sum d_x$. Prevzaté z [13].

Kapitola 3

Využité knižnice

Cieľom tejto kapitoly je oboznámiť čitateľa s vybranými knižnicami. Pri každej knižnici sú uvedené dôvody, prečo bola efektívnym výberom pri riešení tejto témy. Všeobecne boli preferované open-source knižnice, ktoré sú voľne dostupné aj na viacerých platformách. Ďalším kritériom bola ich rozšírenosť. Vysoká rozšírenosť umožňuje rýchlejšie riešiť vzniknuté problémy pri implementácií vlastného algoritmu tým, že existuje viac používateľov s podobným problémom a zároveň viac odborníkov, ktorý poznajú riešenie na daný problém. Takáto rozšírenosť taktiež svedčí aj o kvalite danej knižnice. Ďalšie kritéria pre výber boli kompletnosť dokumentácie či výskyt cvičných tutoriálov. Preferovaným jazykom boli C++ alebo Python.

3.1 Dlib

Knižnica Dlib vznikla v roku 2002 a je rozširovaná doteraz. Hlavným vývojárom je Davis King. Je licencovaná pod licenciou Boost, jedná sa teda o open-source softvér. Jej dizajn je založený na softwarovom inžinerstve založenom na komponentoch. Dala by sa teda klasifikovať ako zbierka vzájomne nezávislých, rozsiahle dokumentovaných softwarových komponentov. Je dostupná v jazykoch C++ a Python na systémoch Mac OS X, MS Windows, Linux, Solaris a mnoho ďalších [15].

Obsahuje nástroje na riešenie rozličných problematík, ako napríklad lineárna algebra, spracovanie obrazu, parsovanie XML a textu, či riešenie optimalizačných problémov. Zároveň obsahuje nástroje na vytváranie neurónových sietí, SVM, HOG, vďaka čomu je vhodná na riešenie tejto práce. Jej súčasťou je aj anotačný nástroj imglab, pomocou ktorého boli anotované snímky v databáze zbraní. Na oficiálnej stránke[1] sa nachádzajú dôkladne dokumentované tutoriály, ktoré umožňujú sa v danej problematike rýchlo zorientovať [15].

3.2 OpenCV

OpenCV je skratkou pre Open Source Computer Vision Library, je to teda knižnica s voľne šíriteľným kódom, určená na spracovanie počítačového vidienia. Je šíriteľná za podmienok GPL a BSD licencií a je dostupná na viacerých platformách (MS Windows, Linux, Android, Mac OS). Pôvodne je napísaná v jazyku C++, má však rozhrania aj pre jazyky Java a Python.

[1] www.dlib.net

Rieši problematiku počítačového videnia a momentálne má viac než 2500 algoritmov, pričom mnohé z nich implementujú najmodernejšie prístupy. Tieto algoritmy sa následne používajú napríklad v detekcii a klasifikácii objektov, sledovaní pohybu objektu, vyhľadávaní podobných objektov z databázy, rozpoznávaní scény, a majú mnoho ďalších použití. Plno využíva schopnosti moderných viacjadrových procesorov a grafických kariet (CUDA). Zároveň je jednou z najpoužívanejších knižníc počítačového videnia - komunitu tvorí viac ako 47000 ľudí a počet stiahnutí presahuje 14 miliónov. Využíva ju široká verejnosť, ale aj najväčšie a najznámejšie IT firmy ako Google, Microsoft, Intel, alebo IBM.

Do tejto práce bola knižnica OpenCV vybraná kvôli jednoduchosti implementovania aloritmov ako napríklad SURF.

3.3 Tensorflow

Tensorflow je momentálne jedna z najpoužívanejších open-source knižníc pre riešenie problematiky strojového učenia. Je vyvíjaný od roku 2011, kedy bol známy pod menom DistBelief. Nie je viazaný na architektúry a tak dokáže pracovať na CPU, GPU ako aj na mobilných zariadeniach. Pracuje na bázi grafov, kde každý uzol je matematickou operáciou a hrany sú viacdimenzionálne úložné priestory - tensory. Ponúka dostatočnú úroveň abstrakcie na jednoduchú implementáciu algoritmov strojového učenia [38].

Na účely detekcie objektov bol vyvinutý Tensorflow Object Detection API[2] a Tensorflow Model Zoo[3] ponúka predtrénované modely spolu s konfiguráciami pre prenosové učenie (Transfer learning).

[2]Tensorflow Object Detection API sa nachádza na stránke https://github.com/tensorflow/models/tree/master/research/object_detection

[3]Tensorflow Model Zoo nájdete tu: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Kapitola 4

Experimentálna časť

Cieľom tejto kapitoly je popísať použitý spôsob riešenia. Na začiatku sú špecifikácie jednotlivých výpočetných zariadení, na ktorých prebiehali experimenty, a následne je predstavená vytvorená databáza. Ďalej je popísaný spôsob hodnotenia úspešnosti použitých detekčných algoritmov. Nasledujú jednotlivé experimenty, u ktorých bude popísaný ich spôsob implementácie, použitý dataset a výsledky na danom datasete. Prvá časť experimentov sa zaoberá testovaním daných detekčných algoritmov na obrázkoch z databáze IMFDB. Je porovnaná úspešnosť jednotlivých modelov, ich silné a slabé stránky. Pri priaznivých výsledkoch sú modely použité v druhej časti. Druhá časť sa zaoberá testovaním detekčných algoritmov na videu. Na zistenie robustnosti voči šumu boli porovnané výsledky detekcie pre neupravené a zašumované video. Na zlepšenie efektivity detekcie bol následne zvolený filter na odstránenie šumu. Boli porovnané výsledky pre zašumené a odšumené video.

4.1 Použité zostavy

Keďže jedným z faktorov efektivity algoritmov je rýchlosť, je nutné, aby sa uviedla informácia ohľadom počítačov, na ktorých boli experimenty vykonané. Celkovo sa použili 2 počítače a to: Notebook Lenovo Y500 a výskumný počítač skupiny STRaDe. Informácie o nich sú uvedené nižšie a boli získané pomocou programu Hardinfo.

Lenovo Y500:

- Operačný systém: Ubuntu 18.04.2
- Procesor: Intel® Core™ i7-3630QM @ 2.40 GHz
- Operačná pamäť: 8GB DDR3
- Grafická karta: Nvidia GeForce GT 650M 2GB
- Pevný disk: Western Digital Black 750GB, 7200 ot/min

Výskumný počítač skupiny STRaDe:

- Operačný systém: Ubuntu 18.04.2
- Procesor: Intel® Core™ i9-9900K @ 3.60GHz
- Operačná pamäť: Kingston HyperX Predator 32GB (2x16GB) DDR4

- Grafická karta: 2x ASUS TURBO GeForce GTX 1080 8GB
- Pevný disk: ADATA XPG GAMMIX S11, M.2 - 960GB

4.2 Databáza obrazov

Na účely trénovania a testovania jednotlivých algoritmov bola vytvorená anotovaná databáza, ktorej obrazy boli prevzaté z Internet Movie Firearms Database - IMFDB [1], konkrétne z kategórií Behind the scenes a Screenshots. IMFDB bola vybraná z niekoľkých dôvodov. Prvým dôvodom je jej rozsiahlosť. Vybrané kategórie obsahujú viac ako 155 000 obrázkov zbraní. Ďalším dôvodom je variácia v držaní zbran. IMFDB samozrejme obsahuje veľa fotiek akčných hrdinov v tradičných filmových pózach, ale zároveň obsahuje snímky, na ktorých sú zbrane nosené, alebo držané netradičným spôsobom. Väčšia variácia snímok zvyčajne umožňuje lepšiu detekciu. Variabilná je aj veľkosť snímok, kde najmenšie snímky sú veľké okolo 425x245 pixelov a najväčšie majú typicky veľkosť okolo 1920x1080.

Vytvorená databáza pôvodne obsahovala 495 dlhých a krátkych zbraní v 472 obrázkoch, no pre účely SSD bola rozšírená na 1406 zbraní v 1265 obrázkoch. Rozdelenie na trénovacie a testovacie datasety je popísané v kapitolách príslušných experimentov. Zbrane v databázi sú zvyčajne držané ľuďmi, keďže ich chceme detekovať v akcii, t.j. keď sú (neoprávnene) držané v ruke.

4.3 Spôsob hodnotenia jednotlivých modelov

Pre hodnotenie jednotlivých modelov použijeme metódu pomocou **Chybovej matice** (Confusion matrix). Jednotlivé stĺpce tejto matice (obr. 4.1) predstavujú klasifikované triedy. Každý riadok predstavuje skutočnú triedu daného objektu. Hodnota TP - True Positive označuje počet objektov klasifikovaných ako správne, ktoré sú aj v skutočnosti správne. Hodnota FP teda označuje počet objektov klasifikovaných ako správne, no v skutočnosti sú nesprávne. FN určuje počet objektov klasifikovaných ako nesprávne, ktoré sú v skutočnosti správne, a TN označuje počet správne klasifikovaných nesprávnych objektov [26, 7].

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	TP	FN
	Nesprávne	FP	TN

Tabuľka 4.1: Chybová matica

Na kvantifikovanie kvality detektora sa používajú tri hodnoty - presnosť (accuracy), precíznosť (precision) a senzitivita (recall). Presnosť určuje, koľko krát sa z celkových hodnotení podarilo objekt správne klasifikovať. Môžeme ju vyjadriť vzťahom 4.1.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precíznosť určuje úspešnosť klasifikátora pri určovaní správnych tried. Je vyjadrená v rovnici 4.2.

$$Preciznost = \frac{TP}{TP + FP} \tag{4.2}$$

[1] http://www.imfdb.org/wiki/Category:Images

Senzitivita vyjadruje pomer nájdených a nenájdených objektov. Je určená podľa rovnice 4.3.

$$Senzitivita = \frac{TP}{TP + FN} \tag{4.3}$$

4.4 Detekcia pomocou HOG a SVM

4.4.1 Prvý experiment

Platformou na tento experiment bol notebook Lenovo Y500. Úlohou prvého experimentu bolo zistiť, či je algoritmus HOG+SVM na úlohu detekcie zbraní vôbec použiteľný. Na toto zistenie sme sa rozhodli detekovať zbraň - pištoľ - v jednej orientácii z toho dôvodu, že detekcia daného objektu je závislá na orientácii. Pomocou HOG+SVM budeme detekovať iba pištole kvôli tomu, že telá (rámy) jednotlivých pištolí sú podobné, a tak bude na detekciu všetkých pištolí stačiť jeden model. Napríklad pri útočných puškách sú rozdiely v tvare tela pomerne veľké (ak napríklad porovnáme AK-47 a FAMAS), a bolo by pravdepodobne potrebné mať viac detekčných modelov. Výsledky z detekcie pištolí preto budeme brať ako najrýchlejší a najlepší prípad detekcie pri použití tohoto algoritmu. Rotácia použitá v tomto experimente je na obrázku 4.1.

Obr. 4.1: Rotácia pištole v prvom experimente (Colt 1911)

Dataset

Výhodou algorimu HOG + SVM má byť použitie pomerne malých trénovacích datasetov. Preto v tomto experimente bolo vybraných 39 fotiek z prvej verzie databáze ako trénovacie a 80 fotiek ako testovacie. Testovacie snímky však nemajú vždy na obrázku pištoľ a pištoľ navyše nemusí byť v detekovateľnej rotácii. V tejto konkrétnej rotácii sa v testovacom datasete nachádza len 8 obrázkov. 8 obrázkov sa môže zdať ako malé číslo, no netreba však zabudnúť, že ostatných 72 obrázkov sa musí klasifikovať ako obrázok bez zbrane. Použité obrázky majú rôzne rozlíšenie. Zbrane boli anotované pomocou nástroja imglab, ktorý je dodávaný s knižnicou dlib.

Medzi trénovacie pištole patria najznámejšie filmové zbrane, ako napríklad Colt 1911, Browning Hi-Power, Glock 19, Walther PP, Beretta 92FS, SIG P226 a iné.

Implementácia

Algoritmus bol implementovaný v jazyku C++ s použitím knižnice dlib. V tejto knižnici je implementovaný vylepšený - Felzenszwalbov [9] algoritmus HOG. Najskôr sa vytvorí

HOG deskriptor z trénovacích obrázkov spôsobom spomínaným v kapitole 2.5. Veľkosť tohoto deskriptoru je rovnaká ako veľkosť posuvného okna, ktorá je zároveň parametrom tohto detektoru. Tieto deskriptory slúžia ako pozitívne príklady pre SVM. Všetko, čo v testovaných dátach nie je označené ako hľadaný objekt sa automaticky berie ako negatívny príklad pre učenie SVM. Pri detekcii sa používa HOG pyramída, ktorá obsahuje postupne zmenšovaný obraz - pomer zmenšovania je ďalší parameter, pričom pri každom zmenšení prebieha vyhladzovanie hrán daného obrazu. Po každej úrovni postupne prechádza sliding window a pri nájdení zbrane v niektorej úrovni pyramídy je nájdený objekt upscalovaný a zobrazený. Tento pomer je vždy menší ako jedna, a čím je väčší, tým je obrazová pyramída hustejšia. Ďalšími parametrami pre SVM sú c a ε, kde c-parameter určuje pomer medzi najväčším priestorom medzi triedami SVM a počtom správne klasifikovaných objektov - čím vyššie c, tak tým viac sa snaží SVM optimalizovať pre správne klasifikovanie objektov. ε je parameter, ktorý určí minimálnu potrebnú chybovosť pre ukončenie trénovania.

Priebeh experimentu

Cieľom experimentu bolo nájsť vhodné parametre na to, aby sa dosiahlo čo najväčšej presnosti algoritmu. Ako prvé sme vyskúšali parametre zo skúšobného programu na detekciu tvárí, dodávaného s dlibom[2], teda s pomerom zmenšovania 5/6 (v kóde sa nachádza parameter 6, ale tento parameter určuje menovateľ tohoto zlomku a dlib určí čitateľ ako $menovatel - 1$), c=1, ε=0.01, sliding window = 80x80. SVM bol natrénovaný za 48 sekúnd, ale nebola nájdená ani jedna zbraň. Následne sme skúsili zväčšiť veľkosť kĺzavého okna na 100x100, ale tým sa iba zvýšil čas trénovania na 1 m 25 s. Toto sa opakovalo aj s veľkosťou 120x120. Nefungovala ani zmena na 60x60, tak sme sa rozhodli vrátiť veľkosť sliding window na 80x80 a zmeniť zmenšovací pomer.

Skúsili sme nastaviť parameter zmenšovania na dvojnásobok, čiže 12. Pomer sa teda rovnal 11/12. Toto však neprispelo ku detekcii a trénovací čas sa zvýšil na 1 m 22 s. Ďalší pokus bol s pomerom 39/40, ktorý taktiež nedospel k žiadnym výsledkom.

Vrátili sme sa znova k pôvodným hodnotám a tentokrát sme zvyšovali c-parameter. Pri $c = 15$ sa detekovali 4 z 8 obrázkov a vyskytla sa jedna nesprávna detekcia (False Positive - obr. 4.2). SVM bolo natrénované za jednu minútu a osem sekúnd. Rozhodli sme sa ďalej zvyšovať c. Pri $c = 40$ a $c = 1000$ sme dostali rovnaké výsledky, preto sme sa rozhodli, že hodnota c ostane 15.

V tomto bode sme pre $c = 15$ zvyšovali hodnotu zmenšovacieho pomeru. Pri 11/12 sa správne detekovalo 5 zbraní a hodnota nesprávnych detekcií klesla na nulu. Pre ešte vyššie hodnoty tohoto pomeru sme nedosiahli lepších výsledkov (napr. pri 23/24 sa detekovali správne iba tri zbrane). Nepomáhalo ani zväčšovanie a zmenšovanie kĺzavého okna, najlepší výsledok sme dostali stále pri veľkosti 80x80.

Výsledné parametre pre tento algoritmus sú teda: $\varepsilon = 0.01$, $c = 15$, pomer zmenšovania 11/12 a veľkosť sliding window je 80x80. Tento detektor bol na počítači Lenovo Y500 natrénovaný za dve minúty a tridsaťsedem sekúnd. Chybová matica vyjadrená tabuľkou 4.2.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5 + 72}{5 + 72 + 0 + 3} = \frac{77}{80} = 96.25\% \tag{4.4}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{5}{5 + 0} = \frac{5}{5} = 100\% \tag{4.5}$$

[2] http://dlib.net/fhog_object_detector_ex.cpp.html

Obr. 4.2: Nesprávna detekcia - False Positive

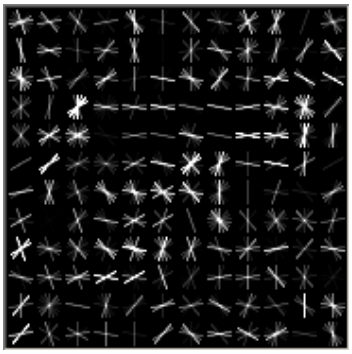
		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 5$	$FN = 3$
	Nesprávne	$FP = 0$	$TN = 72$

Tabuľka 4.2: Chybová matica

$$Senzitivita = \frac{TP}{TP + FN} = \frac{5}{5+3} = \frac{5}{8} = 62.5\% \tag{4.6}$$

Vyhodnotenie experimentu

Vyhodnotenie algoritmu sa nachádza v rovniciach 4.4, 4.5 a 4.6. Detektor demonštroval presnosť 96.25% a precíznosť 100%, čo sú veľmi dobré výsledky. Nižšia senzitivita 62.5% mohla byť spôsobená nesprávnym považovaním niektorých zbraní za detekovateľné. Algoritmus takisto demonštroval pomerne dobrú odolnosť voči skoseniu. Čas potrebný na vyhodnotenie jedného snímku je takmer zanedbateľný a priaznivý je aj čas tréningu - 2 m 37 s v najefektívnejšej verzií. Tieto vlastnosti robia tento algoritmus vhodným kandidátom pre ďalšie rozšírenie.



Obr. 4.3: Výsledný HOG deskriptor a ukážka robustnosti algoritmu.

Obrázok 4.3 ukazuje výsledný HOG deskriptor, ktorý vizualizuje histogramy gradientov. Pre účely zobrazenia sú gradienty otočené o 90 stupňov - takýmto spôsobom vzniknú na obrázku viditeľné hrany predmetu - pištole držanej zľava rukou.

Aj keď bol SVM trénovaný na obrázkoch zbrane Walther PPK, tak detekcia nie je zaručená. Detekciu môže do istej miery ovplyvniť spôsob držanie pištole a pozadie, príkladom je obrázok 4.4.

Obr. 4.4: Výsledky pre Walther PPK.

4.4.2 Druhý experiment

Cieľom druhého experimentu bolo zistiť efektivitu HOG + SVM detektoru pri manuálnom pridaní rotácií, keďže algoritmus nie je prirodzene invariantný voči rotácií. Ku detektoru z prvého experimentu bola pridaná zbraň držaná sprava a mierená rovno, zbraň držaná z zľava a mierená nadol a zbraň držaná sprava a mierená nahor (obr. 4.5).

Obr. 4.5: Orientácie zbraní pridaných detektorov, zľava: orientácia *right*, *right down*, *left up*.

Dataset

Použili sme rovnaký testovací dataset ako pri prvom experimente. Keďže však boli pridané rotácie, tak by tento krát malo byť detekovateľných 28 zbraní z celkových 80. Na tréning jednotlivých detektorov bolo použitých v priemere 30 fotiek na jeden detektor.

Implementácia

Prvým krokom bolo vytvorenie trénovacieho kódu pre každú orientáciu. V každom takomto súbore sme orientáciu zakomponovali do názvu. Z prvého experimentu máme súbor

fhog_gun_detector_left.cpp, zbraň držaná sprava má v názve namiesto left, right. Podobne to je aj so zbraňami mierenými do prava dole a ľava hore, tieto orientácie sú označené ako rd (right down) a lu (left up). Trénovanie prebieha rovnako, ako v prvom experimente, je treba však zistiť ideálne parametre každého detektoru.

Po trénovaní sa jednotlivé SVM uložia na disk. Tie sú načítané programom *fhog_detect_guns.cpp*, ktorý pomocou nich vyhodnotí každý testovací obraz.

Priebeh experimentu

Ďalším krokom bolo nájdenie správnych parametrov pre detekciu jednotlivých rotácií. Toto hľadanie sa vykonávalo empiricky, ako v prvom experimente, preto nebude vypísaný celý priebeh tohoto hľadania, ale iba výsledky.

Pre rotáciu *right* bola deteckia najúspešnejšia s parametrami $c = 15$ a pomerom zmenšovania 23/24, u rotácie *lu* $c = 40$ a pomerom 23/24, pri rotácií *rd* sme použili $c = 15$ a pomer 9/10.

Vyhodnotenie

Správne sa vyhodnotilo 9 zbraní, pričom sa raz vyskytla dvojitá detekcia. To však nepočítame ako chybu, keďže cieľom v tejto práci je zbrane detekovať a nie ich klasifikovať podľa rotácie. Takisto bol nájdený jeden revolver, čo budeme rátať ako chybu, keďže celý dataset pre trénovanie detektorov neobsahoval ani jeden revolver a navyše nebol algoritmom vyznačený celý, ale iba sčasti. Chybová matica je vyjadrená tabuľkou 4.3.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 9$	$FN = 17$
	Nesprávne	$FP = 1$	$TN = 53$

Tabuľka 4.3: Chybová matica pre experiment č.2

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{9 + 53}{9 + 53 + 1 + 17} = \frac{62}{80} = 77.5\% \tag{4.7}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{9}{9 + 1} = \frac{9}{10} = 90\% \tag{4.8}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{9}{9 + 17} = \frac{9}{26} = 34.61\% \tag{4.9}$$

Vyhodnotenie detektoru sa nachádza v rovniciach 4.7, 4.8 a 4.9. Najpríosnejším rozšírením prvého experimentu bol detektor *right*. Naopak, rotácie *left up* a *right down* boli nájdené v testovacej sade iba raz a zapríčinili nízku senzitivitu. Neefektívnosť vyhľadávania zbraní v týchto rotáciách mohla byť zapríčinená tým, že sa uhol natočenia trénovacích zbraní mohol líšiť, čo mohlo spôsobiť nepresné natrénovanie SVM. Keďže tento prístup pre hľadanie rotácií nebol efektívny, ostáva nám sa použiť iný prístup.

4.4.3 Tretí experiment

Keďže sa predošlý pokus o rozšírenie detekcie o rotácie manuálnym spôsobom skončil neúspechom, využijeme na to v tomto experimente funkcie knižnice dlib. Využívajú sa datasety pre *left* a *right* z predošlých experimentov.

Dataset

Používa sa trénovací dataset pre detektory *left* a *right* z predošlých experimentov. Mali by byť detekovateľné pištole zboku v ľubovoľnej rotácií, čo dohromady tvorí 28 zbraní.

Implementácia

V tomto riešení sa využíva funkcia *rotate_image_dataset()* z knižnice dlib. Táto funkcia umožňuje rotovať celý dataset. Rotujeme teda trénovací dataset pre detektory *left* a *right* o 15 stupňov. Funkcia používa radiány, preto sme využili konštantu 0.2617993878 rad, ktorá sa približne rovná 15 stupňom. Celkovo sa používa $360/15 = 24$ detektorov pre *left* a 24 detektorov pre *right* a ich rotácie.

O tréning sa starajú aplikácie *fhog_gun_left_rotator*, *fhog_gun_right_rotator* a *fhog_train_rotations*. Prvé dve obsahujú trénovací kód a pararametrami sú okrem lokácie databázy aj natočenie a názov SVM pre uloženie na disku. Následne ich *fhog_train_rotations* spúšťa s parametrami pre postupné rotácie. Na vyhodnotenie týchto 48 detektorov používame aplikáciu *fhog_find_rotations*.

Vyhodnotenie

Správne bolo detekovaných 11 zbraní. Tentokrát boli okrem revolveru aj ďalšie dve nesprávne detekcie (False Positive). Navyše sa pri použití 48 detektorov čas na detekciu zvýšil až na 15 sekúnd, závisiac od rozlíšenia fotky. Čas pre natrénovanie detektorov bol 4 hodiny a 53 minút, jeden detektor bol teda natrénovaný za približne 6 minút.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 11$	$FN = 17$
	Nesprávne	$FP = 3$	$TN = 50$

Tabuľka 4.4: Chybová matica pre experiment č.3

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{11 + 50}{11 + 50 + 3 + 17} = \frac{61}{81} = 75.3\% \tag{4.10}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{11}{11 + 3} = \frac{11}{14} = 78.57\% \tag{4.11}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{11}{11 + 17} = \frac{11}{28} = 39.28\% \tag{4.12}$$

Vyhodnotenie detektoru sa nachádza v rovniciach 4.10, 4.11, 4.12 a chybovej matici 4.4. V porovnaní s druhým experimentom sa počet nájdených zbraní zvýšil len o trochu a algoritmus klasifikoval o dva obrázky viac nesprávne, čo viedlo ku zníženiu presnosti a precíznosti. Vzhľadom k týmto prinajlepšom priemerným výsledkom padlo rozhodnutie vyskúšať iný algoritmus.

4.5 Detekcia pomocou algoritmu SURF

Podobne, ako pri prvom experimente HOG + SVM, bolo úlohou prvého experimentu zistiť, či je algoritmus SURF vôbec vhodný na problematiku detekcie zbraní. Celý experiment bol realizovaný na notebooku Lenovo Y500.

Dataset

Používa sa prvá verzia databázy. V algoritme SURF sa trénovacie snímky zbraní musia vyhľadať v testovacích snímkoch. Preto boli jednotlivé zbrane vystrihnuté z trénovacieho datasetu (celý snímok so zbraňou by sa nedokázal nájsť v inom snímku).

Implementácia

Pre implementáciu bola využitá knižnica OpenCV verzie 3.3.1, novšia verzia nebola použitá kvôli problémom s kompatibilitou. Kód pre vyhľadanie kľúčových bodov SURF bol prevzatý zo stránok OpenCV.[3]. Priebeh vyhľadávania kľúčových bodov je rovnaký ako v kapitole 2.6, s jediným rozdielom, že minimálny hessián musí byť určený programátorom. Rovnaké (podobné) kľúčové body sú nájdené za použitia brute force matcheru.[4] Porovnanie kľúčových bodov pomocou brute force matcheru sa zakladá na ich vzájomnej euklidovskej vzdialenosti. Pri nízkej vzdialenosti sa dvojice označia ako zhoda (match).

Ďalej zistíme minimálnu vzdialenosť zo všetkých zhôd a vyfiltrujeme tie zhody, ktoré ju majú väčšiu ako trojnásobok minimálnej vzdialenosti. Táto operácia zaručí, že jednotlivé zhody si sú naozaj podobné. Tieto body sa následne použijú na nájdenie homografie pomocou algoritmu RANSAC [24, 39, 32]. Pomocou nájdenej homografie nájdeme rohy objektu v scéne a vykreslíme ich.

Priebeh experimentu

Jediným parametrom pre tento algoritmus je minimálny hessián (determinant Hessovej matice), ktorý musí byť prekročený na to, aby sa bod bral ako kľúčový. Táto hodnota sa musí empiricky zistiť. Všimli sme si, že pri hodnotách hessiánu, ktoré sú menšie ako 1000, sú obrázky presýtené kľúčovými bodmi. Preto sme používali hessiány s hodnotami 1000, 1500, 2000 pre postupné pokusoch v snahe nájsť zhodu medzi obrázkom zbrane a scény so zbraňou.

Manuálne bolo vyskúšaných 10 scén zo zbraňami, ktoré sa v trénovacej sade nachádzali viackrát (Glock 17, AK-47 a podobné), no žiadna z 409 fotiek zbraní nebola v scénach nájdená (obr. 4.6). Tento experiment sme zopakovali pre všetky 3 hodnoty hessiánu, no aj tak nebola nájdená žiadna zbraň. Na vyskúšanie funkčnosti algoritmu sme použili aj databázu zbraní z práce [27], ktorá obsahuje zbrane na bielych pozadiach. Výsledok sa ale nezmenil.

Preto sme museli overiť funkčnosť algoritmu. Prvým testom bol, či algoritmus nájde danú zbraň v scéne z ktorej pochádza (obr. 4.7). Pri hessiáne 1000 bola zbraň bez problémov nájdená. Ďalším krokom bolo otestovať rotáciu. Na to sme vytvorili fotky knihy [32]. Na

[3] https://docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html

[4] https://docs.opencv.org/3.0-beta/doc/tutorials/features2d/feature_description/feature_description.html

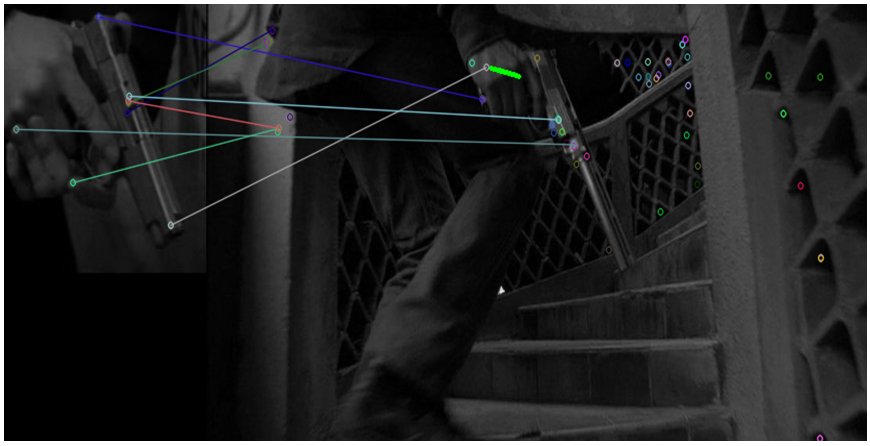
to sme odfotili knihu kolmo zhora a hľadáme ju vo fotke s otočenou knihou položenou na stole. Kniha bola nájdená ako pri hessiane 1000, tak aj 6000, ktorý je na fotke 4.8.

Vyhodnotenie

Z praktickej časti tohoto experimentu vyplývajú tieto fakty:

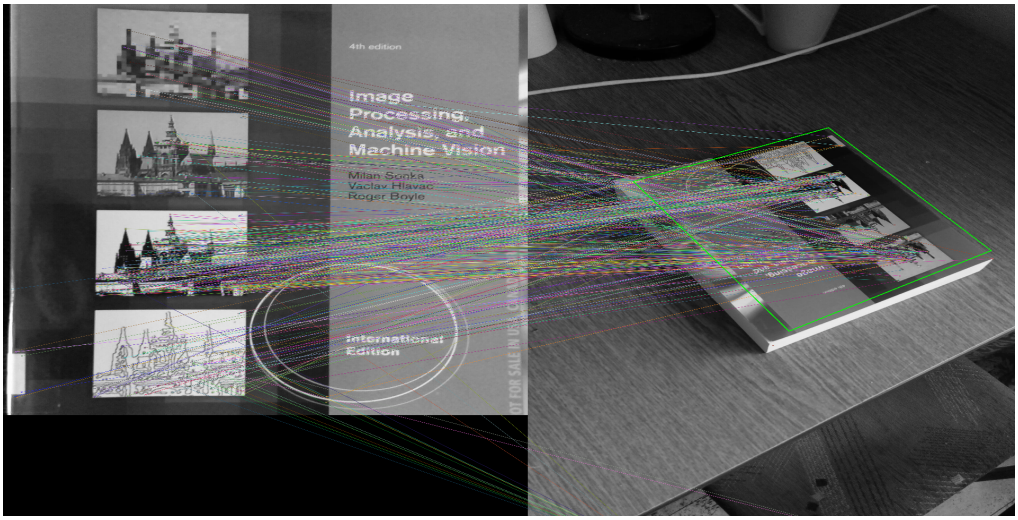
- Algoritmus SURF nie je vhodný pre vyhľadávanie podobných objektov v odlišných scénach.
- Zbrane v rozlíšení, aké máme v databáze vystrihnutých zbraní - v rozmedzí od 60x60 až do 500x500 - nemajú dostatok textúry na to, aby sa dokázali vytvoriť efektívne kľúčové body. SURF vyhodnotil ako kľúčové body zvyčajne prechody medzi zbraňou a pozadím.
- SURF je efektívny pre hľadanie viacej texturovaných objektov - predné strany kníh, fotografie, vzory.

Tieto vlastnosti robia bohužiaľ algoritmus SURF nevhodný pre detekciu zbraní v obraze.



Obr. 4.6: Detekcia SURF, na obrázku sú zobrazené spojené kľúčové body.

Obr. 4.7: SURF dokáže detekovať zbraň v pôvodnom obraze.



Obr. 4.8: Detekcia SURF na knihe, čiary označujú spojené kľúčové body, zelený štvoruholník je detekcia.

4.6 Detekcia pomocou SSD

Posledným skúšaným algoritmom je Single Shot Multibox Detection. Vybrali sme si jeho implementáciu v Tensorflowe pomocou Object Detection API. Bol použitý predtrénovaný model SSD[5] využívajúci Mobilenet verzie 1, ktorý bol natrénovaný na datasete Microsoft COCO [21]. Tento model bol použitý kvôli veľmi dobrej presnosti a rýchlosti klasifikácie. Na pretrénovanie sme použili konfiguráciu pre tréning tohoto modelu na Oxford-IIIT Pet Dataset[6] [28]. Natrénovali sme 3 modely, ktoré sa navzájom líšili buď použitým datasetom, alebo dĺžkou trénovacieho času.

Dataset bol tentokrát labelovaný nástrojom LabelImg. Ten pre každý súbor vytvoril XML súbor, a zapísal doň informácie o objektoch vo formáte PASCAL VOC. Tieto súbory boli následne prevedené do formátu CSV pomocou mierne upravených skriptov z práce[36]. Následne sme z tejto práce použili ešte skript na generovanie súborov TFRecord, ktoré vyžaduje Tensorflow, zo súborov CSV.

Zároveň sme využili pôvodne testovacie datasety z databáz pre validáciu jednotlivých modelov. Pri labelovaní jednotlivých zbraní sme využili dve triedy - *gun_short* a *gun_long* pre respektívne krátke a dlhé zbrane. Zle priradenú triedu pre nájdenú zbraň však nebudeme brať ako chybu, lebo úlohou tejto práce nie je klasifikácia strelných zbraní, ale iba ich detekcia. Pre testovanie sme museli vytvoriť ďalší dataset, ktorý sa skladá zo 100 obrázkov. Vyskytuje sa v ňom 55 zbraní na 50 obrázkoch a ďalších 50 obrázkov je bez zbrane.

V tejto sekcií bude pri každom modeli uvedená hodnota **stratovej funkcie** (loss function). Hodnota tejto funkcie dáva používateľovi najavo, ako dobre funguje jeho detektor. Pre SSD je kompletný vzorec tejto funkcie uvedený v práci [22], ale zjednodušene ho môžeme vyjadriť podľa rovnice 4.13,

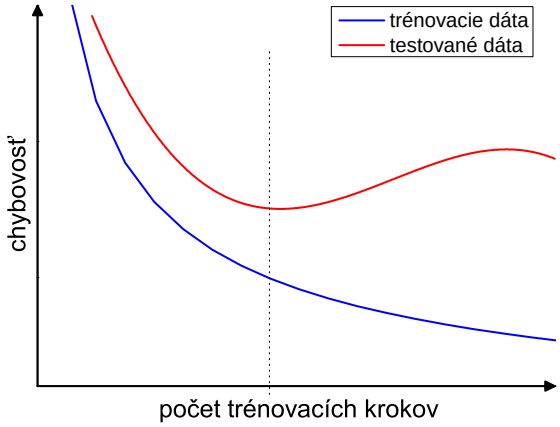
[5] http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz

[6] Rovnaká konfigurácia (až na počet tried) bola použitá pre tréning tohoto modelu na COCO datasete. Nachádza sa na: https://github.com/tensorflow/models/blob/master/research/object_detection/samples/configs/ssd_mobilenet_v1_pets.config

$$total\_loss = confidence\_loss + \alpha * location\_loss, \tag{4.13}$$

kde *total_loss* je výsledná hodnota stratovej funkcie, *confidence_loss* je hodnota vyjadrujúca istotu detektora, že sa v danej ohraničenej oblasti nachádza objekt. *location_loss* je hodnota vyjadrujúca vzdialenosť predikovanej oblasti detekcie a skutočnej oblasti. α určuje pomer, akým *location_loss* prispieva do celkovej straty [10].

Typicky platí pravidlo, že čím nižšia je hodnota stratovej funkcie, tým presnejší je detektor. To však platí iba pre validačné dáta. Pre veľmi nízke hodnoty (od 0.5 a nižšie) je detektor väčšinou veľmi nepresný pre detekciu na dátach, na ktorých nebol trénovaný ani validovaný. Je to tým, že jednotlivé parametre a váhy sa nastavia pre veľmi presnú detekciu na validačnom datasete a kvôli tomu nedokáže detekovať mierne odlišné objekty v testovacom datasete. Tento jav nazývame **pretrénovanie** (overfitting) [12] a je zobrazený na obrázku 4.9. Preto je v praxi lepšie zastaviť tréning skôr - okolo hodnoty 1 (napríklad v rozmedzí 1.2 až 0.8).



Obr. 4.9: Chyba pre trénovacie dáta klesá až do hodnoty 0, ale pre testovacie dáta sa od určitého bodu začne znova zvyšovať. Upravené z [12].

4.6.1 Prvý model

V tomto modeli sme na tréning použili prvú verziu databáze, kde bolo 402 zbraní použitých na tréning a 80 zbraní na následnú validáciu. Po zapnutí trénovania bola hodnota stratovej funkcie monitorovaná pomocou aplikácie Tensorboard.

Tréning prebiehal na výskumnom počítači skupiny STRaDe. Po 4 hodinách a 20 minútach sa hodnota stratovej funkcie nejako významne nemenila, tak bol tréning zastavený. Vykonalo sa 37 964 trénovacích krokov a výsledná hodnota stratovej funkcie bola $1,011$. Vývoj stratovej funkcie je zobrazený na grafe 4.10.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{43 + 41}{43 + 41 + 11 + 12} = \frac{84}{107} = 78.5\% \tag{4.14}$$

Obr. 4.10: Priebeh stratovej funkcie.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 43$	$FN = 12$
	Nesprávne	$FP = 11$	$TN = 41$

Tabuľka 4.5: Chybová matica pre prvý natrénovaný model SSD.

$$Preciznost = \frac{TP}{TP + FP} = \frac{43}{43 + 11} = \frac{43}{54} = 79.62\% \tag{4.15}$$

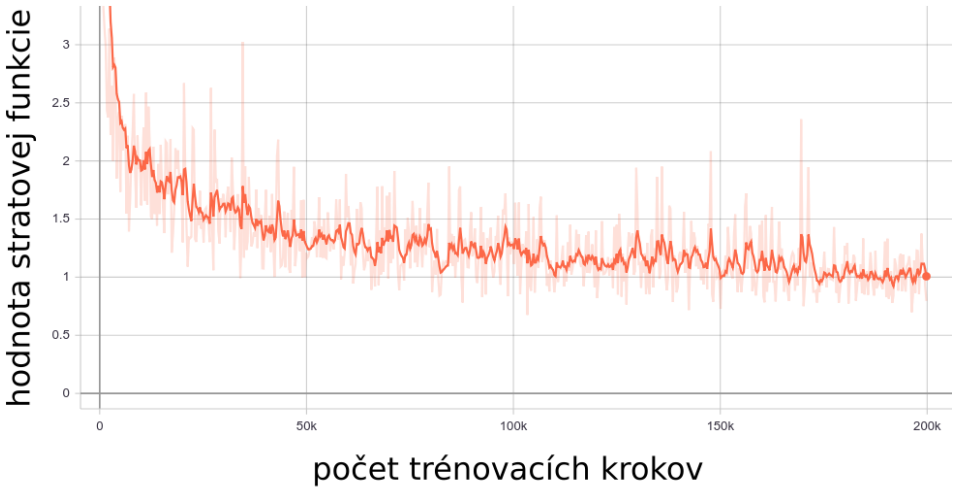
$$Senzitivita = \frac{TP}{TP + FN} = \frac{43}{43 + 12} = \frac{43}{55} = 78.18\% \tag{4.16}$$

Chybová matica modelu sa nachádza v tabuľke 4.5. Vyhodnotenie modelu sa nachádza v rovniciach 4.14 až 4.16. Model ukázal pomerne dobrú presnosť pri detekcií zbraní, čo v skutočností sú v obraze (True Positive). Nekonzistentne však vyhľadával napríklad brokovnice, alebo pištole v niektorých polohách a osvetleniach. Niektoré zbrane mali dvojité detekcie (ostreľovacie pušky) a iné zbrane mali príliš veľké detekčné okno. Zároveň však ako zbraň považoval aj niektoré iné predmety, ako autá, gitary, mobilné telefóny a iné. Ďalším krokom bude pokus o spresnenie pri vyhľadávaní zbraní a rozšírenie pre detekcie ďalších zbraní a zbraní v rôznych polohách.

4.6.2 Druhý model

Cieľom tohoto modelu bolo vylepšiť výsledky prechádzajúceho modelu. Z tohoto dôvodu, bol trénovací dataset rozšírený na 1232 zbraní a validačný set na 174 zbraní. Kvôli tomuto zvýšenému počtu sme nechali model vykonať 200-tisíc trénovacích krokov, čo je aj limit pre Pets dataset [28]. Trénovanie trvalo 22 hodín a 35 minút a skončilo na hodnote stratovej funkcie $0,7954$ (obr. 4.11). Chybovou maticou tohoto modelu je tabuľka 4.6.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{38 + 48}{38 + 48 + 2 + 17} = \frac{86}{105} = 81.9\% \tag{4.17}$$



Obr. 4.11: Priebeh stratovej funkcie, pre model č. 2. TensorBoard automaticky vyhladzuje grafy pre lepšiu čítateľnosť, skutočný priebeh je na grafe vyblednutý.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 38$	$FN = 17$
	Nesprávne	$FP = 2$	$TN = 48$

Tabuľka 4.6: Chybová matica pre experiment č.2

$$Preciznost = \frac{TP}{TP + FP} = \frac{38}{38 + 2} = \frac{38}{40} = 95\% \tag{4.18}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{38}{38 + 17} = \frac{38}{55} = 69.09\% \tag{4.19}$$

Oproti predchádzajúcemu modelu sa síce výrazne znížil počet nesprávnych detecií (FP), ale zároveň zreteľne klesol aj počet správnych detekcií, napriek takmer strojnásobeniu veľkosti databázy. Toto sa odrazilo vo zvýšení všetkých metrík, okrem senzitivity, ktorá klesla (rovnice 4.17 - 4.19). Tieto fakty nasvedčujú tomu, že model bol pretrénovaný, a aj keď ukazoval nízku stratu pri trénovaní na validačnom datasete, nie je taký efektívny na dátach, na ktorých netrénoval. Uvidíme ako model ovplyvní skoršie ukončenie trénovacieho procesu.

4.6.3 Tretí model

Predchádzajúci model mal problémy s pretrénovaním (overfitting), tak sa ich v tomto modeli snažíme vyriešiť. Tento model využíva rovnakú databázu a nastavenia ako druhý, ale narozdiel od neho bolo trénovanie zastanevné na kroku 112 156. Hodnota chybovej funkcie v tomto bode je $0,9644$. Priebeh stratovej funkcie je možné sledovať na grafe 4.12.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{41 + 47}{41 + 47 + 3 + 14} = \frac{88}{105} = 83.8\% \tag{4.20}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{41}{41 + 3} = \frac{41}{44} = 93.18\% \tag{4.21}$$

Obr. 4.12: Priebeh stratovej funkcie pre model č. 3.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 41$	$FN = 14$
	Nesprávne	$FP = 3$	$TN = 47$

Tabuľka 4.7: Chybová matica pre model č.3

$$Senzitivita = \frac{TP}{TP + FN} = \frac{41}{41 + 14} = \frac{41}{55} = 74.54\% \tag{4.22}$$

Z rovníc 4.20 až 4.22 a chybovej matice 4.7 vidíme, že až na precíznosť sa model oproti modelu č.2 čiastočne zlepšil. Môžeme predpokladať, že ideálny pomer Presnosti, Precíznosti a Senzitivity by sme našli v rozmedzí 60 až 80 tisíc trénovacích krokov, pretože aj tento model ešte javí známky pretrénovania. Napriek tomu model dosiahol veľmi dobrých výsledkov pre detekciu. Jeho najväčšia slabosť je nízka senzitivita. Čas pre detekciu je na zostave Lenovo Y500 od 2 do 10 sekúnd, závisiac od rozlíšenia, spolu aj s diskovými operáciami. Týmto krokom je model pripravený na detekciu vo videu.

4.7 Testy na videu

V tejto sekcií sa budeme zaoberať efektívnosťou algoritmu SSD na videu. Budú vykonané tri testy, a to na normálnom videu, zašumovanom videu a následne odšumovanom videu.

Na účel týchto testov sme vybrali video od Paula Harella - Steyr GB review[7]. Použité rozlíšenie je 1280x720 pixelov. Toto video bolo vybrané z nasledovných dôvodov:

- Obsahuje zbraň väčšinu času natočenú zboku, pričom pomerne veľka časť trénovacej databázy sa nachádza v tejto orientácií.
- Pištoľ Steyr GB je pomerne vzácna a nenachádza sa v trénovacích datasetoch. Preto pri použití tohoto videa uvidíme efektívnosť pri hľadaní zbrane, na ktorej neboli modely trénované.
- Skladá sa z viacerých scén:
  - Grafické intro a outro.

[7] https://www.youtube.com/watch?v=X4_u3ceQNMU

- Ukázanie funkčnosti zbrane veľmi blízko pri kamere.

- Predstavenie a rozobratie zbrane v strednej vzdialenosti od kamery.

- Streľba na terče, kde je buď zbraň ďalej od kamery (strieľa sa aj z Beretty 92FS, ktorá sa v trénovacej databáze nachádza veľmi často), alebo zbraň nie je v obraze, lebo sa kamera sústredí na terče.

Za účelom týchto testov bol pomocou knižnice OpenCV vytvorený program *frameGetter* v jazyku C++. Pomocou neho sa z daného videa každú sekundu extrahuje jedna snímka. Tieto snímky následne vyhodnotíme pomocou najlepšieho - tretieho modelu SSD. Model na bázi HOG a SVM nebude na videu testovaný z dôvodu, že detekcia je pomalá a model neponúka uloženie výsledkov detekcie.

4.7.1 Test na neupravenom videu

Cieľom tejto sekcie je za prvé, otestovať model SSD na prípade zo skutočného života a za druhé, poskytnúť referenčné dáta pre porovnávanie s ostatnými testami. Celkovo sa z videa extrahovalo 515 snímkov, z ktorých je 270 bez zbrane, alebo so zbraňou v nedetekovateľnej polohe (snímky so zbraňou položenou na stole rátame ako snímky bez zbrane, takmer celá trénovacia databáza totiž obsahuje držané zbrane) a 235 so zbraňou.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 150$	$FN = 95$
	Nesprávne	$FP = 0$	$TN = 270$

Tabuľka 4.8: Chybová matica pre test s nezmeneným videom.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{150 + 270}{150 + 270 + 0 + 95} = \frac{420}{515} = 81.55\% \tag{4.23}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{150}{150 + 0} = \frac{150}{150} = 100\% \tag{4.24}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{150}{150 + 95} = \frac{150}{245} = 61.22\% \tag{4.25}$$

Chybovou maticou pre tento test je tabuľka 4.8 a jeho vyhodnotenie sa nachádza v rovniciach 4.23 až 4.25. Model dokázal správne detekovať takmer všetky výskyty zbrane pri strednej a malej vzdialenosti od kamery, avšak akonáhle sa veľkosť zbrane priblížila približne 70x70 pixelov, tak ju model nebol schopný nájsť. O tomto fakte svedčí vypočítaná senzitivita. Keby sme zbrane tejto veľkosti brali ako nedetekovateľné, tak by sa hodnota FN rovnala 11, čo by malo za následok stúpnutie senzitivity na hodnotu 93,16%. Rozmer 70x70 je pre ľudské oko ešte rozoznateľný, ale pre počítačové videnie to už nemusí stačiť. Model sa však snažíme hodnotiť čo najkritickejšie, a tak budeme uvádzať nižšiu senzitivitu. Na zostave Lenovo Y500 bol čas na detekciu na jednej snímke 4 sekundy, na výskumnom počítači bol tento čas 0.696 sekundy.

Ďalšou dobrou vlastnosťou modelu je jeho precíznosť. Model nepredikoval ani jednu zbraň na mieste, kde skutočne nebola.

4.7.2 Test na zašumovanom videu

Nahrané video častokrát nebýva perfektné a vyskytujú sa v ňom rôzne anomálie, akou je napríklad šum (kapitola 2.2). Použité testovacie video malo nízky podiel šumu, tak sme do neho pridali gaussovský šum pomocou nástroja ImageMagick. Cieľom tohoto testu je otestovať robustnosť vybraného modelu voči šumu. Tabuľka 4.9 popisuje chybovú maticu pre tento experiment.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 130$	$FN = 116$
	Nesprávne	$FP = 8$	$TN = 261$

Tabuľka 4.9: Chybová matica pre test na zašumovanom videu.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{130 + 261}{130 + 261 + 8 + 116} = \frac{391}{515} = 75.92\% \tag{4.26}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{130}{130 + 8} = \frac{130}{138} = 94.2\% \tag{4.27}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{130}{130 + 116} = \frac{130}{246} = 52.84\% \tag{4.28}$$

Vyhodnotenie experimentu sa nachádza v rovniciach 4.26 až 4.28. Oproti pokusu bez šumu sa vyskytli tentokrát už aj False Positives. Tie sa nachádzajú predovšetkým v grafickom outre, alebo v prechodoch cez čiernu do inej scény. Ďalej bola šumom negatívne ovplyvnená detekcia zbraní, senzitivita klesla o 8.38% a celková presnosť o 5.63%. Z týchto výsledkov môžeme usúdiť, že algoritmus je pomerne robustný voči šumu.

4.7.3 Test na odšumovanom videu

V tejto sekcií sa zaoberáme zlepšením výsledkov vyhľadávania v zašumonvanom obraze. Na to sme pomocou programu ImageMagick použili mediánový filter o veľkosti 5x5. Mediánový filter sme použili kvôli jeho jednoduchosti a efektívnosti.

		Klasifikované hodnoty	
		Správne	**Nesprávne**
Skutočné hodnoty	**Správne**	$TP = 148$	$FN = 97$
	Nesprávne	$FP = 9$	$TN = 261$

Tabuľka 4.10: Chybová matica pre test na odšumovanom videu.

$$Presnost = \frac{TP + TN}{TP + TN + FP + FN} = \frac{148 + 261}{148 + 261 + 9 + 97} = \frac{409}{515} = 79.41\% \tag{4.29}$$

$$Preciznost = \frac{TP}{TP + FP} = \frac{148}{148 + 9} = \frac{148}{157} = 94.26\% \tag{4.30}$$

$$Senzitivita = \frac{TP}{TP + FN} = \frac{148}{148 + 97} = \frac{148}{245} = 60.4\% \tag{4.31}$$

Z chybovej matice (tab 4.10) a rovníc 4.29 - 4.31 môžeme pozorovať podobné výsledky ako pri nezašumovanom videu, čo svedčí o efektívnosti mediánového filteru. Výskyt false positives sa však pomocou tohoto filtra nepodarilo odstrániť.

Príklady výsledkov jednotlivých experimentov sú na obrázku 4.13.

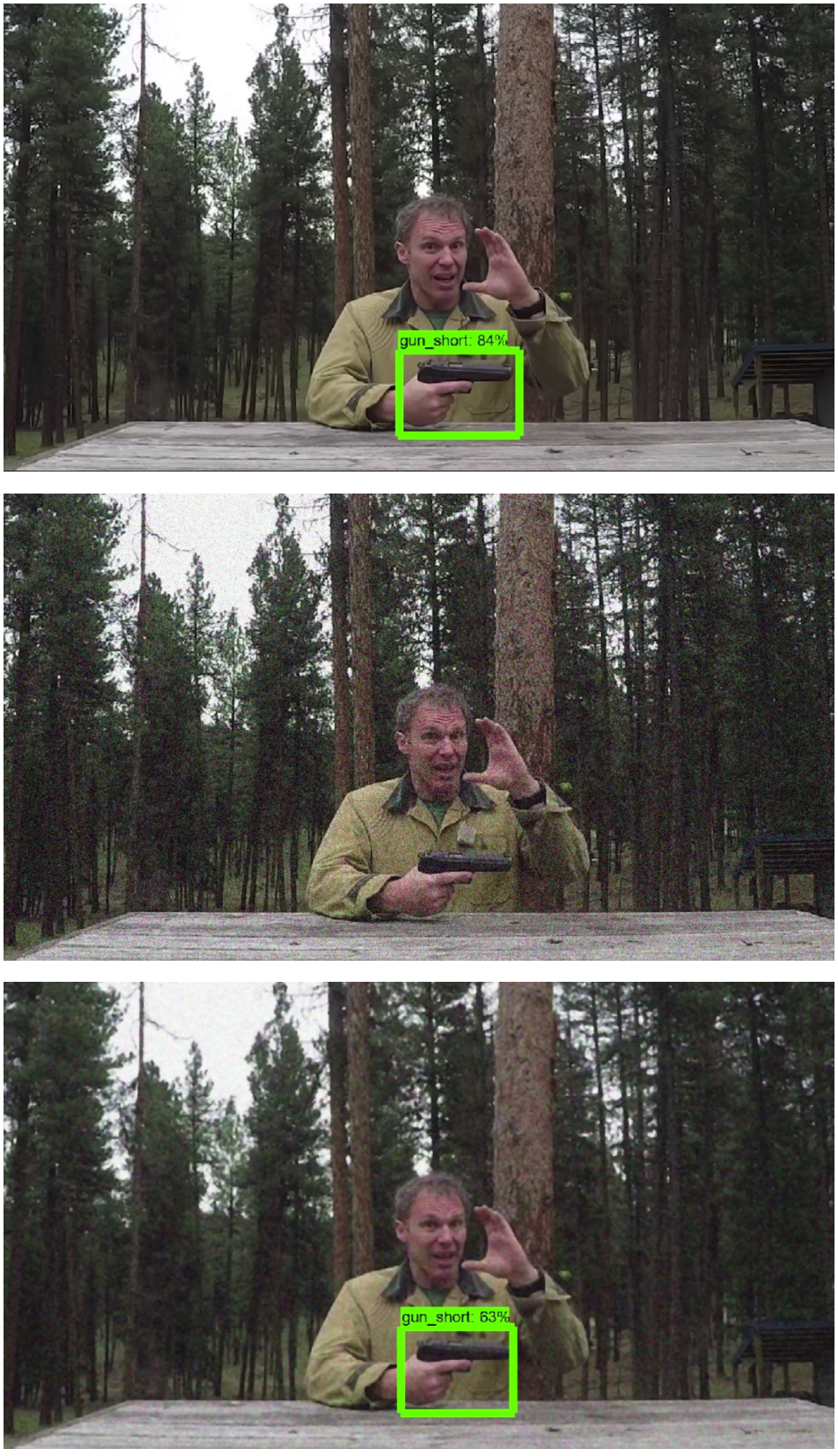
4.8 Zhrnutie

Za účelom detekcie zbraní v obraze boli použité tri detekčné algoritmy, pričom z dvoch z nich sme sa pokúsili vytvoriť čo najefektívnejšiu verziu. Algoritmus SURF sa ukázal pre tieto účely nevhodný. Pri jeho použití sa našla väčšina kľúčových bodov na pozadí, alebo prípadne na prechode medzi zbraňou a pozadím, čo viedlo k detekcií iba v prípade, v ktorom hľadáme zbraň vystrihnutú z danej scény (obrázok 4.7).

Detektor na bázi HoG a SVM mal zo začiatku priemerné výsledky, no ako sme začali postupne pridávať rotácie, tak sme predĺžili trénovací čas, aj čas pre detekciu v snímke, ktorý sa v najhorších prípadoch rovnal približne pätnástim sekundám pri použití 48 detektorov. Pre detekciu bolo použitých 24 rotácií zbrane držanej sprava a 24 rotácií zbrane držanej zľava, pričom každá rotácia sa od predošlej líšila o 15 stupňov. Ani pri týchto rotáciách sme nedosiahli najlepších výsledkov - bolo nájdených 39.28% zo všetkých pištolí (rovnica 4.12). Detekovali sme navyše iba pištole zboku, keby sme rozšírili detektor o viac orientácií (napríklad zbraň namierená do kamery), alebo viac strelných zbraní, tak by sa čas strávený detekciou presunul do abysmálnych hodnôt. Preto môžeme usúdiť, že tento algoritmus nájde väčšie použitie pre viac špecializované objekty, ako napríklad tváre na fotke, kde ľudia zvyčajne stoja a tým pádom sa nemusí riešiť toľko rotácií. Tváre sú navyše symetrické, takže nemusíme vytvárať detektory, ktoré zachycujú objekt zľava alebo sprava. Jedinými výhodami tohoto algoritmu je fakt, že pri detekcií neposkytoval veľké množstvo False Positive vyhodnotení a fakt, že pri použití málo detektorov sú trénovacie časy a časy na vyhľadávanie veľmi malé.

Najlepšie výsledky sme dosiahli pomocou detektoru SSD. Boli vytvorené tri modely, z ktorých najlepšie výsledky dosiahol tretí model, ktorý sme nechali trénovať 12 hodín a 41 minút, za ktorých sa vykonalo 112 156 trénovacích krokov. Tento model dokázal nájsť takmer tri štvrtiny zo všetkých zbraní v testovacej databáze a jeho celková presnosť činila 83.8%. Jeho veľmi dobrou vlastnosťou je, že pri dostatočnom trénovaní nedetekuje veľké množstvo False Positives. Čas pre detekovanie zbrane na jednej snímke zvyčajne nepresahoval 3 sekundy pre snímky do veľkosti 1920x1080.

Ďalej bol tento detektor testovaný na videu s rozlíšením 1280x720. Pri tomto videu bol priemerný čas na detekciu 2 sekundy a bolo nájdených 61.22% zo všetkých zbraní a celková presnosť bola 81.55%. Model preukázal veľmi dobrú schopnosť detekcie pri zbraniach, ktoré sa nachádzali v strednej až veľmi malej vzdialenosti od kamery, ale nedokázal detekovať zbrane vo väčšej vzdialenosti. Typicky neboli detekované zbrane o rozmeroch okolo 70x70 pixelov. Algoritmus zároveň demoštroval dobrú odolnosť voči šumu vo vstupnom videu. Ďalej sme sa presvedčili, že pre efektívnu redukciu šumu môžeme použiť mediánový filter o veľkosti 5x5. Namerané hodnoty presnosti a senzitivity pri použití mediánového filtru na zašumovaný obraz neklesli ani o 3%, čo je vynikajúci výsledok.



Obr. 4.13: Porovnanie detekcie pre pôvodný obraz (hore), zašumovaný obraz (v strede) a odšumovaný obraz (dole) metódou SSD.

4.9 Možnosti rozšírenia

Model SSD nedokázal detekovať zbrane malých rozmerov $\sim$ 70x70 pixelov a menšie. Do budúcnosti by bolo možné rozšíriť databázu o fotky zbraní z väčšej diaľky (teda na fotke by bola zbraň spolu aj s pozadím), pretože vo vytvorenej databázi sú zbrane označené tak, aby bolo v označení čo najmenej pozadia. Tento fakt môže mať za dôsledok zníženú detekciu pre malé rozmery zbraní.

Tento model taktiež mohol využiť napríklad na real-time detekciu zbraní pomocou systémov CCTV, kde by detekcia upozornila CCTV operátora na možné nebezpečenstvo. V tejto práci neboli dosiahnuté real-time časy detekcie z nasledujúcich dôvodov:

1. Zostava Lenovo Y500 nedokáže pre knižnicu Tensorflow utilizovať svoju grafickú kartu, a tak detekcia prebieha iba pomocou procesoru.

2. Využitý kód z Tensorflow Object Detection API je orientovaný skôr na výzkum než rýchlosť. Preto v ňom chýbajú optimalizácie pre rýchlosť detekcie a vykonávajú sa z hľadiska rýchlosti prebytočné operácie, ako napríklad vloženie obrázku do grafu, kde osi x a y zobrazujú počet pixelov.

3. Algoritmus je ďalej zaťažený diskovými operáciami načítania a uloženia.

Kapitola 5

Záver

V tejto práci sme sa zaoberali problematikou detekcie strelných zbraní v obraze. V teoretickej časti bol najprv uvedený pojem strelná zbraň, spolu so základnými archetypmi týchto zbraní. Následne bola uvedená kapitola z počítačového videnia ohľadom obrazového šumu, ktorý môže ovplyvňovať kvalitu detekcie. Ďalej bola venovaná pozornosť jednotlivým metódam strojového učenia. Boli vysvetlené neurónové siete a algoritmy pre detekciu v obraze, ktoré sú na nich založené - konkrétne konvolučné neurónové siete a Single Shot Multibox Detector, ktorý bol kvôli rýchlosti a presnosti uvedenej v práci [22] ďalej využitý v tejto práci na detekciu zbraní. Využité a vysvetlené boli aj algoritmy strojového učenia, ktoré nie sú založené na neurónových sieťach a to algoritmy SURF a HoG + SVM, ktoré sú všeobecne známe a využívané.

Ďalej bola vytvorená anotovaná databáza snímok, ktorá vo svojej finálnej fáze obsahovala 1265 obrázkov s 1406 zbraňami. Pre jej tvorbu bola využitá Internet Movie Firearms Database, ktorá obsahovala veľkú zásobu zbraní. Navyše dané zbrane sú držané hercami, a práve držané zbrane sú v tejto práci hlavnými objektmi detekcie.

Následne boli jednotlivé algoritmy implementované. Pre algoritmus HoG so SVM bola použitá knižnica dlib, pre SURF OpenCV a pre SSD bola použitá implementácia v Tensorflowe. Zároveň bol implementovaný program frameGetter, ktorý vyberá každú sekundu jednu snímku z videa. Snímky z tohto programu boli následne použité na hodnotenie na videu, pričom súčasťou tohoto hodnotenia boli aj snímky so šumom a jeho následnou filtráciou mediánovým filtrom. Z testovania na databáze aj videu a jeho variantách sa ako najlepší a najrobustnejší ukázal model SSD, ktorý dosiahol presnosť až 83.8%.

Detekcia pomocou SSD mala jednu chybu a to, že menšie objekty, veľké cca 70x70 pixelov sa nedali detekovať. Táto chyba by mohla byť odstránená priradením obrázkov s väčším pomerom pozadia ku zbrani (teda, zbraní zobrazených z väčšej diaľky). Ďalším rozšírením by mohlo byť napríklad optimalizovanie pre real-time detekciu na nasadenie do CCTV systémov.

Literatúra

[1] CS231n: Convolutional Neural Networks for Visual Recognition. [Online; navštívené 29.1.2019].
URL http://cs231n.github.io/convolutional-networks/

[2] Deep Learning with TensorFlow - Perceptron Tutorial. [Online; navštívené 20.1.2019].
URL https://www.simplilearn.com/what-is-perceptron-tutorial

[3] Zákon č. 190/2003 Z. z.: Zákon o strelných zbraniach a strelive a o zmene a doplnení niektorých zákonov. [Online; navštívené 20.4.2019].
URL https://www.zakonypreludi.sk/zz/2003-190#prilohy

[4] Barbu, T.: Variational Image Denoising Approach with Diffusion Porous Media Flow. *Abstract and Applied Analysis*, ročník 2013, č. 1, 2013: s. 1–8.
URL https://www.hindawi.com/journals/aaa/2013/856876/?fbclid=IwAR1h6mPKG7_-fqqvtYxoXYnRRgd3dUdBaj-WJbMq96lZTAqgkn7MXZAPi10

[5] Beneková, L.; Jakubcová, K.; Lesayová, A.; aj.: Kriminalistika 1. [Online; navštívené 22.4.2019].
URL http://www.1sg.sk/www/data/01/projekty/2016_2017/idols/web_kriminalistika1/index.html

[6] Cattin, P.: Image Restoration: Introduction to Signal and Image Processing. University of Basel, 2016, [Online; navštívené 11.1.2019].
URL https://miac.unibas.ch/SIP/06-Restoration.html#(1)

[7] Dataschool.io: Simple guide to confusion matrix terminology. [Online; navštívené 10.4.2019].
URL
https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[8] Farooque, M. A.; S.Rohankar, J.: SURVEY ON VARIOUS NOISES AND TECHNIQUES FOR DENOISING THE COLOR IMAGE. *International Journal of Application or Innovation in Engineering and Management*, ročník 2, č. 11, 2013: s. 217–221.
URL http://www.ijaiem.org/volume2issue11/IJAIEM-2013-11-24-070.pdf

[9] Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; aj.: Object Detection with Discriminatively Trained Part Based Models.
URL http://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf

[10] Forson, E.: Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning. [Online; navštívené 24.3.2019].

URL https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

[11] G. Howard, A.; Zhu, M.; Chen, B.; aj.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 04 2017.

[12] Gupta, T.: Deep Learning: Overfitting. [Online; navštívené 24.3.2019].
URL
https://towardsdatascience.com/deep-learning-overfitting-846bf5b35e24

[13] Herbert Bay, T. T., Andreas Ess; Gool, L. V.: Speeded-Up Robust Features (SURF). 2006.
URL http://www.vision.ee.ethz.ch/en/publications/papers/articles/eth_biwi_00517.pdf

[14] Jankových, R.: *Hlavňové zbraně a střelivo*. 2012, ISBN 978-80-260-2384-5.

[15] King, D. E.: Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, ročník 10, 2009: s. 1755–1758.

[16] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks.
URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[17] Kumar, A.: Artificial Neural Networks. Indian Agricultural Statistics Research Institute, [Online; navštívené 22.1.2019].
URL https://pdfs.semanticscholar.org/52ce/c0004295cce4a62685b5f2de6e315fedf94e.pdf

[18] Lagandula, A. C.: McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron. [Online; navštívené 19.1.2019].
URL https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

[19] Lagandula, A. C.: Perceptron: The Artificial Neuron (An Essential Upgrade To The McCulloch-Pitts Neuron). [Online; navštívené 20.1.2019].
URL https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

[20] LeCun, Y.; Bottou, L.; Bengio, Y.; aj.: Gradient-Based Learning Applied to Document Recogintion. [Online; navštívené 30.1.2019].
URL http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

[21] Lin, T.-Y.; Maire, M.; Belongie, S.; aj.: Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, editácia D. Fleet; T. Pajdla; B. Schiele; T. Tuytelaars, Cham: Springer International Publishing, 2014, ISBN 978-3-319-10602-1, s. 740–755.

[22] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. In *ECCV*, 2016.

[23] Mallick, S.: Histogram of Oriented Gradients.
URL https://www.learnopencv.com/histogram-of-oriented-gradients/

[24] Mallick, S.: Image Alignment (Feature Based) using OpenCV (C++/Python). [Online; navštívené 8.3.2019].
URL https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/

[25] Mathworks: Documentation - Integral Image. [Online; navštívené 19.4.2019].
URL https://www.mathworks.com/help/images/integral-image.html

[26] Narkhede, S.: Understanding Confusion Matrix. [Online; navštívené 10.4.2019].
URL https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[27] Olmos, R.; Tabik, S.; Herrera, F.: Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, ročník 275, 2018: s. 66–72.
URL https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/weaponsDetection/AutomaticHandgunDetection.pdf

[28] Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; aj.: Cats and Dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[29] Ray, S.: Understanding Support Vector Machine algorithm from examples (along with code).
URL https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[30] Russ, J. C.: *The Image Processing Handbook, Fifth Edition*. CRC Press, 2007, ISBN 0-8493-7254-2.

[31] Shiffman, D.: *The Nature of Code*. [Online; navštívené 22.1.2019].
URL http://wtf.tw/ref/shiffman.pdf

[32] Sonka, M.; Hlavac, V.; aj.: *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2005, ISBN 1-133-59369-0.

[33] Sontakke, M. D.; Kulkarni, M. S.: DIFFERENT TYPES OF NOISES IN IMAGES AND NOISE REMOVING TECHNIQUE. *International Journal of Advanced Technology in Engineering and Science*, ročník 3, č. 1, 2015: s. 102–115.
URL https://www.ijates.com/images/short_pdf/1420630126_P102-115.pdf

[34] Stecanella, B.: An introduction to Support Vector Machines (SVM). [Online; navštívené 2.4.2019].
URL https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[35] Torres, J.: Learning process of a neural network. [Online; navštívené 23.1.2019].
URL https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7

[36] Tran, D.: How to train your own Object Detector with TensorFlow's Object Detector API. [Online; navštívené 30.3.2019].
URL https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9

[37] Wang, C.-F.: A Basic Introduction to Separable Convolutions. [Online; navštívené 29.3.2019].
URL https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

[38] Yegulalp, S.: What is TensorFlow? The machine learning library explained. 06 2018.
URL https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

[39] Yuan, E.: RANSAC. [Online; navštívené 8.3.2019].
URL http://eric-yuan.me/ransac/

[40] ZbraněKvalitně.cz: Nauka o zbraních. [Online; navštívené 22.4.2019].
URL https://zbranekvalitne.cz/zbrojni-prukaz/nauka-o-zbranich

Príloha A

Obsah priloženého DVD

Súčasťou práce je aj priložené DVD s nasledujúcou štruktúrou:

- **HOG** - Adresár so zdrojovými kódmi, ktoré boli použité na experimenty s algoritmom HOG.
- **SURF** - Adresár so zdrojovými kódmi, ktoré boli použité na experimenty s algoritmom SURF.
- **SSD** - Adresár, ktorý obsahuje použité Tensorflow Object Detection API nastavené na experimenty so SSD.
- **frameGetter** - Adresár so zdrojovými kódmi vytvoreného programu *frameGetter*, dodávaný spolu s testovacím videom.
- **gun_db** - Adresár obsahujúci vytvorenú databázu obrázkov
- **bp_text** - Adresár obsahujúci zdrojové súbory pre generovanie technickej správy.
- **xdebna01_BP.pdf** - Táto bakalárska práca v .pdf formáte.
- **README** - Používateľská príručka.