VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE A KLASIFIKACE VOJENSKÝCH CÍLŮ VE VIDEOSIGNÁLU

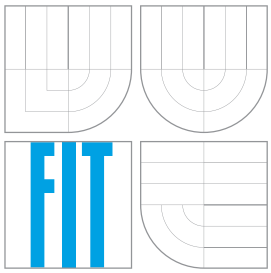
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL KOŠÍK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE A KLASIFIKACE VOJENSKÝCH CÍLŮ VE VIDEOSIGNÁLU

DETECTION AND CLASSIFICATION OF MILITARY TARGETS IN A VIDEOSIGNAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL KOŠÍK
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Cílem této práce je návrh a implementace vhodných algoritmů na detekci a klasifikaci vzdálených cílů. Čtenář bude seznámen s dvojicí algoritmů pro detekci pohybu. Jedná se o odčítání dvou po sobě jdoucích snímků a o komplexnější algoritmus, který je založený na Bayesovském klasifikátoru. Dále mu budou představeny dva možné způsoby klasifikace, konkrétně pomocí support vector machines a pomocí množiny lineárních klasifikátorů. V závěru práce budou dané algoritmy zhodnoceny.

Abstract

The aim of this thesis is to design and implement algorithms for detection and classification of distant targets. The reader will become familiar with algorithms for detection of a movement, namely image differentiation and an algorithm based on a Bayes classifier. In the following two possible ways of classification, one using support vector machines and the second one utilizing a set of linear classifiers, will be introduced. At the end of this thesis results of the algorithms will be described and evaluated.

Klíčová slova

OpenCV, termální kamera, detekce, klasifikace, vzdálené cíle, support vector machines, lineární klasifikátor

Keywords

OpenCV, thermal camera, detection, classification, distant targets, support vector machines, linear classifier

Citace

Michal Košík: Detekce a klasifikace vojenských cílů ve videosignálu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Detekce a klasifikace vojenských cílů ve videosignálu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing., Dipl.-Ing. Martina Drahanského, Ph.D.

.....
Michal Košík
19. mája 2010

Poděkování

Týmto by som sa chcel poďakovať svojmu vedúcemu Doc. Ing., Dipl.-Ing. Martinovi Drahanskému, Ph.D. a konzultantovi Ing. Filipovi Orságovi, Ph.D. za odbornú pomoc pri tvorbe tejto práce a konzultácie s ňou spojené. Ďalej by som sa rád poďakoval Doc. Ing. Teodorovi Balážovi, CSc. za pomoc pri tvorbe videí a v neposlednom rade aj svojej rodine a priateľom za ich pomoc a podporu a Stefanie Naumann za prechádzky, ktoré mi dodali silu pokračovať v práci.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce je navrhnúť vhodné algoritmy a postupy, ktoré nám umožnia detekciu a klasifikáciu vzdialených objektov s malým rozlíšením za pomoci termálnej kamery príp. noktovízoru a implementovať ich v rámci aplikácie. Táto aplikácia by potom bola schopná samostatne rozhodovať o triede objektu detekovaného v obraze.

Dôvodom pre zvolenie tohoto zadania bola rastúca potreba klasifikácie a rozpoznávania v rôznych oblastiach života– automatizované systémy pracujúce v reálnom čase dokážu pracovať bez prestávky, nepotrebujú žiadny alebo len minimálny ľudský dozor a často aj vykazujú lepšie výsledky. Takisto môžu slúžiť v oblasti bezpečnosti na detekciu a prevenciu hrozieb.

Text tejto práce je rozdelený do šiestich kapitol a záveru.

Druhá kapitola obsahuje úvod do problematiky. Zoznámime sa tu s problematikou týkajúcou sa snímacích sústav, bližšie si predstavíme termálnu kameru a noktovízor. Takisto tu budú zodpovedané niektoré otázky týkajúce sa spracovania a predspracovania obrazu, detekcie a klasifikácie.

Tretia kapitola pojednáva o postupe pri návrhu aplikácie, o zvolenom jazyku a rozdelení aplikácie na logické celky.

Štvrtá kapitola nám priblíži spôsob získania dát a vytvorí nám približný obraz o tom, s akými dátami budeme pracovať.

Piata kapitola obsahuje popis samotnej implementácie. Nájdeme tu rozpísané navrhnuté algoritmy a obsahuje aj ukážku trénovania support vector machines.

Šiesta kapitola sa venuje testovaniu. Sú v nej rozobrané vlastnosti, výhody a nevýhody použitých algoritmov a zároveň navrhnuté možné vylepšenia do budúcnosti.

Kapitola 2

Úvod do problematiky

V tejto kapitole sa zoznámime so základnými pojmami a princípmi, ktoré súvisia s problematikou tejto práce a ktoré budú ďalej využívané.

2.1 Knižnica OpenCV

OpenCV je open-source knižnica, ktorá vznikla z iniciatívy firmy Intel. Jej hlavným cieľom bolo vytvoriť nástroj pre podporu počítačového videnia so silným zameraním na aplikácie bežiace v reálnom čase. OpenCV je knižnica napísaná v jazykoch C a C++ a je multiplatformová, čo znamená, že funguje pod operačnými systémami Linux, Windows a MacOS a taktiež aj na niektorých vstavaných zariadeniach.

Prvá verzia tejto knižnice bola vydaná v roku 1999. Odvtedy bola ďalej vyvíjaná a zdokonaľovaná a aktuálna verzia (2.1) obsahuje vyše 500 optimalizovaných funkcií [18], medzi ktoré patria aj funkcie na sledovanie objektu v scéne, kalibráciu kamery, stereo-víziu a mnohé ďalšie. Knižnica je rozdelená do nasledujúcich častí:

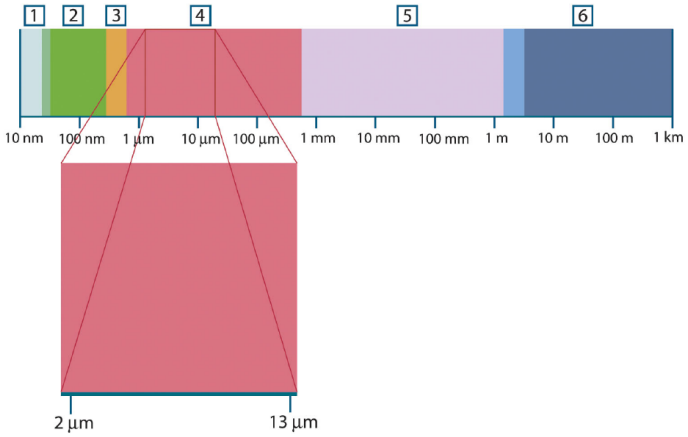
- CXCORE – základné štruktúry a algoritmy
- CV – spracovanie obrazu a algoritmy na podporu počítačového videnia
- HighGUI – základné užívateľské rozhranie, podpora načítania, zobrazenia a uloženia videa a obrázkov
- MLL – algoritmy pre počítačové učenie, štatistické klasifikátory a algoritmy na zhlukovanie (clustering)
- CvAux – experimentálne algoritmy, stereo vízia, 3D sledovanie, ...; táto časť knižnice však ešte nie je dostatočne zdokumentovaná

Všetky tieto časti dávajú dohromady efektívny nástroj na riešenie väčšiny problémov súvisiacich s počítačovým videním.

Táto podkapitola bola čerpaná z [3].

2.2 Elektromagnetické spektrum

Elektromagnetické spektrum zahrňuje elektromagnetické žiarenie všetkých možných vlnových dĺžok – nemá žiadnu principiálnu spodnú ani hornú hranicu [6]. Človekom bolo umelo



Obrázok 2.1: Elektromagnetické spektrum. 1: Röntgénové žiarenie; 2: Ultrafialové žiarenie; 3: Viditeľné spektrum; 4: Infračervené žiarenie; 5: Mikrovlny; 6: Rádiové vlny [5]

rozdelené na niekoľko oblastí podľa toho, aké metódy sa používajú na produkovanie a detekciu daného typu žiarenia. Rozdelenie sa nachádza na obrázku 2.1.

Termografia využíva infračervenú oblasť. Táto oblasť je zvyčajne ešte ďalej (takisto umelo) rozdelená na 5 podoblastí [7]:

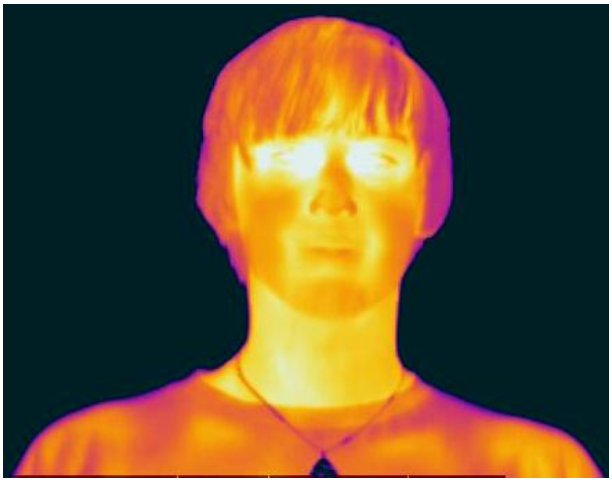
- blízke infračervené žiarenie (skr. NIR) — 0,7 – 1,1 μm
- IR krátkej vlnovej dĺžky (skr. SWIR) — 1,1 – 2,5 μm
- IR strednej vlnovej dĺžky (skr. MWIR) — 2,5 – 7 μm
- IR dlhej vlnovej dĺžky (skr. LWIR) — 7 – 15 μm
- IR veľmi dlhej vlnovej dĺžky (skr. VLWIR) — 15 – 1000 μm

2.3 Termálna kamera

Termálna kamera je zariadenie, ktoré meria a zaznamenáva (príp. okamžite zobrazuje) infračervené žiarenie vychádzajúce z objektu. Jednoducho povedané, všetky objekty s teplotou nad 0° K vykazujú pohyb svojich molekúl. Tento pohyb produkuje tepelnú energiu, ktorá ale nie je vysielaná vo viditeľnom, ale v infračervenom spektre, ktoré je ľudskému oku neviditeľné. Termálne kamery detekujú vyžarovanú infračervenú energiu a elektronicky spracúvajú túto informáciu do videa príp. obrazu, aby sme mohli „vidieť“ vyžiarenú energiu [5].

Toto je rozdiel od ostatných obraz-snímacích zariadení, ktoré v scéne potrebujú aspoň isté množstvo viditeľného svetla, zatiaľ čo termálna kamera nám umožňuje vidieť objekty aj v absolútnej tme.

Vzhľadom k tomu, že infračervené žiarenie má väčšiu vlnovú dĺžku ako viditeľné svetlo, termálne kamery sú schopné detekovať vyžarovanú energiu aj cez dym, prach, hmlu, dážď a sneh. Keďže ale vyžarovanie je funkciou povrchovej teploty objektu, nedokážeme vidieť cez objekt, dokonca ani cez sklo. Preto musí byť optika týchto kamier vyrobená zo špeciálnych látok, ako je napr. germánium, ktoré prepúšťa až 95 % infračerveného žiarenia [9].



Obrázok 2.2: Výstupný obraz z termálnej kamery; najteplejšie regióny sú najsvetlejšie

Teplota detekovaná kamerou však nezávisí iba na teplote objektu, ale aj na ďalších faktoroch, ako je emisivita (veličina, ktorá hovorí, koľko žiarenia je vyžarovaného z objektu v porovnaní s dokonalým čiernym telesom s rovnakou teplotou), vzdialenosť od kamery, okolité žiarenie dopadajúce na objekt a odrážajúce sa od neho a v neposlednom rade aj pohltenie istej časti žiarenia atmosférou [5].

Termálne kamery je možné rozdeliť do dvoch základných skupín – na chladené a nechladené [7]. Rozdiel medzi nimi je spôsob, akým je stabilizovaný tepelný detektor kamery.

2.3.1 Chladené termálne kamery

Pôvodne boli všetky termálne kamery chladené. Detektor takejto kamery je kryogenicky chladený na teploty pod 0° C, čo značne zvyšuje citlivosť kamery. Niektoré chladené termálne kamery sú schopné detekovať vyžarovanie ľudského tela až do vzdialenosti 20 km [9]. Ich hlavná nevýhoda je ich vysoká cena a takisto aj nutnosť ich pravidelne rekalibrovať.

2.3.2 Nechladené termálne kamery

Oproti tomu nechladené termálne kamery stabilizujú detektor elektronicky. Ešte pred niekoľkými rokmi nedosahovali citlivosť chladených kamier, no ako technológia pokročila, dnes sa im už v citlivosti a dosahu takmer vyrovnajú [7]. Nepotrebujú pravidelnú rekalibráciu a ich cena sa zvyčajne pohybuje na úrovni jednej tretiny ceny chladených kamier.

2.3.3 Výstupný obraz

Výstupom z kamery je sekvencia snímkov, kde každá teplota má pridelenú určitú farbu. Podľa nastavenia kamery môžu byť najteplejšie oblasti na snímke najsvetlejšie až biele, najchladnejšie naopak tmavé až čierne (viď. obr. 2.2) alebo aj inverzne, príp. je výstup v úrovniach šedej.

2.3.4 Dosah termálnej kamery

Zvyčajne prvá otázka, ktorá nás napadne, keď prídeme do styku s termálnou kamerou, je „Ako ďaleko s ňou dokážeme vidieť?“. Na túto otázku ale neexistuje jednoznačná odpoveď. Väčšinou sú totiž tieto kamery schopné vidieť Slnko, ktoré je od Zeme vzdialené viac ako 146 miliónov km. To ale neznamená že sú schopné na rovnakú vzdialenosť detekovať aj ostatné objekty.

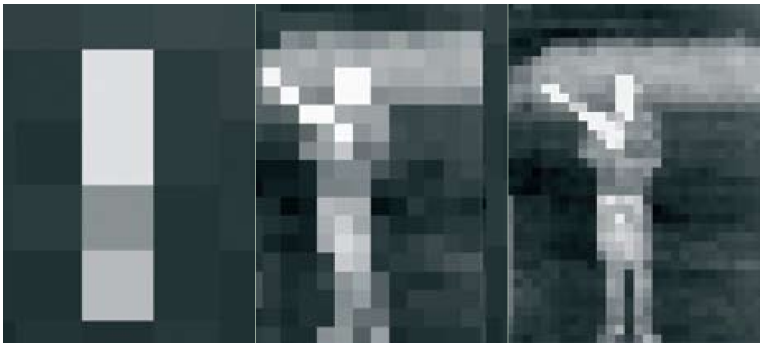
Vzdialenosť, na ktorú sme schopní vidieť objekt termálnou kamerou sa volá dosah (anglicky *range* [7]). Do výpočtu tejto vzdialenosti vstupujú rôzne premenné vrátane typu kamery, typu objektívu, veľkosti a vlastností objektu, atmosferických podmienok a takisto aj definíciu samotného slova „vidieť“.

2.3.5 Johnsonove kritériá

Na definíciu slovného spojenia „vidieť objekt“ môžme použiť takzvané Johnsonove kritériá, ktoré sa často používajú na charakterizovanie termálnych kamier. Tieto kritériá rozlišujú medzi 3 úrovňami videnia:

- Detekcia – sme schopní detekovať, či sa objekt na obraze nachádza alebo nie. Na toto potrebujeme, aby bola kritická dimenzia objektu pokrytá aspoň 1,5 pixelom.
- Rozpoznanie – sme schopní určiť, o aký objekt sa jedná, teda napr. či náš objekt je osoba, auto alebo kamión. Na rozpoznanie potrebujeme aspoň 6 pixelov cez kritickú dimenziu objektu
- Identifikácia – tento pojem je často používaný vo vojenskej terminológii, ktorý znamená odlíšenie priateľa od nepriateľa. Aby sme to boli schopní dosiahnuť, kritická dimenzia objektu musí byť pokrytá aspoň 12 pixelmi

Tieto kritéria nám dávajú 50 % pravdepodobnosť správneho zaradenia objektu v rámci jednej úrovne videnia.



Obrázok 2.3: DRI cieľa na základe jeho rozlíšenia [4]

Bližšie si to vysvetlíme na obrázku 2.3. Kritická dimenzia pre človeka je 0,75 m. Táto hodnota bola zistená pomocou empirických pozorovaní vyplývajúcich zo štatistickej analýzy termálnych dát. Na detekciu, rozpoznanie a identifikáciu (skrátene DRI) potrebujeme $1, 5$, 6 a 12 pixelov cez 0,75 metra v obraze, z čoho vyplýva:

- Detekcia: $1, 5$ pixelu / $0, 75$ m $= 2$ pixely na meter.

- Rozpoznanie: 6 pixelov / 0,75 m = 8 pixelov na meter.
- Identifikácia: 12 pixelov / 0,75 m = 16 pixelov na meter.

Predpokladajme, že osoba je vysoká 1,8 m a široká 0,5 m a má dobrý kontrast voči okoliu. Na detekciu potrebujeme, aby objekt zaberal aspoň 3,6 krát 1 pixel (obr. 2.3 vľavo – vidíme, že sa na obraze niečo nachádza). Pre rozpoznanie musí mať objekt rozmery aspoň 14,4 krát 4 pixely (obr. 2.3 v strede – vieme rozpoznať, že na obraze sa vyskytuje nejaká osoba). Na identifikáciu potrebujeme najmenej 28,8 krát 8 pixelov (obr. 2.3 vpravo – vidíme osobu držiacu zbraň).

Táto podkapitola bola čerpaná z [4].

2.4 Noktovízor

Noktovízor (takisto aj zariadenie na nočné videnie) na rozdiel od termálnej kamery vyžaduje aspoň nejaké množstvo viditeľného svetla, napr. svetlo vyžarované hviezdami alebo odrazené od mesiaca, na to, aby mohol efektívne fungovať. Noktovízor zachytáva aj veľmi slabé svetlo a zosilňuje ho, čím nám umožňuje vidieť aj za zlých svetelných podmienok. Pokiaľ ale na snímanú scénu nedopadá žiadne svetlo, noktovízor nebude fungovať vôbec. Výsledný obraz môže byť taktiež ovplyvnený dažďom, snehom, dymom alebo hmlou. Hlavná výhoda týchto zariadení je ich cena, ktorá je rádovo nižšia ako u termálnych kamier [9], [7].

Obrázok 2.4: Výstupný obraz z noktovízoru [10]

2.5 Snímacie sústavy

Táto časť sa týka snímacích sústav vo všeobecnosti, nie iba termálnych kamier či noktovízorov. V reálnom svete môžu obecne nastať 4 situácie podľa relatívneho pohybu snímacej sústavy a snímanej scény [8]:

- statická kamera, statické objekty
- statická kamera, dynamické objekty

- dynamická kamera, statické objekty
- dynamická kamera, dynamické objekty

O pohybe v obraze hovoríme v posledných 3 zmienených prípadoch.

2.6 Digitalizácia

Na to, aby bolo možné jednotlivé snímky z kamery strojovo spracovať, musia byť prevedené do digitálnej podoby.

Reálna scéna,ktorú sníma kamera, má teoreticky nekonečný rozsah obrazových hodnôt. Bude ale zobrazená s konečným množstvom pixelov a v konečnom počte farieb.

Digitalizácia je prechod od spojitej funkcie $f(x, y)$ (za využitia kvantovania a vzorkovania) k diskrétnej hodnote $I_{i,j}$ a to ako v definičnom obore funkcie $f(x, y)$, tak aj v jej obore hodnôt H [21]. Takto transformovaný obraz je už možné počítačovo spracovať.

2.7 Predspracovanie obrazu

Obraz, nad ktorým chceme vykonávať nami požadované operácie, ako je napr. vyhľadávanie alebo sledovanie, býva zvyčajne nejakým spôsobom predspracovaný. V tejto časti si vysvetlíme základné operácie používané práve na predspracovanie obrazu.

2.7.1 Prevod do odtieňov šedej

Obraz v odtieňoch šedej je obraz, ktorého každý bod obsahuje informáciu o jase. Tento prevod sa z RGB obrazu uskutoční za použitia vzorca [12], [21]:

$$I = 0,299R + 0,587G + 0,114B$$

Takto prevedený obraz má za cenu straty istého množstva informácii menšiu veľkosť a dá sa rýchlejšie spracovať.

2.7.2 Vyhladzovanie

Vyhladzovanie (anglicky *smoothing* alebo *blurring* [3]) sa často používa na odstránenie šumu v obraze alebo chýb spôsobených prevodom alebo optikou kamery. Tiež je ho možné využiť, keď chceme značne zmenšiť rozlíšenie obrazu.

Pri vyhladzovaní dochádza ku vzniku redundantnej informácie. Najjednoduchší algoritmus na vyhladzovanie je nahradenie hodnoty pixelu spriemerovanou hodnotou okolitých pixelov. Keďže pixel obsahujúci šum má značne odlišnú hodnotu, dôjde k jej eliminácii resp. rozprestretí do okolitých pixelov [3].

2.7.3 Prahovanie

Prahovanie (anglicky *thresholding* [12]) sa zvyčajne prevádza na obraze v odtieňoch šedej. Spočíva v rozdelení jasovej škály na dve časti a nahradenie každej z nich jedinou hodnotou podľa vzťahu:

$$i' = \begin{cases} L, & i < T \\ H, & i \geq T, \end{cases}$$

kde i je vstupná hodnota, i' je výstupná hodnota, T je prah a L a H sú dve výstupné hodnoty. Všetky hodnoty s intenzitou nižšou ako má prah T sú nahradené hodnotou L, s intenzitou vyššou alebo rovnou prahu T sú naopak nahradené hodnotou H [21].

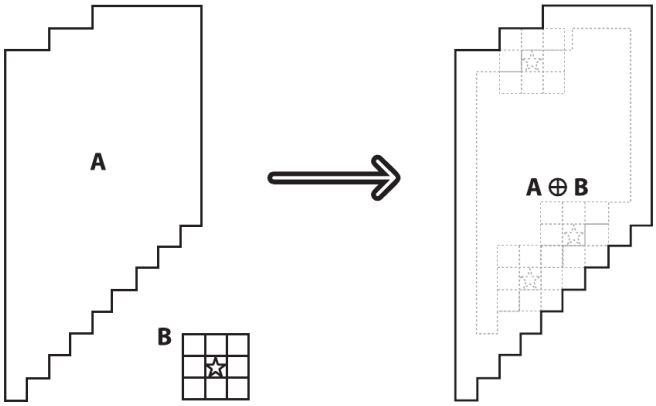
2.7.4 Dilatácia

Dilatácia (anglicky *dilation*) je konvolúcia určitého obrazu, ktorý budeme nazývať A, s istým jadrom (anglicky *kernel*), ktoré budeme nazývať B. Jadro môže byť ľubovoľného tvaru a veľkosti a má práve jeden definovaný „kotviaci bod“ (anglicky *anchor point*) [3]. Zvyčajne je jadro štvorec s kotviacim bodom uprostred. O jadre sa dá uvažovať ako o maske. Presúvaním jadra B cez obraz A, hľadáme maximálnu hodnotu pixelov prekrytých jadrom B a tú dosadíme do pixelu umiestneného pod kotviacim bodom. Tento proces spôsobí, že jasné oblasti v obraze rastú (viď obr. 2.5)

Matematický zápis dilatácie je [3]:

$$dilate(x, y) = \max_{(x',y')\in kernel} src(x + x', y + y')$$

Hlavné využitie dilatácie je pri obraze, ktorý prešiel prahovaním. V tomto prípade máme iba 2 možné hodnoty a dilatácia tu pomáha nájsť tzv. „prepojené komponenty“ (anglicky *connected components* [3]), čo sú tie časti obrazu, ktoré by mali tvoriť jeden celok, ale vplyvom napr. šumu sa rozdelili na niekoľko častí. Dilatácia ich (za cenu zväčšenia výsledného celku) dokáže opätovne prepojiť [14].



Obrázok 2.5: Princíp dilatácie [3]

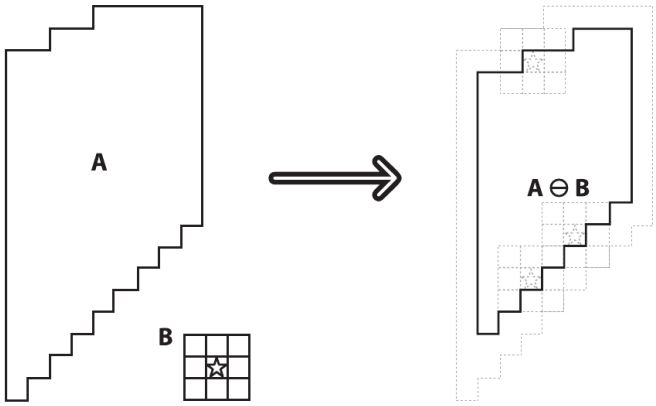
2.7.5 Erózia

Erózia (anglicky *erosion* [3]) je opačná operácia ako dilatácia. Použitie erózie sa rovná nájdeniu lokálneho minima v oblasti jadra. Erózia generuje nový obraz z originálneho za použitia nasledujúceho algoritmu: ako sa jadro B presúva cez obraz, hľadáme minimálnu hodnotu pixelov prekrytých jadrom B, ktorú následne dosadíme do pixelu umiestneného pod kotviacim bodom. Tento proces spôsobí, že tmavé oblasti v obraze rastú (viď obr. 2.5).

Matematický zápis erózie je:

$$erode(x, y) = \min_{(x',y')\in kernel} src(x + x', y + y')$$

Vo všeobecnosti, kdekoľvek dilatácia rozšíri oblasť A, tam ju erózia zmenší [3], [14].



Obrázok 2.6: Princíp erózie [3]

2.7.6 Morfologický gradient

Ďalšia operácia, ktorú tu zmienime, je morfologický gradient. Najjednoduchšie bude, keď začneme priamo vzorcom [3]:

$$gradient(src) = dilate(src) - erode(src)$$

Výsledkom tejto operácie použitej na binárny resp. prahovaný obraz je izolovanie hraníc existujúcich oblastí, ako ukazuje obr. 2.7.

Pri obraze v odtieňoch šedej nám táto operácia ukáže, ako rýchlo sa v ňom mení jas.

Morfologický gradient sa často používa, keď chceme izolovať hranice svetlých oblastí, aby sme s nimi mohli jednať ako s jedným celistvým objektom (príp. ako s celistvými časťami objektov) [3].

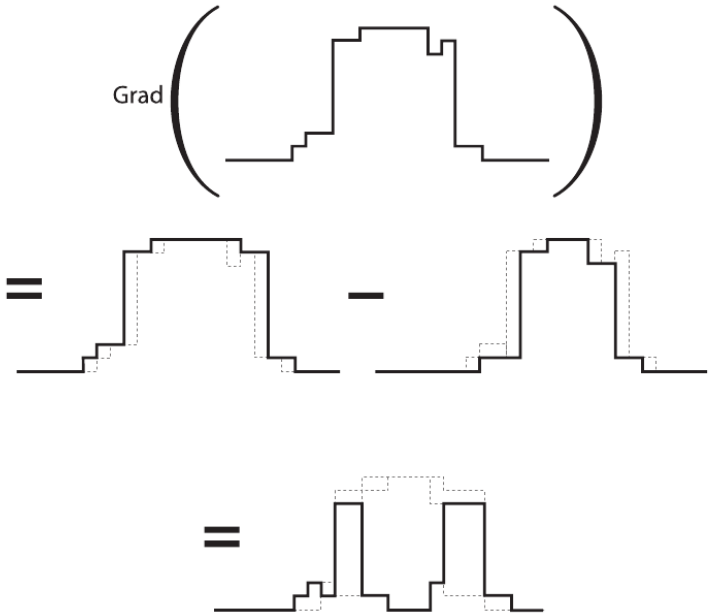
2.8 Segmentácia

Cieľom segmentácie je izolovať objekty alebo ich časti zo zvyšku obrazu. Dôvod je jednoduchý – keď napr. statická kamera sleduje nejakú oblasť, zvyčajne neustále sníma tú istú scénu (to isté pozadie), v ktorej sa nachádzajú objekty, ktoré nás nezaujímajú. Čo nás ale môže zaujímať sú osoby a vozidlá, ktoré náhle vstúpia do scény. Presne tieto udalosti by sme radi zaznamenali a príp. ďalej spracovali, rovnako ako by sme radi ignorovali dobu, po ktorú sa v scéne nič nedeje. Na to potrebujeme vedieť oddeliť popredie od pozadia.

Existuje množstvo algoritmov na oddelenie pozadia (veľmi dobrý prehľad a porovnanie rôznych algoritmov sa nachádza v [17]), no my sa v tejto časti budeme zaoberať iba jediným – jednoduchým odčítavaním snímkov.

2.8.1 Odčítavanie snímkov

Jedná sa o jednoduchú metódu na detekciu pohybu v obraze založenú na zistení rozdielov v obrazoch nasnímaných v rôznych časových okamihoch. Pokiaľ sa odpovedajúce obrazové



Obrázok 2.7: Princíp gradientu [3]

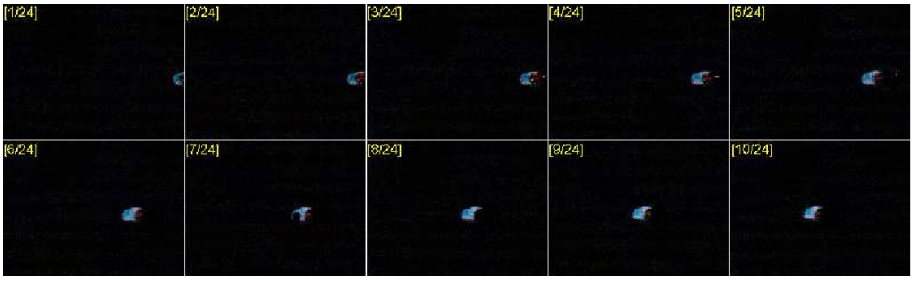
body líšia rozdielom jasových hodnôt o viac, ako stanovenú hranicu ξ, je na príslušnú pozíciu binárneho rozdielového obrazu zanesená hodnota 1 (čo je vlastne aplikácia prahovania). Pokiaľ snímané obrazy označíme $f_1(x, y)$ a $f_2(x, y)$ a zavedieme hodnotu ξ určujúcu prah rozdielnosti bodov, je možné výsledný binárny rozdielový obraz d definovať ako:

$$d(x,y) = \begin{cases} 0 & |f_1(x,y) - f_2(x,y)| < \xi \\ 1 & |f_1(x,y) - f_2(x,y)| \geq \xi \end{cases}$$

Obrázok 2.8: Jednotlivé snímky z kamery [8]

Jednotková hodnota (reprezentujúca časť obrazu, kde nastal pohyb) so súradnicami x a y vo výstupnom binárnom súbore môže byť spôsobená niekoľkými stavmi:

- $f_1(x, y)$ je prvok pohybujúceho sa objektu a $f_2(x, y)$ je prvok nepohybujúceho sa pozadia alebo naopak.
- $f_1(x, y)$ je prvok pohybujúceho sa objektu a $f_2(x, y)$ je prvok iného pohybujúceho sa objektu.



Obrázok 2.9: Výsledok odčítania dvoch po sebe idúcich snímkov [8]

- $f_1(x, y)$ a $f_2(x, y)$ sú prvky rovnakého pohybujúceho sa objektu v mieste rôzneho jasu.
- vplyvom šumu a iných nepresností pri snímaní.

Na obrázku 2.8 vidíme sekvenciu snímkov nasnímaných kamerou a na obrázku 2.9 zasa výsledok použitia odčítania dvoch po sebe idúcich snímkov. Ako môžme z týchto obrázkov vidieť, je možné pomocou jednoduchého odčítavania snímkov detekovať prítomnosť pohybu v obraze, ale už nie jeho smer ani veľkosť.

Táto podkapitola bola čerpaná z [8].

2.9 Strojové učenie

Strojové učenie (anglicky *machine learning* [11]) sa snaží premeniť vstupné dáta na výstupné informácie. Po naučení sa z určitej kolekcie dát chceme, aby stroj bol schopný zodpovedať otázky o dátach, napr.: „Čo je najviac podobné tomuto súboru dát?“, „Je na obrázku lietadlo?“, „Na akú reklamu užívateľ odpovie?“ a pod. Aby sme to zhrnuli, strojové učenie premieňa dáta na informácie tým, že z nich extrahuje určité príznaky alebo vytvára určité pravidlá.

Strojové učenie môže byť s učiteľom alebo bez učiteľa. V prípade učenia s učiteľom majú testovacie dáta označenie, ktoré hovorí o ich príslušnosti k určitej triede. Pokiaľ sa jedná o učenie bez učiteľa, dáta nenesú žiadne značenie a snažíme sa ich rozdeliť iba podľa ich vlastností – používajú sa tu rôzne zhlukovacie metódy [3].

2.10 Extrakcia príznakov

Z obdržaných dát (zvyčajne potom, čo sme sa uistili, že sú relevantné), potrebujeme vyextrahovať určité príznaky (tento proces sa anglicky nazýva *features extraction* [3]). Tieto príznaky by mali byť [1]:

- gaussovského rozloženia
- nekorelované
- nízkodimenzionálne

Takéto príznaky získané z viacerých dát potom môžme navzájom porovnávať, zisťovať ich podobnosť v rámci dát patriacich do jednej kategórie a v konečnom dôsledku na ich základe vybudovať príp. natrénovať klasifikátor. Príznaky sa zvyčajne predávajú vo forme tzv. „príznakového vektora“ [20].

2.11 Klasifikácia

Klasifikácia (anglicky *classification* [11]) je priradenie určitej triedy k objektu. Pod pojmom trieda sa rozumie množina dát s rovnakými vlastnosťami – v jej vnútri sú si dáta podobnejšie ako medzi triedami navzájom [20]. Vstupom klasifikátora sú príznaky predávané vo forme príznakového vektora. Klasifikátor potom na základe tohoto vektora dokáže určiť, do akej triedy daný objekt patrí.

V praxi sa najčastejšie vyskytujú dve skupiny klasifikátorov:

- binárne klasifikátory – rozhodujú medzi dvoma triedami, zvyčajne ide o rozhodnutie áno – nie (odpoveď na otázku: „Patrí objekt do danej triedy?“)
- klasifikátory pre viac tried – rozhodujú priamo o príslušnosti k jednej z tried

2.11.1 Problémy pri klasifikácii

Pri trénovaní klasifikátorov platia dve základné pravidlá [3]:

- je lepšie mať viac trénovacích dát ako menej
- je lepšie vybrať kvalitnejšie príznaky ako kvalitnejšie trénovacie algoritmy

Pokiaľ sa nám podarí vybrať vhodné príznaky (ktoré navzájom nebudú závislé a nebudú sa moc meniť v závislosti od podmienok), potom takmer každý trénovací algoritmus pracuje dobre.

Základné problémy, s ktorými sa môžme stretnúť, sú [3]:

- klasifikátor sa dokonale naučí trénovaciu sadu dát, zapamätá si ju ale aj so šumom a v reálnom nasadení nie je schopný správne klasifikovať – v tomto prípade je vhodné zvoliť menej príznakov, príp. viac trénovacích dát, ktoré môžu pomôcť vyhladiť šum
- predpoklady o výzore modelu sú príliš zidealizované pre dané dáta, teda model úplne nesedí – výber ďalších príznakov alebo použitie mocnejšieho algoritmu môže dopomôcť k lepšej zhode

Ďalší problém lepšie pochopíme na nasledujúcom príklade – chceme vytvoriť klasifikátor, ktorý na základe istých vlastností (príznakov) húb dokáže tieto rozdeliť na jedlé a jedovaté.

V tomto prípade je nemožné pozbierať všetky huby a vyextrahovať z nich príznaky, ktoré nás zaujímajú. Musíme sa uspokojiť iba s istou reprezentatívnou vzorkou pre jedovaté a pre jedlé huby. Keď si z nich vyextrahujeme príznaky a dáme si ich zobraziť, zistíme, že sa nám tieto dve triedy istým spôsobom prekrývajú. Nech stanovíme hranicu medzi triedami akokoľvek, nikdy sa nám nepodarí tieto 2 triedy úplne separovať.

Tento problém sa nazýva „cena rozhodnutia“ (anglicky *cost* [3]). Posunutím rozhodovacej hranice môžme dosiahnuť, že náš klasifikátor odhalí všetky jedlé huby, no raz za čas za jedlú označí aj jedovatú. Alebo naopak, vždy správne rozpozná jedovaté huby, no za cenu, že medzi nimi budú aj jedlé.

Nastavenie tejto hranice závisí na riešenom probléme – pokiaľ chceme predávať jedlé huby do potravín, nastavíme klasifikátor tak, aby sme minimalizovali šancu, že sa medzi predanými hubami vyskytne nejaká jedovatá.

2.11.2 Matica zámien

V reálnom svete sa klasifikátorom zvyčajne nedarí dosahovať dokonalej presnosti klasifikácie. Jedným zo spôsobov, ako vyhodnotiť úspešnosť klasifikátoru, je použiť tzv. „maticu zámien“ (anglicky *confusion matrix*) [15]. Táto matica obsahuje v riadkoch jednotlivé triedy, do ktorých objekty naozaj patria a v stĺpcoch zasa triedy, ktoré klasifikoval klasifikátor. Pre dve triedy môže táto matica vyzerať nasledovne:

	Trieda1	**Trieda2**
Trieda1	75 %	25 %
Trieda2	25 %	75 %

Tabuľka 2.1: Príklad matice zámien

V ideálnom prípade obsahuje táto matica 100 % (príp. plný počet objektov, ktoré spadajú do danej triedy) na hlavnej diagonále a nuly mimo nej.

2.12 Typy klasifikátorov

2.12.1 Lineárny klasifikátor

Lineárny klasifikátor je klasifikátor, ktorý klasifikuje na základe hodnoty lineárnej kombinácie príznakového vektora. Medzi triedami sa snaží vytvoriť tzv. „separačnú líniu“, na základe ktorej potom klasifikuje. Separačná línia sa líši podľa počtu príznakov – pri jednom ňou je bod, pri dvoch priamka, pri troch rovina atď.

Lineárny klasifikátor sa dá matematicky zapísať ako

$$y(\mathrm{x}) = \mathrm{w}^T\mathrm{x} + w_0,$$

kde w^T je vektor váh, x je príznakový vektor a w_0 je nejaká konštanta. Klasifikátor potom vyberie jednu triedu v prípade, že $y(\mathrm{x}) > 0$, inak vyberie druhú.

Tento klasifikátor môže vracať pre objekty patriace do rovnakej triedy rôzne číselné hodnoty – „skóre“, ktoré v tomto príp reprezentuje vzdialenosť od rozhodovacej hranice. Existuje však aj tzv. „zobecnený lineárny klasifikátor“, ktorý má matematický zápis:

$$y(\mathrm{x}) = f(\mathrm{w}^T\mathrm{x} + w_0),$$

kde f sa nazýva „aktivačná funkcia“.

Táto podkapitola bola čerpaná z [2].

2.12.2 Perceptron

Perceptron je lineárny klasifikátor s aktivačnou funkciou napr. v tvare:

$$f(\mathrm{w}^T\mathrm{x} + w_0) = \begin{cases} 0 & \mathrm{w}^T\mathrm{x} + w_0 < 0 \\ 1 & \mathrm{w}^T\mathrm{x} + w_0 \geq 0 \end{cases}$$

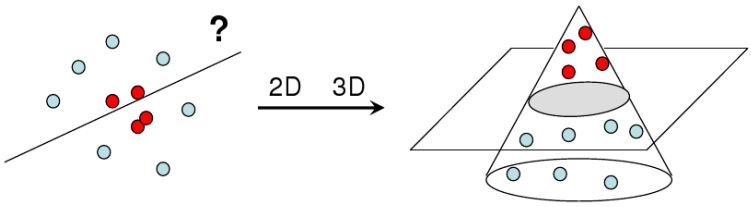
Perceptron poskytuje tzv. „tvrdé rozhodnutie“ o tom, do ktorej triedy daný objekt patrí. Rozhodovanie by síce viedlo k rovnakému výsledku ako pri lineárnom klasifikátore, ale napr. pre učiaci algoritmus bude vhodnejšie definovať si požadovaný výstup klasifikátora ako $\{0, 1\}$.

Táto podkapitola bola čerpaná z [20] a z [2]

2.12.3 Support vector machines

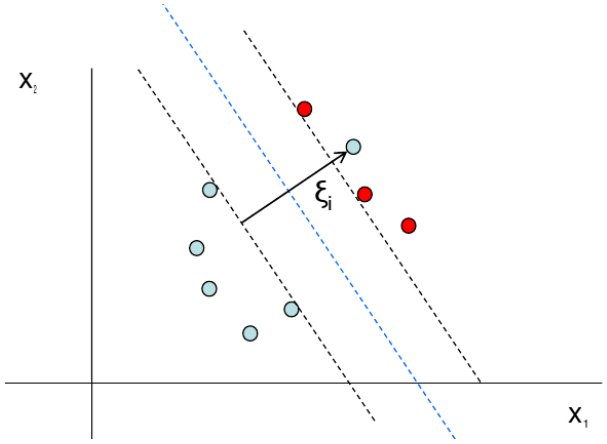
Suport vector machines (skrátene SVM) sú súborom metód učenia sa s učiteľom, ktoré dokážu reprezentovať dáta ako body v n-rozmernom priestore, ktoré sú rozmiestnené tak, aby ich bolo možné čo najlepšie rozdeliť hyper-rovinou. Táto hyper-rovina je umiestnená tak, aby jej vzdialenosť od každej triedy bola čo najväčšia. Je možné povedať, že SVM sú rovnaké ako perceptron, rozdiel je však v trénovacom algoritme [16].

V prípade lineárne separovateľných dát, keď sa podarí nájsť optimálnu oddeľujúcu hyper-rovinu, dáta ležiace na jej okraji sú označované ako „support vectors points“ a riešenie je reprezentované ako ich lineárna kombinácia. Ostatné dáta sú ignorované, čo znamená že rovnaké výsledky môžme dostávať aj s oveľa menšou vhodne zvolenou množinou trénovacích dát [11].



Obrázok 2.10: Mapovanie 2-rozmerného priestoru do 3-rozmerného pre lepšiu separáciu dát [16]

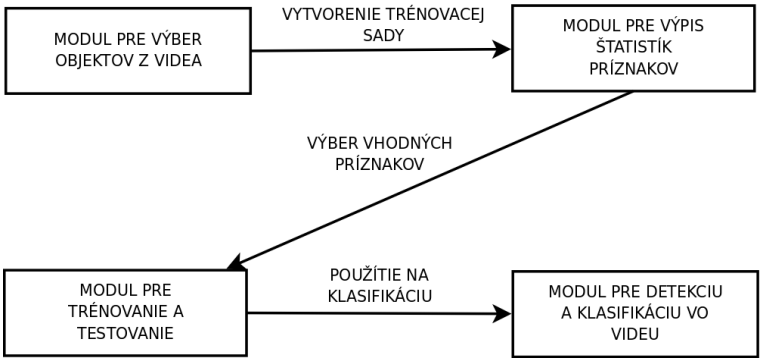
Často sa stáva, že trénovacie dáta nie sú separovateľné do tried, teda nie je možné ich rozdeliť hyper-rovinou. Jedným z možných riešení je umiestniť dáta do viac-rozmerného priestoru a vytvoriť separačnú hyper-rovinu v ňom, ako ukazuje obr. 2.10. Takýmto spôsobom je možné separovať akékoľvek trénovacie dáta. Lineárna separácia v tomto priestore je vlastne rovná nelineárnej separácii v pôvodnom priestore. Pri prekrývajúcich sa dátach sa naopak zavádzajú premenné (anglicky *slack variables* [16]), ktoré oslabujú obmedzujúce podmienky (viď obr. 2.11) .



Obrázok 2.11: Princíp fungovania slack variables [16]

Kapitola 3

Návrh



Obrázok 3.1: Návrh aplikácie a spôsob jej fungovania

Prvým dôležitým bodom pri návrhu aplikácie je výber programovacieho jazyka. Keďže zadanie nešpecifikuje žiadny konkrétny jazyk, zámerne bol vybraný jazyk C++, pretože aplikácia je veľmi vhodná na objektové spracovanie. Taktiež budeme pracovať s knižnicou OpenCV (konkrétne s jej verziou 2.0, napísanou v jazyku C[1]), ktorá poskytuje množstvo funkcií z oblasti spracovania obrazu a takisto aj funkcie na jednoduché zobrazenie obrazu alebo videa.

Aplikácia bude mať za úlohu detekovať a klasifikovať ciele vo videosignále, ktoré sa nachádzajú vo väčšej vzdialenosti. Na natrénovanie klasifikátora budeme potrebovať nejaké trénovacie dáta, z ktorých budeme extrahovať príznaky. Zvyčajne ale nie je jednoduché takéto dáta získať (hlavne pokiaľ sa jedná o zábery z termálnej kamery alebo noktovízoru). Preto bude aplikácia umožňovať, aby si užívateľ vybral z vstupného videa objekty, ktoré ho zaujímajú a tie si nechal uložiť. Keďže sa bude jednať o jednoduché obrázky, databáza nebude potrebná.

Po vytvorení trénovacej sady je zvyčajne vhodné zistiť, ktoré príznaky dobre oddeľujú jednotlivé triedy objektov. Z toho dôvodu bude vhodné, ak sa implementuje nejaký štatistický modul, ktorý dokáže vypísať charakteristické hodnoty jednotlivých príznakov pre danú triedu.

Ďalší modul sa bude starať o automatické natrénovanie klasifikátora. Zároveň bude

[1] V dobe začatia prác na aplikácii ešte nebola k dispozícii (momentálne) aktuálna verzia 2.1 a manuálové stránky k verzii knižnice implementovanej v jazyku C++ neboli kompletné.

poskytovať aj testovaciu časť, ktorá predloží klasifikátoru užívateľom vopred vytvorený zoznam obrázkov objektov a vypíše, koľko objektov klasifikátor zaradil do ktorej triedy.

Posledným modulom bude samozrejme modul, ktorý bude detekovať a klasifikovať objekty vo vstupnom videu na základe natrénovaného alebo užívateľom vytvoreného klasifikátora. Takisto bude aj umožňovať zápis výsledného videa s detekovanými a klasifikovanými cieľmi na disk.

Všetky moduly aj s ich vzájomným logickým prepojením sú zobrazené na obr. 3.1.

Aplikácia bude ovládaná pomocou spúšťacích parametrov, ktoré budú určovať modul, ktorý sa bude počas jej behu používať, a takisto aj názvy videosúborov, súborov s klasifikátormi a pod., ktoré budú nevyhnutné pre beh toho-ktorého modulu.

Kapitola 4

Získanie dát

Ešte predtým, ako sa dostaneme k samotnej implementácii, pozrieme sa bližšie na testovacie dáta (konkrétne videá), s ktorými budeme pracovať. Pomôže nám to tak lepšie pochopiť navrhnuté algoritmy a postupy použité pri implementácii.

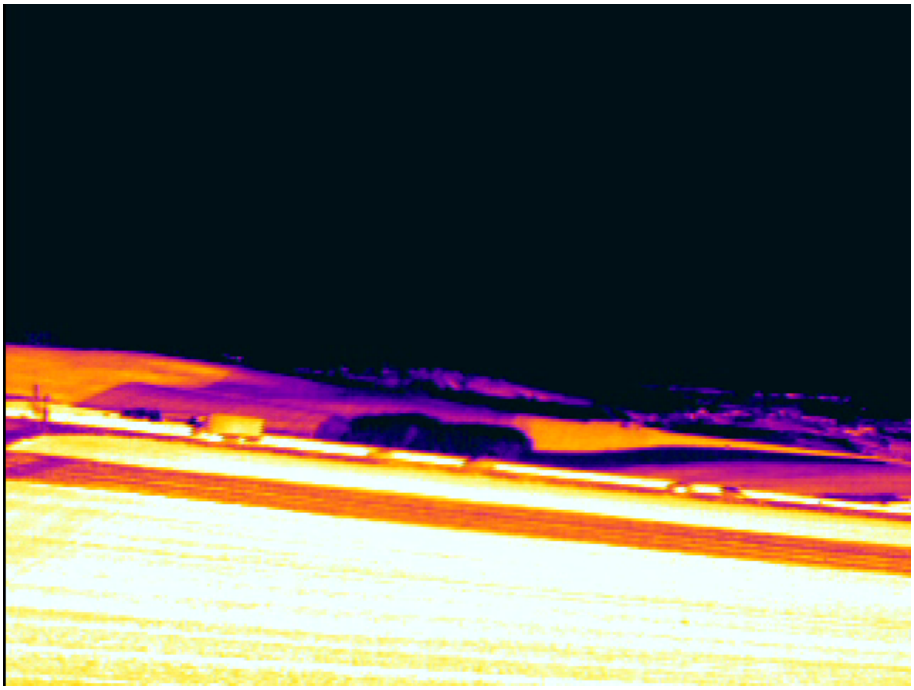
Obrázok 4.1: Žuráň – miesto, z ktorého boli snímané videá

Videá boli získané iba z termálnej kamery, noktovízor totiž nebol k dispozícii. Snímané boli z vrchu Žuráň a na záberoch sa nachádza blízka cesta 1. triedy (viď. obr. 4.1). Vzdialenosť miesta snímania od cesty je približne 400 metrov. Vlastnosti a nastavenia kamery sú uvedené v tabuľke 4.1.

Zorné pole	23°
Teplotný rozsah	17 – 32° C
Koeficient emisivity	1,0
Vlhkosť	50 %

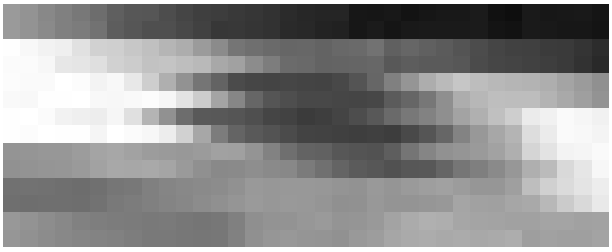
Tabuľka 4.1: Nastavenie kamery

Kamera bola statická a snímala dynamické objekty.



Obrázok 4.2: Snímok z testovacieho videa

Ako je možné vidieť na obr. 4.2, ciele vo videu (v tomto prípade jedno nákladné a štyri osobné vozidlá) sami o sebe neposkytujú mnoho informácii a sú pomerne ťažko rozpoznateľné. Priemerné rozmery osobného vozidla sú totiž 31 x 13 pixelov, u nákladných vozidiel to je 57 x 21 pixelov[1]. Pre lepšiu ilustráciu slúži obr. 4.3 reprezentujúci osobné vozidlo, ktorý bol pre lepšie rozpoznanie prevedený do úrovní šedej. Vozidlo sa pohybuje zľava – doprava.



Obrázok 4.3: Zväčšený výrez osobného vozidla aj s okolím

Pri cieľoch takýchto rozmerov je už značne náročné snažiť sa ich klasifikovať na základe nájdenia ich častí, ktoré ich jednoznačne identifikujú (v prípade vozidla to môžu byť napr. kolesá alebo okná). Z tohoto dôvodu bol zvolený iný prístup, o ktorom sa viac dočítame v kapitole 5.2.

[1]Tieto hodnoty boli získané zo štatistického modulu aplikácie a sú trochu zavádzajúce. Pod pojmom „rozmer“ sa tu totiž myslí veľkosť detekovanej oblasti, ktorá, ako si ukážeme neskôr (viď kapitolu 6.1.1), môže byť v závislosti od zvolenej metódy detekcie väčšia ako sú skutočné rozmery vozidla.

Kapitola 5

Implementácia

Implementácia sledovala návrh a ako výsledok bola vytvorená aplikácia v jazyku C++ s využitím knižnice OpenCV obsahujúca niekoľko modulov, ktoré boli ukázané na obr. 3.1. Táto aplikácia dokáže detekovať a klasifikovať osobné a nákladné vozidlá a ohraničiť ich vo videu obdĺžnikom vo farbe príslušiacej k danej triede vozidla. Je takisto možné vytvoriť si ľubovoľné množstvo nových klasifikátor a tie potom zahrnúť do aplikácie bez nutnosti zmeny zdrojového kódu.

V tejto kapitole si najprv popíšeme hlavné implementované algoritmy a následne pomocou nich vytvoríme vyššie zmienené moduly.

Všetky funkcie spomenuté v tejto kapitole začínajúce na `cv` patria do knižnice OpenCV a ich bližší popis aj s parametrami je možné nájsť napr. v online referenčnom manuáli [19].

5.1 Detekcia

Vzhľadom na skutočnosť, že sme snímali scénu statickou kamerou, bolo možné použiť algoritmy, ktoré detekujú pohyb v scéne podľa zmeny hodnoty pixelu v konkrétnom bode.

Na detekciu boli použité dva algoritmy, pričom oba detekujú cieľ na základe pohybu.

V prvom prípade ide o algoritmus na odčítavanie dvoch po sebe idúcich snímkov, ktorý bol popísaný v 2.8.1 a bol rozšírený o niekoľko ďalších krokov. Jedná sa o jednoduchý a pritom v konečnom dôsledku efektívny algoritmus, ktorý je v aplikácii nastavený ako primárny.

Druhý algoritmus vytvára isté vektory príznakov reprezentujúce popredie a pozadie a následne pomocou Bayesovského klasifikátora rozhoduje, ktoré časti obrazu budú klasifikované ako pozadie a ktoré ako popredie. Tento algoritmus je implementovaný v knižnici OpenCV a dostatočne zdokumentovaný v [13], preto si ho tu nebudeme podrobnejšie rozoberať.

5.1.1 Odčítavanie dvoch po sebe idúcich snímkov

Algoritmus pracuje s dvojicou snímkov – predchádzajúcim a aktuálnym. Oba snímky musia byť v odtieňoch šedej. Predchádzajúci snímok sa vždy berie ako model pozadia. Pred prvým cyklom algoritmu sa ako predchádzajúci aj aktuálny snímok použije jeden a ten istý snímok, teda v prvom cykle algoritmus nedetekuje nijaké objekty[1].

[1] V tejto práci sa rozlišujú pojmy „objekt“ a „cieľ“ – objekt je výsledkom detekcie a jediné, čo môžme o ňom povedať, je, že môže sa v obraze v mieste jeho detekcie niečo nachádza. Naopak cieľ je objekt, ktorý bol úspešne klasifikovaný.

Algoritmus postupuje nasledovne (v každom ďalšom kroku pracuje s výstupom kroku predchádzajúceho):

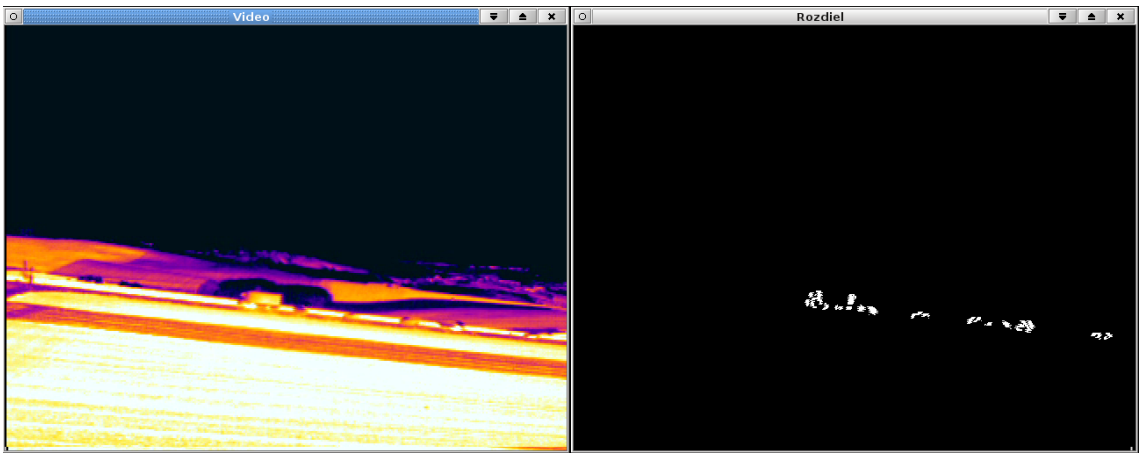
1. vyhlaď aktuálny snímok
2. odčítaj od seba predchádzajúci a aktuálny snímok
3. použi prahovanie
4. aplikuj dilatáciu
5. nájdi obrysy objektov a pre každý vypočítaj obdĺžnik, ktorý ho ohraničuje
6. zisti, či obdĺžnik spadá do požadovaného intervalu – ak nie, ignoruj objekt
7. vráť zoznam týchto obdĺžnikov

Vyhladenie snímku

Na vyhladenie snímku je použitá funkcia `cvSmooth()`. Je však potrebné si uvedomiť, že vyhladením obrazu sa zmenšia rozdiely na ostrých prechodoch, čo môže pri malých pohybujúcich sa objektoch spôsobiť, že nebudú detekované.

Odčítanie od predchádzajúceho snímku a prahovanie

Výsledok po odčítaní dvoch po sebe idúcich snímkov a následnom prahovaní výsledku vidíme na obr. 5.1. Pohyb je reprezentovaný bielymi pixelmi, statické pozadie čiernymi. Vidíme, že každé vozidlo je tvorené niekoľkými oddelenými regiónmi. Na odčítanie dvoch snímkov a prevedenie výsledku do absolútnej hodnoty bola použitá funkcia `cvAbsDiff()`.

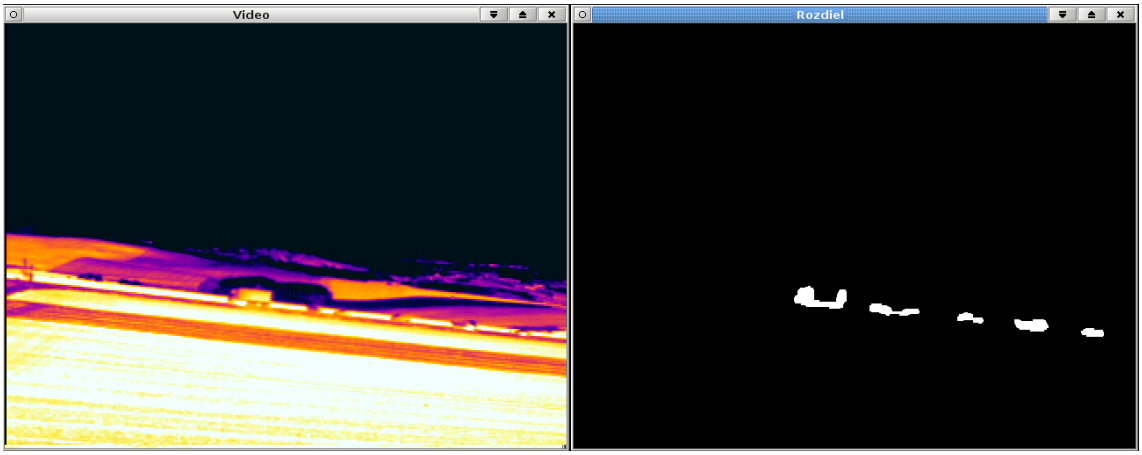


Obrázok 5.1: Výsledok odčítania dvoch po sebe idúcich snímkov a následného prahovania

Dilatácia

Na spojenie oddelených regiónov patriacich jednému objektu bola použitá dilatácia a následne erózia[2], pretože dilatácia spôsobuje zväčšenie jasných plôch, čo pri cieľoch takých malých rozmerov, s akými pracujeme, môže viesť k značnej deformácii informácie o veľkosti. Často ale jedna iterácia dilatácie a erózie nestačia, preto je možné si zvoliť ich počet. Výsledný obraz po dilatácii si môžme pozrieť na obr. 5.2.

Na dilatáciu bola použitá funkcia `cvDilate()`, na eróziu funkcia `cvErode()`.



Obrázok 5.2: Aplikácia dilatácie

Nájdenie obrysov a ich ohraničenie

Po aplikácii dilatácie a erózie môžme vyhľadať v snímku jednotlivé pohybujúce sa objekty a ohraničiť ich. Na nájdenie obrysov existuje v knižnici OpenCV funkcia `cvFindContous()` (podrobnejšie informácie je možné nájsť v [3], príp. taktiež v [19]). Táto funkcia vracia zoznam nájdených obrysov. Pre každý nájdený obrys sa vypočíta obdĺžnik (ktorého strany sú rovnobežné so stranami snímku) pomocou funkcie `cvBoundingRect()`.

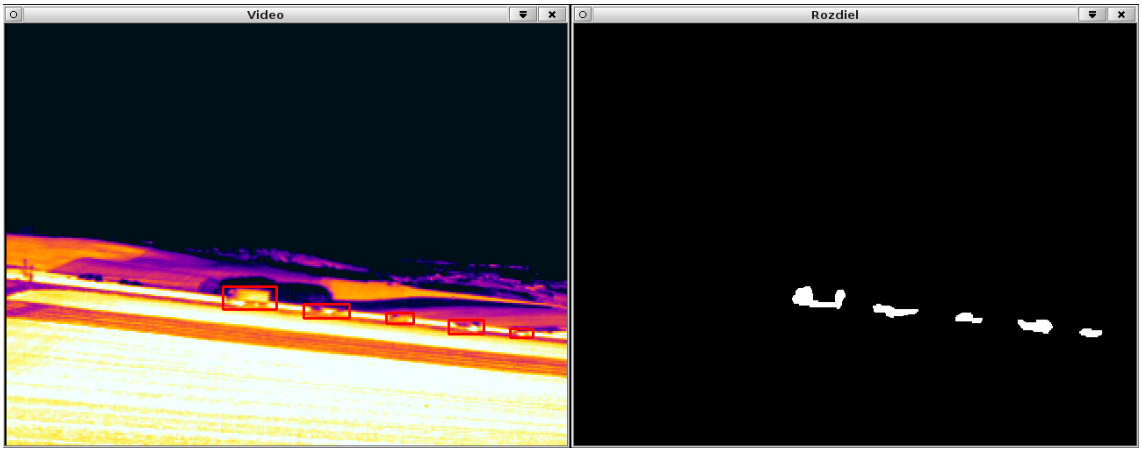
Filtrovanie objektov podľa veľkosti

Algoritmus umožňuje nastaviť minimálnu a maximálnu šírku a výšku obdĺžnika, ktorý ešte bude považovať za objekt. Vďaka tomuto jednoduchému opatreniu je možné odfiltrovať napr. občasný šum (reprezentovaný zvyčajne zmenou iba niekoľkých pixelov na rôznych miestach snímku) ako aj objekty, ktoré sú mimo našu oblasť záujmu a ktorých veľkosť v obraze dokážeme jednoznačne určiť.

Vrátenie zoznamu detekovaných obdĺžnikov

Pokiaľ objekt prejde filtrom, informácia o obdĺžniku, ktorý ho ohraničuje, je uložená do zoznamu, ktorý predstavuje návratovú hodnotu algoritmu. Vďaka tomuto zoznamu je potom možné detekované objekty napr. ohraničiť obdĺžnikom priamo v snímku (ako to ukazuje obr. 5.3) alebo ho ďalej predať klasifikátoru, ktorý priamo určí, o aký objekt sa jedná.

[2]Empirickým pozorovaním bol počet iterácii erózie nastavený na hodnotu o jedna menšiu ako je počet dilatácií



Obrázok 5.3: Vykreslenie obdĺžnikov okolo detekovaných objektov

5.2 Klasifikácia

Na klasifikáciu boli vybrané dva typy klasifikátorov – support vector machines (najmä kvôli ich schopnosti byť natrénované aj na malom objeme dát [16]) a množina lineárnych klasifikátorov s rôznymi váhami. Support vector machines (ďalej už len SVM) je možné trénovať automaticky, pri množine lineárnych klasifikátorov musí užívateľ sám správne odhadnúť jednotlivé hranice príznakov a ich váhy.

5.2.1 Postup pri klasifikácii

Najprv bola pomocou modulu pre výber objektov z videa vytvorená trénovacia sada cieľov, ktoré boli rozdelené na dve triedy – osobné a nákladné vozidlá. Následne boli zvolené príznaky, pomocou ktorých sa natrénovali SVM a takisto sa aj zvolili rozdeľovacie línie u množiny lineárnych klasifikátorov. Nakoniec boli tieto klasifikátory vytvorené a použité.

5.2.2 Zvolené príznaky

Na klasifikáciu bolo vybraných niekoľko príznakov. Prihliadalo sa na to, že detekované objekty majú malé rozlíšenie, ako aj na to, že farebný profil objektov sa môže v závislosti na podmienkach a nastaveniach meniť (napr. na obr. 4.2 sa javí, že pole, ktoré je v spodnej časti obrázku, je najteplejšou oblasťou na snímku, aj keď v skutočnosti iba reflektuje teplo dopadajúce naň z iného zdroja – v tomto prípade Slnka). Vytvoriť za týchto podmienok nejaký klasifikátor by bolo náročné, preto sa zvolila extrakcia príznakov z binárneho obrazu, ktorý vzniká v rámci detekcie. V tomto obraze máme totiž pomerne presne ohraničené miesta pohybu, ktoré nám udávajú tvar objektu.

Vzhľadom na riešený problém (a možnosť prípadného rozšírenia o ďalšie triedy objektov v budúcnosti) bolo zvolených sedem rôznych príznakov.

- rozmery objektu
- plocha, ktorú zaberá
- obvod objektu

- pomer výšky a šírky
- pomer plochy opísanej elipsy a plochy objektu
- pomer plochy opísaného kruhu a plochy objektu
- stĺpcová reprezentácia objektu

Prvé tri príznaky síce nie sú invariantné voči rotácii alebo zmene merítka, zohrávajú ale dôležitú úlohu za podmienky, že sa ciele pohybujú z jednej strany obrazu na druhú bez približovania sa, vzďaľovania sa alebo otáčania sa, ktorá je v prípade testovacích videí splnená.

Pomer výšky a šírky objektu dosahuje pri rozpoznávaní osobných a nákladných vozidiel približne rovnaké hodnoty, avšak pokiaľ by v budúcnosti bola vznesená požiadavka aj na rozpoznávanie ďalších tried objektov, ako sú napr. osoby, pôjde o dôležitý príznak, na základe ktorého sa tieto budú môcť diferencovať napr. od vozidiel.

Posledné tri príznaky sú invariantné voči zmene veľkosti a pri použití vhodného algoritmu na nájdenie orientácie objektu aj invariantné voči rotácii.

Bližšie sa pozrieme na výpočet stĺpcovej reprezentácie objektu, ktorá prebieha nasledovne:

1. zmeň veľkosť binárneho obrazu objektu na vopred dohodnutú (je možné ju nastaviť)
2. pre každý stĺpec v novovytvorenom obraze vypočítaj, koľko pixelov v ňom zaberá objekt (biele body)
3. vráť zoznam týchto hodnôt (t.j. počet vrátených hodnôt bude rovnaký ako šírka obrazu)

Tento príznak vlastne vzorkuje objekt po jeho výške. Keďže majú po prvom kroku algoritmu všetky objekty jednotnú veľkosť, dokážeme pomocou neho určiť rozdiely v tvaroch objektov.

Predávanie príznakov

Nie je nutné získavať príznaky po jednom, ale je možné priamo požiadať o predanie všetkých príznakov naraz ako jeden vektor hodnôt typu `float` (zvolený typ bude vysvetlený v kapitole 5.3). Vďaka tomu je možné do zdrojového kódu kedykoľvek pridať extrakciu ďalších príznakov, ktoré sa iba následne uložia do tohoto vektora, bez nutnosti meniť ostatné časti kódu.

5.3 Trénovanie SVM

Knižnica OpenCV poskytuje priamo triedy a metódy (táto časť knižnice je najnovšia a je napísaná v C++) na vytvorenie a trénovanie rôznych typov klasifikátorov [19]. My si na tomto mieste podrobnejšie ukážeme, ako natrénovať SVM.

SVM sú v OpenCV zastúpené triedou `CvSVM`, ktorá ponúka metódy na trénovanie, klasifikáciu a uloženie príp. načítanie vytvoreného modelu.

Na trénovanie sú potrebné dve matice – trénovacia matica a matica s príslušnosťami.

Trénovacia matica je dvojrozmerná matica, ktorá je typu `CV_32F`, čo je 32-bitová matica float čísel, čo je aj dôvod, prečo sú všetky príznaky predávané ako vektor hodnôt typu `float`.

Táto matica obsahuje v stĺpcoch príznaky príslušiace jednému konkrétnemu vzoru, ktorý bol vopred (zvyčajne manuálne) zaradený do určitej triedy. Teda pre každý jeden vzor existuje jeden riadok (ako to ukazuje tabuľka 5.1), z čoho vyplýva, že matica má rozmery *počet vzorov* * *počet príznakov*.

príznak 0 vzoru 0	príznak 1 vzoru 0	...	príznak n vzoru 0
príznak 0 vzoru 1	príznak 1 vzoru 1	...	príznak n vzoru 1
...	...	...	...
príznak 0 vzoru m	príznak 1 vzoru m	...	príznak n vzoru m

Tabuľka 5.1: Trénovacia matica

Matica s príslušnosťami je jednorozmerná riadková matica, ktorá obsahuje číslo triedy, do ktorej konkrétny objekt patrí. Jej veľkosť je teda určená počtom vzorov.

Implementovaný trénovací algoritmus pracuje s dvoma kategóriami vzorov – pozitívnymi a negatívnymi. Každý vzor je reprezentovaný svojim termálnym a binárnym obrázkom (segmentovaným z jeho pohybu). Tieto vzory sú predané vo forme textových súborov, ktoré obsahujú názvy jednotlivých obrázkov vzorov uložených na disku. Algoritmus potom pracuje nasledovným spôsobom:

1. z textových súborov vyextrahuj názvy obrázkov uložených na disku
2. načítaj tieto obrázky
3. pre každý pozitívny vzor:
   (a) vyextrahuj príznaky
   (b) naplň nimi jeden riadok trénovacej matice
   (c) do matice príslušností vlož `1`
4. pre každý negatívny vzor:
   (a) vyextrahuj príznaky
   (b) naplň nimi jeden riadok trénovacej matice
   (c) do matice príslušností vlož `0` (takto sme vlastne vytvorili binárny klasifikátor, ktorý vracia `1`, pokiaľ objekt do triedy patrí, inak vracia `0`)
5. obe matice predaj do metódy `train()` triedy `CvSVM`
6. vytvorený model ulož na disk pomocou metódy `save()` ako XML súbor

5.4 Princíp fungovania množiny lineárnych klasifikátorov

Pojem „množina lineárnych klasifikátorov“ môže byť čiastočne mätúci, preto si ho vysvetlíme podrobnejšie.

Príznakový vektor sa rozdelí sa jednotlivé príznaky a každý príznak sa predá jednému páru lineárných klasifikátorov, z ktorých každý má nastavenú inú separačnú líniu. Pre to, aby bol príznak klasifikovaný ako príznak patriaci do danej triedy, musia oba tieto klasifikátory potvrdiť jeho príslušnosť k triede. Zjednodušene povedané, pár klasifikátorov

tvorí akýsi „interval“, do ktorého musí príznak padnúť. Pokiaľ doňho padne, klasifikátor vracia `1`, inak vracia `0`.

Výsledné hodnotenie dostaneme tak, že sčítame váhované hodnotenia jednotlivých párov a prevedieme ich do intervalu $\{0, 1\}$. Pokiaľ je výsledná hodnota vyššia ako vopred zvolená konštanta, objekt do danej triedy patrí.

Tieto klasifikátory sa netrénujú automaticky, ale je nutné im nastaviť limity a váhy pre jednotlive príznaky, ktoré sa nastavujú v textovom súbore s príponou `.mfc`.

5.5 Ďalšie podporné algoritmy

5.5.1 Jednoduché sledovanie

Pri klasifikácii je vhodné mať informáciu o tom, že aktuálne detekovaný objekt je ten istý ako ten, ktorý bol v predchádzajúcom snímku nejakým spôsobom klasifikovaný ako cieľ. Kvôli tomu bola implementovaná jednoduchá sledovacia metóda.

Každý už raz klasifikovaný cieľ obsahuje snímok zo svojim obrazom z poslednej klasifikácie. Keďže sa jedná o termálny obraz, môžme predpokladať, že teplota objektu sa medzi jednotlivými snímkami nemení vôbec alebo iba minimálne. Z toho vyplýva, že si môžme z predchádzajúceho a aktuálneho snímku vozidla vytvoriť tepelné histogramy, ktoré navzájom porovnáme.

Aby sa čo najviac znížil počet porovnávaní histogramov, aplikovali sme ešte dve doplňujúce podmienky. Novodetekovaný objekt môže byť cieľom iba v prípade, keď:

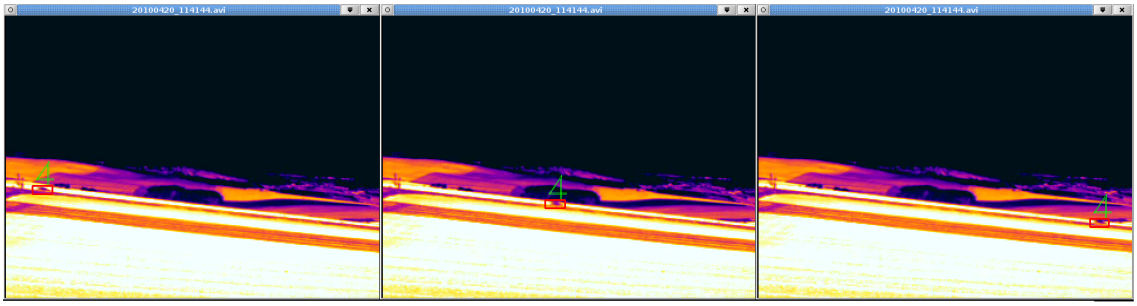
- sa objekt a cieľ nelíšia vo veľkosti; maximálny rozdiel veľkostí je nastavený konštantou
- sa obdĺžniky okolo objektu a cieľa prekrývajú. Vychádzali sme tu z pozorovania, že pohybujúci sa objekt zvyčajne nemá dostatočnú rýchlosť (vzhľadom na jeho rozmery), aby sa pri bežnej rýchlosti snímania kamery 30 snímkov za sekundu posunul o celú svoju dĺžku príp. výšku.

Tento algoritmus, ktorý dokáže poprepájať už raz klasifikované ciele s novodetekovanými objektami, vyzerá nasledovne:

1. získaj pozície a rozmery obdĺžnikov cieľa a objektu
2. pokiaľ sa líšia veľkosťou alebo sa vzájomne neprekrývajú, signalizuj, že sa nejedná o ten istý objekt a skonči
3. zo snímku uloženého v cieli a snímku objektu vytvor tepelné histogramy
4. porovnaj tieto histogramy a pokiaľ sa líšia o menej ako stanovuje na to určená konštanta, preveď nasledujúce činnosti:
   - 4.1 aktualizuj polohu cieľa
   - 4.2 aktualizuj snímok cieľa
   - 4.3 nastav cieľ na „detekovaný“

Pre ukázanie funkčnosti algoritmu bolo každému klasifikovanému cieľu pridelené jedinečné číslo, ktoré bolo následne nad ním vykreslené. Ako vidíme na obr. 6.1, algoritmus funguje spoľahlivo, pokiaľ nedochádza k žiadnemu prekrytiu cieľa. Pokiaľ k prekrytiu dôjde, zvyčajne stratíme informáciu o oboch prekrytých objektoch. Tento problém nebol súčasťou

riešenia tejto práce, avšak v zdrojovom kóde sú pripravené niektoré algoritmy (ako je napr. výpočet aktuálnej rýchlosti cieľa), ktoré by v budúcnosti mohli uľahčiť náhradu aktuálneho sledovacieho algoritmu nejakým robustnejším.



Obrázok 5.4: Jednoduché sledovanie pomocou porovnávania tepelných histogramov

5.5.2 História klasifikácií

Aj dobre navrhnutý klasifikátor sa môže niekedy pomýliť a zaradiť objekt do triedy, kam v skutočnosti nepatrí, len kvôli tomu, že na aktuálnom snímku vplyvom rôznych okolností (napr. šumom) spadajú jeho príznaky viac do inej triedy. Na elimináciu takýchto „občasných“ chybných rozhodnutí nám slúži práve história klasifikácií.

Na začiatok by sme si ešte mali uviesť, že história klasifikácií na svoje fungovanie potrebuje sledovací algoritmus (táto skutočnosť bola aj hlavným dôvodom, prečo bol sledovací algoritmus implementovaný). Jej princíp je potom jednoduchý.

V skutočnosti sa jedná o kruhový zoznam, ktorý umožňuje vložiť aktuálnu triedu, do ktorej bol objekt zaradený a naopak vrátiť hodnotu najčastejšie sa vyskytujúcej triedy. Na začiatku je zoznam inicializovaný na hodnotu `ŽIADNA_TRIEDA`, určenú konštantou. Iba v prípade, že všetky prvky zoznamu obsahujú túto hodnotu, je vrátená ako najčastejšie sa vyskytujúca.

História klasifikácií sa aktívne využíva až pri opätovnej klasifikácii cieľa. Pri prvom výskyte totiž stačí iba nastaviť triedu určenú klasifikátorom a tú pridať do histórie.

Klasifikácia s týmto zoznamom potom vyzerá takto:

1. znovu klasifikuj cieľ a triedu, pre ktorú si sa rozhodol, vlož do kruhového zoznamu
2. zisti najčastejšie sa vyskytujúcu triedu
3. ďalej pracuj s touto triedou

5.5.3 Mazanie cieľov

Pokiaľ sa nám v aktuálnom snímku nepodarí znovu detekovať všetky ciele klasifikované v predchádzajúcom snímku, mohli by sme tieto nedetekované zmazať. Tým by sme ale prišli o ich históriu klasifikácií a v prípade ich opätovného výskytu by sme museli históriu napĺňať znovu. Preto bolo ku každému cieľu pridané počítadlo, koľko snímkov ubehlo od jeho poslednej detekcie a klasifikácie. Až keď cieľ nebol klasifikovaný niekoľko snímkov po sebe (počet snímkov je možné nastaviť parametrom), vymaže sa. V prípade opätovnej detekcie sa počítadlo vynuluje.

5.5.4 Zápis videa na disk

Zápis na disk sa skladá z troch krokov:

1. vytvorenie štruktúry typu `cvVideoWriter()` starajúcej sa o zápis a inicializácia zápisu– na to slúži funkcia `cvCreateVideoWriter`
2. postupné ukladanie jednotlivých snímkov na disk zabezpečuje funkcia `cvWriteFrame()`
3. po ukončení zápisu sa zavolá funkcia `cvReleaseVideoWriter()`

5.5.5 Práca so súbormi

Na prácu so súbormi sa používajú štandardné nástroje dostupné v knižnici `fstream`.

5.5.6 Načítanie vstupných parametrov

Celá aplikácia rozpoznáva až 26 parametrov, preto bola na ich analýzu použitá funkcia `getopt_long()`. Nie všetky parametre je nutné zadávať, sú totiž nastavované automaticky. Po načítaní všetkých vstupných parametrov prebehne kontrola, či boli zadané všetky parametre potrebné pre chod zvoleného modulu. Zoznam týchto parametrov je možné nájsť v prílohe C.

5.6 Použité dátové typy a funkcie z knižnice OpenCV

Predtým, ako si uvedieme jednotlivé moduly, predstavíme si niekoľko dátových typov a funkcií z knižnice OpenCV, ktoré boli často využívané pri implementácii, vďaka čomu získame lepšiu predstavu o komponentoch, z ktorých je každý modul zložený. Bližšie informácie spolu s popisom parametrov a návratových hodnôt je možné nájsť v [3] alebo [19]. Zoznam všetkých použitých dátových typov a tried z knižnice OpenCV je uvedený v tabuľke 5.2.

5.6.1 Použité funkcie

Funkcie pracujúce s CvMat

- `cvCreateMat()` alokuje miesto pre novú maticu
- `cvReleaseMat()` uvoľní vytvorenú maticu
- `CV_MAT_ELEM()` je makro na prístup k prvkom matice

Funkcie pracujúce s IplImage

- `cvCreateImage()` alokuje miesto pre nový obrázok– je možné si zvoliť počet kanálov a farebnú hĺbku
- `cvLoadImage()` načíta obrázok z disku
- `cvCvtColor()` dokáže prevádzať obrázok medzi rôznymi farebnými modelmi; tu slúži na prevod do odtieňov šedej
- `cvResize()` mení veľkosť obrázku

Názov	Stručný popis	Využitie v aplikácii
CvMat	základná štruktúra pre matice, môže nadobúdať mnoho číselných typov	trénovanie SVM, klasifikácia pomocou SVM
IplImage	uchováva obraz, je možné si nastaviť počet kanálov aj farebnú hĺbku	všetky snímky
CvPoint	reprezentuje 1 bod so súradnicami x a y; často sa používa na označenie pozície v obraze	pomocné výpočty
CvSize	uchováva veľkosť, obsahuje *width* a *height*	filter pre detekciu, rozmery objektu
CvRect	synonymum pre obdĺžnik, ktorý je definovaný ľavým dolným rohom a výškou a šírkou	pozícia objektu v snímku
CvHistogram	obsahuje histogram, nad ktorým je možné robiť rôzne operácie, ako je porovnávanie, normalizácia a pod.	jednoduché sledovanie cieľa
CvCapture	štruktúra obsahujúca všetky potrebné údaje o videu, či už z kamery alebo zo súboru	inicializácia videosúboru a načítavanie snímkov
CvSVM	trieda zabezpečujúca vytvorenie, natrénovanie a použitie SVM	klasifikácia

Tabuľka 5.2: Použité štruktúry z knižnice OpenCV

- `cvRectange()` vykreslí do obrázku obdĺžnik
- `cvCopyImage()` skopíruje obrázok do druhého, pre ktorý už ale bolo alokované miesto pomocou funkcie `cvCreateImage()`

Funkcie na zobrazenie obrazu

- `cvNamedWindow()` vytvorí okno, do ktorého sa zobrazujú obrázky
- `cvShowImage()` zobrazí obrázok v okne; pokiaľ okno ešte nebolo vytvorené, vytvorí sa automaticky
- `cvDestroyWindow()` uvoľní sa vytvorené okno
- `cvWaitKey()` zastaví beh programu a pokračuje až po určitom počte milisekúnd

Funkcie pracujúce s CvHistogram

- `cvCreateHistogram()` vytvorí histogram

- `cvCalcHistogram()` vypočíta farebný histogram z obrázku
- `cvNormalizeHist()` prevedie jednotlivé hodnoty histogramu do určitého rozsahu
- `cvCompareHist()` porovná dva histogramy a vráti, na koľko sú si podobné

Funkcie pracujúce s CvCapture

- `cvCaptureFromCam()` načíta video z kamery; podpora priamo pre video z kamery je síce implementovaná, je však v zdrojovom kóde momentálne zakázaná
- `cvCaptureFromFile()` načíta video zo súboru
- `cvGetCaptureProperty` umožňuje získať vlastnosti videa ako sú napr. rozmery a počet snímkov za sekundu
- `cvReleaseCapture()` uvoľní video

Metódy pracujúce s triedou CvSVM

- `train()` natrénuje klasifikátor
- `save()` uloží natrénovaný model na disk
- `load()` načíta natrénovaný model
- `predict()` podľa príznakového vektora klasifikuje daný objekt do triedy

5.7 Moduly

Po oboznámení sa zo základnými algoritmami a aj funkciami a dátovými typmi použitými z OpenCV si môžme konečne bližšie priblížiť jednotlivé moduly a ich činnosť.

5.7.1 Modul pre výber objektov z videa

Tento modul používa detekčné algoritmy na detekovanie pohybu v scéne, ktorý po stlačení určitej klávesy bude označený a užívateľovi bude ponúknutá možnosť vybrať si vhodné objekty a uložiť na disk ako ich výrez z videa, tak aj ich segmentovaný obraz. O každej dvojici je spravený záznam do novovytvorených dvoch textových súborov (jeden obsahuje všetky výrezy videa a druhý segmentované obrazy), čo uľahčuje prácu modulu pre výpis štatistík a modulu na trénovanie – namiesto názvov jednotlivých obrázkov predávame iba dva textové súbory pre každú takto vytvorenú sadu.

5.7.2 Modul pre výpis štatistík

Modul najprv načíta obrázky z disku, ktorých názvy sú uložené v textových súboroch, a potom pre každú dvojicu výrez z videa – segmentovaný obraz spustí extrakciu príznakov. Po extrahovaní príznakov zo všetkých obrázkov uloží zoznam týchto príznakov do jednotlivých textových súborov – do každého zapíše zoznam hodnôt, ktoré jeden konkrétny príznak nadobúdal. Vďaka tomu môžme potom hodnotiť, ako veľmi sa triedy v rámci jedného príznaku prekrývajú.

5.7.3 Modul pre trénovanie

Tento modul trénuje iba SVM. Na svoju činnosť potrebuje pozitívne a negatívne príklady. Jeho činnosť bola podrobne popísaná v kapitole 5.3.

5.7.4 Modul pre testovanie

Využíva extrakciu príznakov, klasifikačný algoritmus (ktorý závisí od nastavenia vstupného parametru) a zoznam dvojíc obrázkov spomenutých v kapitole 5.7.1. Pre každú zadanú dvojicu príznakov vypíše na štandardný výstup meno súboru a odpoveď, či patrí do danej triedy, ktorú chceme testovať.

Vďaka tomuto modulu si môžme otestovať, ako náš klasifikátor zvláda prácu na trénovacích dátach ešte predtým, ako ho nasadíme na reálne videá.

5.7.5 Modul na detekciu a klasifikáciu

Dostávame sa k hlavnému modulu, slúžiacemu na detekciu a klasifikáciu objektov. Tento modul využíva väčšinu doteraz zmienených algoritmov a jeho činnosť prebieha nasledovne:

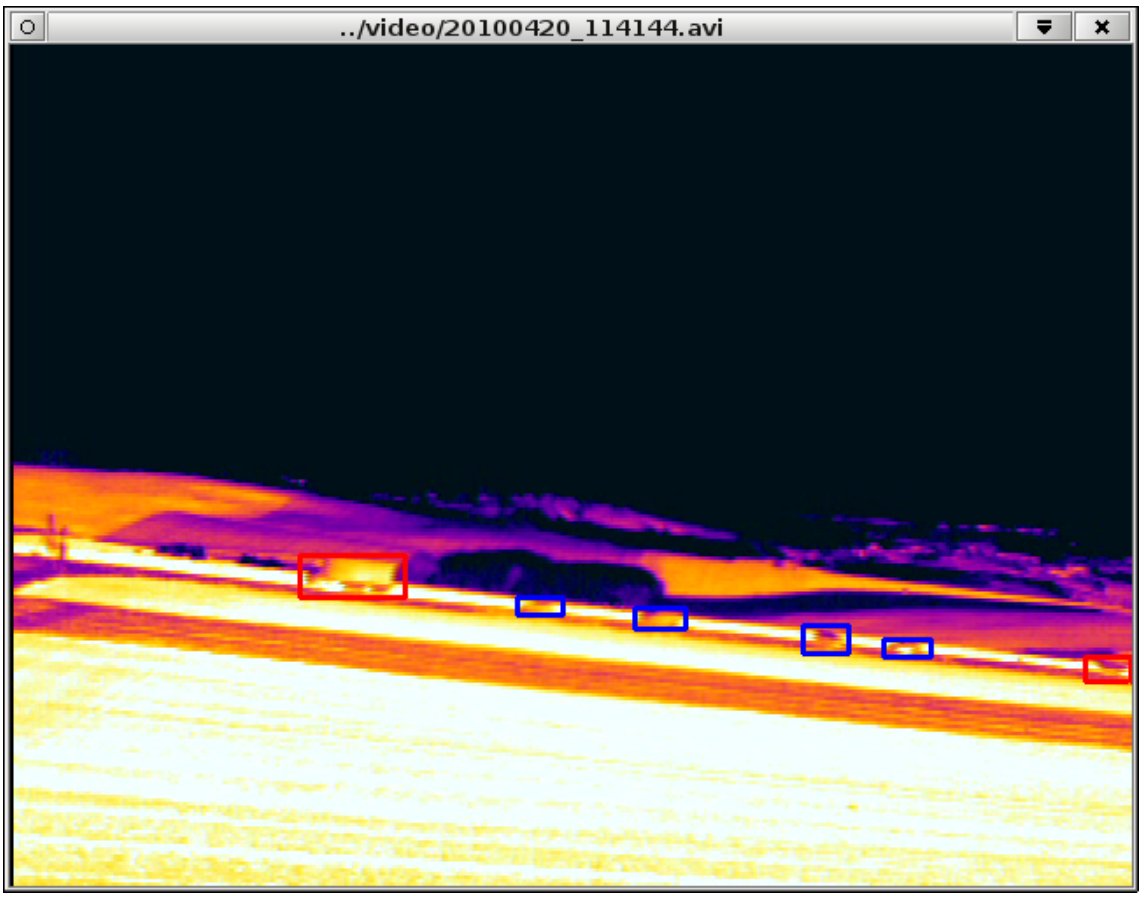
1. podľa toho, čo nastavil užívateľ, zvoľ detekčnú a klasifikačnú metódu, načítaj modely tried, farby a príp. inicializuj zápis videa na disk
2. načítaj prvý snímok z videa a zisti z neho jeho rozlíšenie a počet snímkov za sekundu
3. prvý snímok nastav ako pozadie pre detekčnú metódu
4. pokiaľ si ešte nedošiel na koniec videa, opakuj:
   - 4.1 začni merať čas
   - 4.2 načítaj ďalší snímok
   - 4.3 detekuj pohyb v snímku a získaj zoznam všetkých pohybujúcich sa objektov
   - 4.4 pospájaj objekty detekované v aktuálnom snímku s cieľmi klasifikovanými v predchádzajúcich cykloch
   - 4.5 zmaž všetky dlho nedetekované ciele
   - 4.6 klasifikuj všetky novodetekované objekty; vráť číslo klasifikátora, ktorý hlásil zhodu, inak vráť `-1`
   - 4.7 nakresli obdĺžnik okolo všetkých cieľov detekovaných a klasifikovaných v tejto iterácii cyklu
   - 4.8 zobraz obrázok v okne
   - 4.9 ak bol zvolený zápis, zapíš aktuálny snímok do súboru
   - 4.10 ukonči meranie času a vypočítaj, ako dlho máš čakať pred ďalšou iteráciou cyklu

Ako môžme vidieť, algoritmus načítava prvý snímok mimo cyklu, aby sa mohol naučiť pozadie a zistiť rozmery obrazu, na základe ktorých môže alokovať potrebné obrazové štruktúry. Takisto v každom cykle meria, ako dlho trval, vďaka čomu môže prispôsobiť čakanie medzi jednotlivými iteráciami a tým aj zobrazovaním snímkov.

Modely tried sú uložené v zozname, ktorý sa postupne prechádza. Za pozornosť stojí, že v prípade viacerých tried, medzi ktorými má klasifikátor klasifikovať, v prípade SVM vyberá

prvú, do ktorej daný objekt môže spadať (spôsobené tým, že predikcia SVM vracia `0`, pokiaľ objekt do triedy nepatrí, inak `1`), zatiaľ čo v prípade súboru lineárnych klasifikátorov vyberie v prípade zhody hodnôt číselného skóre viacerých tried prvú, ktorá takéto skóre dosiahla. Toto nám umožňuje nastaviť prioritnú triedu detekcie tým, že ju uvedieme ako prvú. Je potrebné si ešte uvedomiť, že klasifikačný algoritmus vracia index tej triedy v zozname, do ktorej objekt patrí. V prípade jednej triedy teda bude vracať `0`, pokiaľ do nej objekt patrí a `-1`, pokiaľ do nej nepatrí.

Výsledok tohoto algoritmu môžme vidieť na obr. 5.5. Osobné vozidlá sú orámované modrým a nákladné červeným obdĺžnikom.



Obrázok 5.5: Detekcia a klasifikácia cieľov, ako ju prevádza aplikácia

Kapitola 6

Testovanie

6.1 Detekcia

Testovala sa na videách s rozlíšením 640 x 480 pixelov, ktoré mali 25 snímkov za sekundu. Ukážka oboch algoritmov je na obr. 6.1.

Obrázok 6.1: Porovnanie detekčných algoritmov

6.1.1 Odčítavanie snímkov

Hlavnou výhodou tohoto algoritmu je jeho rýchlosť – priemerná doba trvania jednej iterácie bola 4 ms. Pri detekcii rýchlo sa pohybujúcich cieľov sa nevyskytli žiadne problémy – dokázal detekovať všetky ciele.

Problémy začali nastávať pri nákladných vozidlách, najmä kamiónoch (ako môžme vidieť na obr. 6.1 v strede), pretože ich prívesy mali zvyčajne len malý rozdiel teplôt, teda na rozdielovom snímku neboli detekované celé, ale iba ich hrany (záviselo to samozrejme od rýchlosti pohybu – pomaly sa pohybujúce kamióny sa detekovali horšie). Vzhľadom na malé rozmery kamiónov to viedlo k občasnej detekcii dvoch objektov namiesto jedného.

Ďalšou nevýhodou je, že táto metóda detekuje cieľ väčší, ako je v skutočnosti, pretože berie do úvahy aj polohu objektu na predchádzajúcom snímku, zatiaľ čo na aktuálnom sa tam už nachádza skutočné pozadie. Tento problém sa dá jednoducho vyriešiť odčítavaním od prvého snímku (scéna by ale musela byť tepelne stabilná, čo nie je príliš reálny predpoklad) alebo by sa aktuálny snímok použil ako obraz pozadia iba v prípade, ak by v ňom nebol detekovaný nijaký objekt.

Keďže sme ale detekovali pohybujúce sa vozidlá, ktoré sa pohybovali dostatočne rýchlo, tento algoritmus dokázal vo väčšine prípadov detekovať pohyb správne.

Značná výhoda oproti druhému algoritmu je, že práve vďaka odčítavaniu dvoch po sebe idúcich snímkov je tento algoritmus značne odolný voči občasnému traseniu kamerou (napr. vplyvom silného vetra) – dočasne síce môže detekovať pohyb aj v miestach, kde sa v skutočnosti nevyskytuje, no po zastavení trasenia sa dokáže veľmi rýchlo (vlastne do doby dvoch ďalších snímkov) vrátiť do vyrovnaného stavu.

6.1.2 Detekcia popredia a pozadia pomocou Bayesovského klasifikátora

Na rozdiel od predchádzajúceho, tento algoritmus dokáže veľmi dobre vyseparovať tvar vozidla, ako nám to aj ukazuje obr. 6.1 vpravo, vďaka čomu veľkosť vyseparovanej časti odpovedá skutočnej veľkosti vozidla.

Hlavným problémom tohoto algoritmu je priemerná doba trvania jeho jednej iterácie – 120 ms, kvôli čomu je možné ho použiť iba pri záberoch, ktoré majú malý počet snímkov za sekundu, príp. vždy ignorovať niekoľko snímkov alebo ho nasadiť na miestach, kde nepotrebujeme spracovanie v reálnom čase. Riešením tiež môže byť zmenšenie rozmerov obrazu, čo je ale v prípade malých objektov diskutabilné.

Tento algoritmus taktiež nie je odolný voči traseniu záberu a po prerušení trasenia ešte zvyčajne niekoľko sekúnd detekuje pohyb aj v miestach, kde sa nenachádza. Veľmi pomaly pohybujúci sa objekt sa môže stať súčasťou scény a potom, ako z nej zmizne, je v tejto oblasti stále hlásený pohyb.

Záver z tohoto algoritmu je teda nasledovný– pri použití na záberoch s nízkym rozlíšením (pri rozlíšení 320 x 240 trvala jedna iterácia približne 25 ms) alebo v situáciách, kde nie je potrebné spracovanie v reálnom čase a objekty sa pohybujú dostatočne rýchlo, sa jedná o veľmi dobrý algoritmus.

6.2 Klasifikácia

Klasifikácia prebiehala v dvoch krokoch – na trénovacej sade obrázkov a na videozáberoch. Trénovaciu sadu tvorilo dohromady 135 obrázkov osobných a 61 obrázkov nákladných automobilov.

6.2.1 SVM

SVM sa podarilo správne identifikovať všetky trénovacie a testovacie obrázky až na jeden, pri ktorom detekovalo osobné vozidlo ako nákladné. Veľmi dobrá klasifikácia je spôsobená vlastnosťou tohoto algoritmu učiť sa z trénovacích dát a byť vždy schopný nájsť hyper-rovinu, ktorá oddeľuje dve triedy dát. Výsledky klasifikácie sú uvedené v tabuľke 6.1 pomocou matice zámien.

V aplikácií tento algoritmus tiež funguje väčšinou spoľahlivo. Na základe toho, ktorý model bol načítaný ako prvý, dokážeme nastaviť jeho vyššiu prioritu (bližšie popísané v kapitole 5.7.5).

	Osobné vozidlo	**Nákladné vozidlo**
Osobné vozidlo	134	1
Nákladné vozidlo	0	61

Tabuľka 6.1: Výsledky testovania SVM

6.2.2 Sada lineárnych klasifikátorov

Sada lineárnych klasifikátorov poskytuje pri testovaní o trochu horšie výsledky pri klasifikácií nákladných vozidiel (viď tabuľku 6.2), tie sú ale spôsobené manuálnym nastavením a ich prenastavením by bolo možné výsledky zlepšiť. V samotnej aplikácií potom funguje približne rovnako ako SVM.

	Osobné vozidlo	**Nákladné vozidlo**
Osobné vozidlo	135	0
Nákladné vozidlo	5	56

Tabuľka 6.2: Výsledky testovania sady lineárnych klasifikátorov

6.2.3 Zhrnutie klasifikačných algoritmov

Oba klasifikátory dokázali vo veľkej časti prípadov klasifikovať objekty správne. Klasifikáciu zlepšila aj implementácia histórie klasifikácií a s ňou spojeného jednoduchého sledovania, ktorá značne znížila počet chybných klasifikácií a s nimi spojených „preblikávaní“ triedy cieľa v obraze. Bolo by ale potrebné vyskúšať ďalšie typy príznakov, ako je napr. kladenie väčšieho dôrazu na tvar, ktorý pri objektoch s malými rozmermi zohráva významnú rolu.

Kapitola 7

Záver

V tejto práci sa podarilo navrhnúť a implementovať algoritmy na detekciu a klasifikáciu vzdialených cieľov snímaných termálnou kamerou. Boli implementované a otestované dva algoritmy na detekciu – jednoduché odčítavanie dvoch po sebe idúcich snímkov a komplexnejší algoritmus, založený na vytváraní vektorov príznakov reprezentujúcich popredie a pozadie a ďalej následnej klasifikácii pomocou Bayesovského klasifikátora. Tento pojednáva o tom, ktorá časť scény patrí do popredia a ktorá do pozadia.

Ďalej boli implementované a otestované dva klasifikátory – support vector machines a sada lineárnych klasifikátorov, rovnako ako boli navrhnuté možné zmeny potrebné pre nasadenie do reálnej prevádzky.

Taktiež bolo navrhnutých a implementovaných niekoľko podporných algoritmov, ako jednoduché sledovanie alebo história klasifikácií, ktoré pomohli ešte viac stabilizovať výsledky klasifikácie.

Pre ďalšie zdokonalenie práce by bolo vhodné vyskúšať navrhované zmeny v algoritmoch, ako aj navrhnúť a implementovať nové detekčné a klasifikačné metódy a porovnať ich s aktuálnymi, príp. sa pokúsiť extrahovať nové typy príznakov. Taktiež, vzhľadom na množstvo možných parametrov, by bolo dobré dodať aplikácií intuitívne užívateľské rozhranie.

Nahradenie aktuálneho sledovacieho algoritmu nejakým komplexnejším a robustnejším riešením, ktoré by dokázalo aj predikovať ďalší výskyt objektu, by mohlo vďaka už implementovanej histórii klasifikácií zefektívniť a spresniť prácu klasifikátora a zároveň aj eliminovať existujúci problém prekrývania sa objektov.

Poslednou, avšak nemenej zaujímavou možnosťou by určite bolo sledovať tú istú scénu pomocou dvojice (prípadne viacerých) kamier, pričom každá by snímala scénu v inom spektre a príp. aj rozlíšení. Potom by sa detekcia a klasifikácia mohli uskutočňovať na základe kombinácie príznakov z jednotlivých kamier, čo by mohlo posunúť celú túto problematiku do kvalitatívne významne vyššej úrovne, kde by sme mohli očakávať výstupy odolné voči vyššie spomínaným rušivým vplyvom, a teda už oveľa lepšie využiteľné v praxi. Práve preto by táto práca by mohla poslúžiť ako vhodný základ pre ďalšie zdokonalenie detekcie a klasifikácie vojenských cieľov v budúcnosti.

Literatúra

[1] Klasifikace a rozpoznávaní: Extrakce příznaků [online]. 2009-03-09 [cit. 2010-05-10]. URL https://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/03_extrakce%20priznaku/IKR3.pdf

[2] Klasifikace a rozpoznávaní: Lineární klasifikátory [online]. 2009-04-06 [cit. 2010-05-10]. URL http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/05_lin_klasifikatory/IKR5.pdf

[3] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, první vydání, 2008, ISBN 978-0-596-51613-0.

[4] FLIR: Thermal Imaging: how far can you see with it? [online]. 2008 [cit. 2010-05-11]. URL http://www.flir.com/uploadedFiles/ENG_01_howfar.pdf

[5] FLIR System: *ThermaCam^{TM}Researcher Professional Edition*. FLIR System, 2006, Publ. No. 1 558 071.

[6] Halliday, D.; Resnick, R.; Walker, J.: *Fyzika: Elektromagnetické vlny — Optika — Relativita*. Nakladatelství VUTIUM, první vydání, 2000, ISBN 80-214-1868-0.

[7] Holst, G. C.: *Electro - optical imaging system performance*. JCD Publishing, páté vydání, 2008, ISBN 978-0-819-47406-3.

[8] Horák, K.: Dynamické obrazy [online]. 2008 [cit. 2010-05-10]. URL http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/12-Dynamickeobrazy.pdf

[9] HurleyIR: Thermal Imagers: Frequently asked questions [online]. 2007 [cit. 2010-05-11]. URL http://www.hurleyir.com/faqs.html

[10] ITT: Night Vision & Imaging: Image Gallery [online]. 2009 [cit. 2010-05-11]. URL http://www.nightvision.com/gallery/law_enforcement.html#img/law_1.jpg

[11] Kotsiantis, S. B.: Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, , č. 31, 2007: s. 249–268.

[12] Kršek, P.; Španěl, M.: Redukce barevného prostoru [online]. 2007 [cit. 2010-05-10]. URL https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_omezeni_barev_print.pdf

[13] Li, L.; Huang, W.; Gu, I. Y. H.; aj.: Foreground object detection from videos containing complex background. *Proceedings of the eleventh ACM international conference on Multimedia*, 2003: s. 2–10, ISBN:1-58113-722-2.

[14] Láník, A.; Zuzaňák, J.: Extrakce obrazových příznaků [online]. 2009-03-12 [cit. 2010-05-09].
URL http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/04_obrazove_priznaky/IKR4.pdf

[15] Malik, F.: *Extrakce informací z hypertextu*. Diplomová práce, Masarykova Univerzita - Fakulta informatiky, 2006/2007.

[16] Schwarz, P.: Klasifikace a rozpoznávaní: Umělé neuronové sítě a Support Vector Machines [online]. 2009-03-19 [cit. 2010-05-09].
URL http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/06_nn_svm/nn_svm3.ppt

[17] Toyama, K.; Krumm, J.; Brumitt, B.; aj.: Principles and practice of background maintenance. *Proceedings of the 7th IEEE International Conference on Computer Vision*, 1999: s. 255–261.

[18] WWW stránky: Welcome - OpenCV Wiki [online]. 2010-04-06 [cit. 2010-05-09].
URL http://opencv.willowgarage.com/wiki/

[19] WWW stránky: OpenCV 2.1 C Reference [online]. 2010-[cit. 2010-05-13].
URL http://opencv.willowgarage.com/documentation/c/index.html

[20] Španěl, M.: Počítačové vidění [online]. 2007 [cit. 2010-05-16].
URL https://www.fit.vutbr.cz/study/courses/POV/private/lectures/pov_02_statisticke_rozpoznavani.pdf

[21] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Compuret Press, 2004, ISBN 80-251-0454-0.

Dodatok A

Obsah CD

Na priloženom CD sa nachádzajú tieto súbory a zložky:

- `bin/` – zložka, v ktorej sa po preložení nachádzajú binárne súbory s aplikáciou, ďalej modely tried pre klasifikátory, súbor s farbami a súbor `priklady.txt`, ktorý ukazuje niekoľko možností spustenia aplikácie s rôznymi parametrami
- `examples/` – zložka s vytvorenými ukážkovými videami, ktorá obsahuje ukážky fungovania klasifikácie pomocou SVM, ako aj množiny lineárnych klasifikátorov
- `latex/` – zložka obsahujúca zdrojový kód bakalárskej práce v LaTeXe
- `samples/` – zložka, ktorá obsahuje niekoľko už vytvorených trénovacích sád
- `src/` – zložka so zdrojovými súbormi aplikácie
- `video/` – zložka s videami, na ktorých je možné testovať aplikáciu
- `projekt.pdf` – text tejto práce
- `README.TXT` – textový súbor s podrobnejším popisom obsahu CD a návodom na inštaláciu
- `run.sh` – bashovský skript, ktorý za predpokladu, že sú splnené všetky podmienky uvedené v `README.TXT` skompiluje aplikáciu a presunie ju do zložky `src/`

Na CD sa ešte v zložke `bin/` nachádzajú dva súbory – `colors.clrs`, ktorý obsahuje na každom riadku trojicu hodnôt RGB pre nastavenie farby klasifikovaného cieľa. Každému modelu triedy klasifikátora bude potom z tohoto súboru postupne priradený jeden riadok s farbou – prvému klasifikátoru prvý riadok, druhému druhý atď.

Druhý súbor je `trucks.mfs`, ktorý obsahuje nastavenia pre množinu lineárnych klasifikátorov. Riadky v ňom sú komentované, preto nie je nutné sa o nich podrobne rozpisovať. Za zmienku ale stojí poradie hodnôt na riadku. Prvá hodnota určuje minimálnu hranicu príznaku, druhá maximálnu a tretia určuje váhu, ktorú bude tento príznak mať za predpokladu, že jeho hodnota bude v danom intervale.

Dodatok B

Abecedný zoznam tried

Táto časť slúži na stručné zorientovanie sa v zoznam tried a ich činnosti. Viac informácií je možné nájsť v zdrojovom kóde, ktorý je dostatočne komentovaný.

- Classifier – obsahuje oba typy klasifikátorov
- ClassifierSettings – úložisko nastavení pre súbor lineárnych klasifikátorov
- CyclicList – implementácia histórie aktualizácií
- FeaturexExtractor – extrahovanie jednotlivých príznakov príp. vracanie celého príznakového vektora
- HistogramComparison – vytvorenie a porovnávanie histogramov
- ImageDifferencing – odčítanie snímkov
- ImageTransforms – všetky potrebné operácie s obrazom, ako je dilatácia, erózia, zmena veľkosti a pod.
- OpenCVDetection – implementovaný algoritmus na detekciu pohybu z knižnice OpenCV
- Settings – všetky nastavenia aplikácie
- Stats – modul pre výpis štatistík
- Target – trieda reprezentujúca cieľ
- TargetsList – tieda reprezentujúca zoznam cieľov aj s metódami na spájanie aktuálneho a predchádzajúceho výskytu
- TextFileOperations – operácie s textovými súbormi, ako je načítanie a zápis
- Trainer – modul na trénovanie a testovanie klasifikátorov
- Video – trieda starajúca sa o načítavanie videa
- VideosProcessing – modul na detekciu a klasifikáciu
- Writer – modul na vytvorenie trénovacej sady

Dodatok C

Zoznam parametrov pre moduly aplikácie

Na tomto mieste si ukážeme zoznam parametrov, ktoré jednotlivé moduly akceptujú. Povinné parametre majú pred svojim memon hviezdičku.

C.1 Modul pre výber objektov z videa

*`--create-samples`	výber modulu
`--dilation-cycles <n>`	počet cyklov dilatácie
`--blur-kernel-size <n>`	veľkosť masky vyhladzovania
`--training-set-name <name>`	názov obrázkov a textových súborov pre daný výber
`--border <n>`	koľko pixelov okolo objektu chceme vyrezať
*`video_name`	názov videa uloženého na disku

Počas behu aplikácie fungujú nasledovné skratky:

`<medzerník>`	pozastaví video, ponúkne vybrať a uložiť objekty
`<esc>`	ukončí beh aplikácie
`klávesa <t>`	pokiaľ vo výbere, uloží objekt na disk ako cieľ
`klávesa <i>`	pokiaľ vo výbere, ignoruje objekt
`klávesa <f>`	pokiaľ vo výbere, otočí objekt okolo y-novej osy

C.2 Modul pre výpis štatistík

*--stats	výber modulu
*--pos-pictures <file_name>	textový súbor s umiestnením vyrezaných obrázkov cieľov
*--pos-segm-pictures <file_name>	textový súbor s umiestnením segmentovaných obrázkov cieľov
--resized-width <n>	šírka resiznutého obrázku (v pixeloch)
--resized-height <n>	výška resiznutého obrázku (v pixeloch)

C.3 Modul pre trénovanie

*--train	výber modulu
*--pos-pictures <file_name>	textový súbor s umiestnením vyrezaných pozitívnych obrázkov cieľov
*--pos-segm-pictures <file_name>	textový súbor s umiestnením segmentovaných pozitívnych obrázkov cieľov
*--neg-pictures <file_name>	textový súbor s umiestnením vyrezaných negatívnych obrázkov cieľov
*--neg-segm-pictures <file_name>	textový súbor s umiestnením segmentovaných negatívnych obrázkov cieľov
--resized-width <n>	šírka resiznutého obrázku (v pixeloch)
--resized-height <n>	výška resiznutého obrázku (v pixeloch)
--class-name <name>	meno triedy a modelu, ktorý sa uloží na disk

C.4 Modul pre testovanie

*--test	výber modulu
*--pos-pictures <file_name>	textový súbor s umiestnením vyrezaných obrázkov cieľov
*--pos-segm-pictures <file_name>	textový súbor s umiestnením segmentovaných obrázkov cieľov
--resized-width <n>	šírka resiznutého obrázku (v pixeloch)
--resized-height <n>	výška resiznutého obrázku (v pixeloch)
-s	použije sa SVM klasifikátor
-m	použije sa súbor lineárnych klasifikátorov
SVM_files	súbory s modelmi pre SVM, majú príponu .xml
linear_classifier_files	súbory s modelmi pre sadu lineárnych klasifikátorov, majú príponu .mfc

C.5 Modul pre detekciu a klasifikáciu

-i	detekcia pomocou odčítavania snímkov
-b	detekcia pomocou OpenCV algoritmu
-s	použije sa SVM klasifikátor
-m	použije sa súbor lineárnych klasifikátorov
-o <file_name>	uloží prehrávané video do súboru file_name
--delete-target-after <n>	počet snímkov, ktoré treba čakať, aby bol nedetekovaný cieľ úplne vymazaný
--type-history-length <n>	počet klasifikácii, ktoré budú uložené v histórii klasifikácií cieľa
--resized-width <n>	šírka resiznutého obrázku (v pixeloch)
--resized-height <n>	výška resiznutého obrázku (v pixeloch)
--dilation-cycles <n>	počet cyklov dilatácie
--blur-kernel-size <n>	veľkosť masky vyhladzovania
SVM_files	súbory s modelmi pre SVM, majú príponu .xml
linear_classifier_files	súbory s modelmi pre sadu lineárnych klasifikátorov, majú príponu .mfc
*video_name	názov videa uloženého na disku