TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

SYSTÉM PRO AUTOMATICKOU DETEKCI A ROZPOZNÁVÁNÍ POHYBUJÍCÍCH SE VOZIDEL

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Pavel Rada**
Vedoucí práce: doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

SYSTEM FOR AUTOMATIC DETECTION AND RECOGNITION OF MOVING VEHICLES

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Pavel Rada**
Supervisor: doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel Rada**
Osobní číslo:	**M12000219**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro automatickou detekci a rozpoznávání pohybujících se vozidel**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou zpracování a rozpoznávání obrazu.
2. Vytvořte databázi video nahrávek pohybujících se vozidel.
3. Navrhněte a realizujte systém v MS VS C# pro automatickou detekci vozidel. Tento systém by měl umožňovat rozpoznávat druhy jednotlivých vozidel a jejich základní vlastnosti, např. barvu, přibližný typ atd. Dále by mělo být umožněno i změřit přibližnou rychlost vozidla.
4. Otestujte navržený systém na pořízené databázi a proveďte jeho vyhodnocení.

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40 - 50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Davies, E., R.: Machine Vision - Theory, Algorithms, Practicalities. Morgan Kaufmann Press. UK, 2005, ISBN 0-12-206093-8**

[2] **Hlaváč V., Sedláček M.: Zpracování signálu a obrazu, Skripta FEL ČVUT, Praha 2000, ISBN 80-01-02114-9**

[3] **Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. PWS Publishing, USA, 1999, ISBN 0-534-95393-X**

[4] **Nagel, C., Evjen, B., Glynn, J., Watson, K., Skinner, M.: C# 2008, Programujeme profesionálně, Computer Press, EAN:9788025124017, 2013**

Vedoucí diplomové práce:	**doc. Ing. Josef Chaloupka, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant diplomové práce:	**Ing. Karel Paleček** Ústav informačních technologií a elektroniky
Datum zadání diplomové práce:	**12. září 2013**
Termín odevzdání diplomové práce:	**16. května 2014**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Rád bych touto cestou vyjádřil poděkování svému vedoucímu práce doc. Ing. Josefu Chaloupkovi, Ph.D. za jeho cenné rady a trpělivost při vedení mé diplomové práce.
Dále bych rád poděkoval svému konzultantovi Ing. Karlu Palečkovi za jeho rady týkající se implementace klasifikačních algoritmů.

Abstrakt

Tato diplomová práce se zabývá problematikou detekce a klasifikace projíždějících vozidel založené na metodách a algoritmech pro zpracování a rozpoznávání obrazových dat. V první části je popsán algoritmus pro detekci pohybu v obraze. Algoritmus se skládá z několika částí. V prvním kroku je potřeba předzpracovat data ze záznamového zařízení, následuje substrakce pozadí, jejímž výstupem je model prostředí, na jehož základě dojde k detekci nestacionárních objektů v obraze. Substrakce pozadí je dělena dle metod, sloužící pro aktualizaci zmíněného modelu. U těchto metod jsou zkoumány jejich výhody a nevýhody.

Ve druhé části jsou popsány klasifikační algoritmy využívané ve výstupní aplikaci této práce. Jsou to algoritmy PCA (Principal Component Analysis) využívající metodu nejbližšího souseda k určení třídy neznámého obrazu, dále korelační filtr MACE (Minimum Average Correlation Energy) a také klasifikátor SVM (Support Vector Machine) jenž spolupracuje s metodou PCA pro substrakci příznaků. Jedná se v podstatě o vylepšení metody PCA, s tím rozdílem, že klasifikace je realizována klasifikátorem SVM, nikoliv metodou nejbližšího souseda.

V praktické části je navržen program implementující zmíněné algoritmy, jenž je otestován na vytvořené databázi videosekvencí. Výsledky rozpoznávání klasifikátorů a úspěšnost detekce pohybu v obraze je shrnuta v poslední části této práce.

Klíčová slova: detekce pohybu v obraze, klasifikace, lokální příznaky, algoritmus

Abstract

This paper deals with the detection and classification of passing vehicles based on the video recording. The first section describes the algorithm for detecting movement in the image. The algorithm is base of several parts. First we need to preprocess data from the recording device, followed by subtraction of background. Output is background model which is subtracted from current frame. In this manner are detected non-stationary objects in the image. Background subtraction is divided according to the methods used for the update of that model. These methods are investigated base on their advantages and disadvantages.

The second section describes the classification algorithms used in the output application of this work. These include algorithms PCA (Principal Component Analysis) using the nearest neighbor method to determine the class of an unknown image, also correlation filter MACE (Minimum Average Correlation Energy) and SVM (Support Vector Machine) classifier which cooperates with the PCA method for subtraction features. This is basically the enhancement PCA method with the difference that the classification is carried SVM classifier rather than using nearest neighbor.

In the practical part is a designed program which implements the aforementioned algorithms, tested on a video database. Recognition classifiers results and success of motion detection algorithms are summarized in the last section of this paper.

Keywords: motions detection in video sequence, classification, local feature, algorithm

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V inteligentním dopravním systému mohou data z dopravy pocházet z mnoha různých zdrojů, jako jsou detekční smyčky, ultrasonické sensory nebo kamery. Použití kamer, z nichž mnohé jsou již nainstalovány pro monitorování dopravní sítě, ve spojení s technikami počítačového vidění nabízejí atraktivní alternativu k senzorům. Kamerové systémy založeny na videozáznamu jsou mnohem sofistikovanější, protože obsah informace související s obrazovými sekvencemi umožnuje přesné sledování vozidel a klasifikaci. Snímače mají oproti tomu omezené možnosti a jsou i často nákladnější. Vhodný video-systém pro sledování dálniční dopravy musí být adaptivní na různé povětrnostní i světelné podmínky Nejčetnější problém pochází z vržených stínů od světel vozidel a za slunečného dne kdy stíny vždy doprovázejí jedoucí vozidla. Detekce vozidel v takovýchto podmínkách je důležitá, ale jedná se o netriviální problematiku. Po té, co je vozidlo bezpečně detekováno, může být dále zpracováno na základě jeho sémantických vlastností extrahovaných z obrazového záznamu. Všechna tato data se využívají zejména pro zjištění stavů silniční sítě a optimalizace procesu přepravy. Aby byla statistika využitelná i v dalším odvětví, mohou detekovaná vozidla procházet procesem klasifikace, který zařadí jednotlivá vozidla do definovaných tříd, tím vzniká obecný přehled o vytíženosti dálniční sítě. Klasifikace je realizována zejména algoritmy pro rozpoznávání vzorů.

V problematice rozpoznávání vzorů jsou často používány klasifikátory založené na učení vzorových dat, protože nemají žádné zřizovací modelové problémy. V této diplomové práci bude navrhnut systém algoritmů pro detekci a následnou klasifikaci projíždějících vozidel. Jeho prioritní funkcí bude zaznamenávat vozidla jedoucí v jednom ze směrů po dálnici a detekovaná vozidla klasifikovat. Klasifikace bude realizována pomocí moderních algoritmů. Mimo jiné by měl systém umět i určit rychlost projíždějících vozidel na základě známé vzdálenosti a znalosti parametrů videa a také určit barvu vozu. Navržený systém bude následně implementován a otestován na sadě videí. Všechny algoritmy budou zhodnoceny výsledným skórem určující jejich úspěšnost. V závěru jsou vyhodnoceny jednotlivé algoritmy a je navržena diskuse.

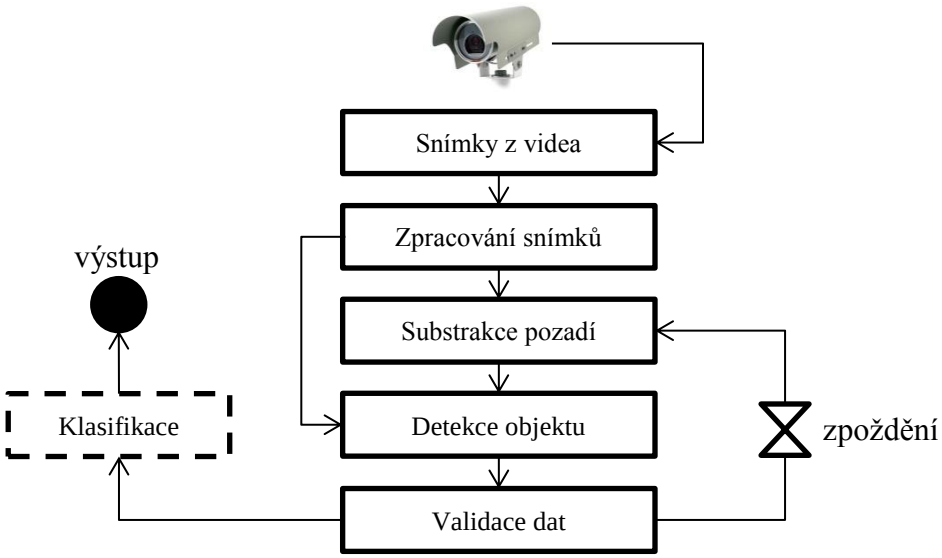
Kapitola 1

Předzpracování obrazových dat

Pro detekování a klasifikaci projíždějících vozidel je velmi důležitý krok předzpracování obrazových dat, který má významný vliv na budoucí úspěšnost klasifikačních algoritmů.

1.1 Obecný princip detekce vozidel

Cílem algoritmu, respektive soustavy algoritmů detekce pohybujících se objektů je detekovat významné změny vyskytující se v celé videosekvenci a zároveň odmítat změny indiferentní. Celý proces detekce je znázorněn na obrázku 1.



Obrázek 1: Diagram toku dat obeckého návrhu pro detekci pohyblivých objektů

Vstupem do obecného algoritmu je série snímků, které na sebe vzájemně navazují. Obecně platí přímá úměra mezi kvalitou zpracovávaného videa a spolehlivosti algoritmů, čili čím kvalitnější video do algoritmu vstupuje, tím jsou výsledky detekce a klasifikace lepší. Kvalita videa je zároveň kritický parametr určující rychlost zpracovávání jednotlivých snímků. Příliš kvalitní záznam s vysokým rozlišením nebude mít na spolehlivost algoritmů (klasifikaci) příliš velký vliv, ale rychlost algoritmů bude ovlivněna zpracováváním velkého množství obrazových informací, čímž bude ohrožena myšlenka zpracovávání videa takzvaně *real-time* neboli v reálném čase [1] Zpracování v reálném čase je značně závislé i na hardwaru, na kterém je systém provozován.

Předzpracování videa je tedy optimalizace respektive snaha o nalezení harmonického stavu mezi velikostí zpracovávaných snímků a kvalitou výstupu algoritmů. Obecně se jedná o podvzorkování snímků, snížení rozlišení videa nebo odstranění chyby vzniklé prokládáním snímku. Ta je nepřijatelná zejména u klasifikačních algoritmů založených na vzorech [2].

Základní princip detekce pohyblivého objektu je založen na existenci modelu prostředí reprezentující pozadí dané scény. Na základě tohoto modelu je možné ze scény vyhodnotit další informace, proto po základním zpracování snímků následuje proces substrakce pozadí vytvářející model prostředí. Navazující blok už zahrnuje samotnou detekci objektů v obraze. Výsledky operací na úrovni obrazových elementů jsou konvertovány na sémantické informace, které již lze zpracovávat na logické úrovni. Posledním blokem je validace dat určující význam daných objektů a také připravuje nalezené objekty k dalšímu zpracování, jako je jejich zařazení do definované třídy [3].

1.2 Odstranění chyby prokládání snímku

Tato chyba byla objevena u nahrávek pořízených za účelem testování algoritmů popisovaných v dalších kapitolách této práce. Jedná se o nežádoucí efekt prokládaného snímání moderních videokamer. Odchylku je nezbytné odstranit ještě před započetím zpracovávání videa, aby nedošlo k degradaci procesu detekce a klasifikace.

1.2.1 Prokládané snímání videokamer

CCD snímač nejprve generuje pole s lichými řádky a následně pole se sudými řádky, které kombinuje do jednoho snímku (*frame*). Celý tento proces je navržen pro snížení šířky pásma. Důvodem jsou jednoznačně nižší výrobní náklady snímacích zařízení, což zahrnuje nejen kamery, ale i televize či vysílací systémy. Prokládané snímání s pevně definovanou šířkou pásma poskytuje dvojnásobnou obnovovací frekvenci pro daný počet řádek, což zlepšuje vzhled snímaných objektů v pohybu. Nevýhodou je potencionální problém zvaný *Interline twitter* ve formě *moire*. Tento druh rušení se zobrazí pouze za podmínky, že subjekt obsahuje detail ve svislé ose, který se blíží horizontálnímu rozlišení formátu videa. Ve výsledku je sudý a lichý půlsnímek nesprávně skombinován a v obraze vznikají alternativní linky posunuty od sebe navzájem. Odstraněním alternativních linek se zabývá proces *deinterlace*. Chybu prokládání snímku je možné shlédnout na obrázku 2, který reprezentuje jedno z vozidel z testovacích nahrávek [8].

1.2.2 Odstranění alternativních linek

Existuje několik metod, jak odstranit problematiku prokládání snímků, z nichž každá produkuje evidentní problém. Většinu technik pro odstranění prokládání lze rozdělit do tří různých skupin. První skupinou jsou metody kombinující sudé a liché řádky existujících snímků. Druhá skupina rozdělí každý snímek zpět na dvě pole, z nichž jedno je rozšířeno na výšku celého snímku. Do poslední skupiny se řadí metody využívající technik obou výše zmíněných metod a řadí se tak mezi metody kompenzace pohybu. Moderní systémy pro odstranění chyby prokládáním využívají techniku jako je detekce hran ve snaze nalézt pohyb mezi poli. Toho se následně využívá pro interpolaci chybějících linek z původní oblasti. Následující výčet obsahuje používané opravné metody z každé skupiny:

1) ***Blending***: Patří do skupiny kombinačních metod a provádí se průměrováním po sobě následujících snímků. Vzniká takzvaný *ghosting* vlivem ztráty vertikálního a časového rozlišení.
2) ***Half-size***: Jedná se o metodu z druhé skupiny, která pracuje pouze s jednou polovinou snímku.
3) ***Movment detection***: Metoda spadá do třetí skupiny a vykazuje nejlepší výsledky při odstraňování alternativních linek. Jsou často kombinovány s detekcí změny scény. Jinak by mohlo dojít k mixtuře scén, které spolu nesouvisejí

Pro odstranění chyby prokládání snímku ve videích určených pro klasifikační algoritmy, bylo přistoupeno k metodám využívající detekci pohybu, protože zaručují nejlepší výsledek. Porovnání obrazu zatíženého touto chybou a jeho následné opravení je možné pozorovat na obrázku 2 [9].

Obrázek 2: vlevo obrázek zatížen chybou prokládání, vpravo opravený obrázek

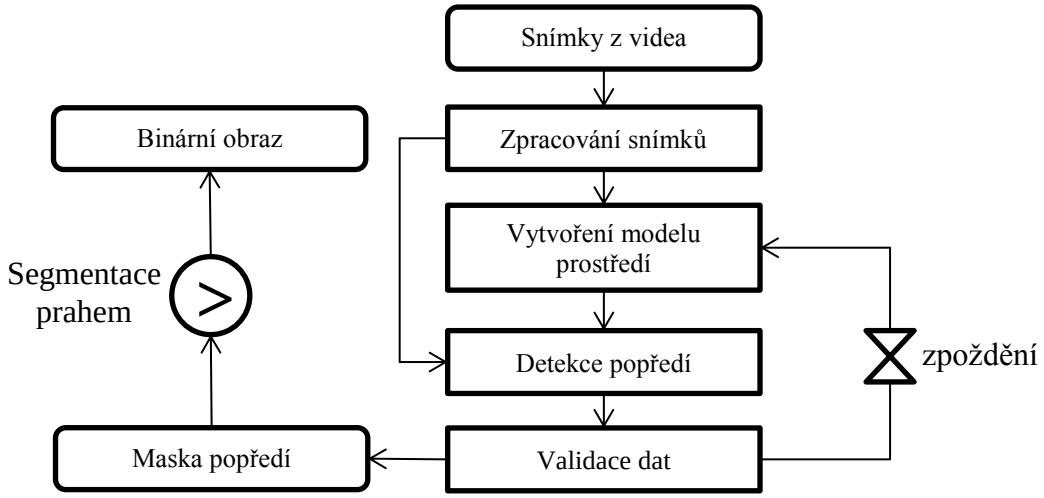
1.3 Substrakce pozadí (BGS)

Substrakce pozadí byla klíčovým prvkem při návrhu systému pro detekci pohybu v obraze. V literatuře je navrženo mnoho metod pro extrakci objektů, které mohou být rozděleny na automatické a poloautomatické. Automatické fungují bez jakéhokoliv lidského zásahu, zatímco poloautomatické vyžadují interakci s uživatelem. Běžný přístup k identifikaci pohybujících se objektů je pomocí techniky substrakce pozadí. Myšlenka substrakce pozadí spočívá v separaci pohyblivých objektů, které zůstanou jako elementy popředí získaných z obrazu díky rozdílu každého snímku a takzvaným **modelem prostředí** dané scény. Tento model se používá jako referenční obraz, který je porovnáván s každým nahraným snímkem. V důsledku toho musí být model přesnou reprezentací scény, ze které jsou odstraněny všechny nestacionární elementy. Dále je potřeba model prostředí neustále aktualizovat, aby byly zohledněny změny ve světelných podmínkách, nebo jakákoliv změna textury pozadí. Vše výše zmíněné by mělo být dosažitelné i přes to, že se prvky v popředí neustále pohybují různou rychlostí od začátku videosekvence. Mnoho algoritmů substrakce pozadí nacházející se v literatuře, nedokáže plně vyřešit problematiku pohybu v komplexním prostředí [10].

Každý dobrý algoritmus pro substrakci pozadí by měl splňovat následující charakteristiky. Měl by být schopný přizpůsobit se různým úrovním osvětlení a to v různých denních dobách. Dále je třeba zvládnout nepříznivý vliv počasí, jako je mlha nebo sníh, který modifikuje scénu. Nakonec by měl být schopen zpracovat pohyblivé objekty, které prvně přecházejí do popředí a po jisté době se stanou pozadím. Charakteristickým znakem metod pro substrakci pozadí je, jakým způsobem jsou aktualizovány jejich modely prostředí. Z tohoto hlediska je dělíme na dvě kategorie:

1) ***Rekurzivní techniky*:** Tento druh udržuje jednotný model prostředí, který je aktualizován s každým novým příchozím snímkem. Obecně jsou výpočetně účinné a mají minimální požadavky na paměť. Příkladem rekurzivních technik je *Running Gaussian Average* (RGA).
2) ***Nerekurzivní metody*:** Nerekurzivní metody udržují ve vyrovnávací paměti množinu předchozích snímků o velikosti N a odhadují model prostředí pouze na základě statických vlastností snímků. To způsobuje vyšší požadavky na paměť než u rekurzivních metod. Vzhledem k tomu, že mají přístup k N nejnovějším snímkům (k historii), jsou velmi adaptivní.

Většina substrakčních mechanismů funguje na principu, který je znázorněn na obrázku 3.



Obrázek 3: Princip algoritmu substrakce pozadí

Ve většině systémů počítačového vidění spočívá zpracování snímků při substrakci pozadí v jednoduchém časovém nebo prostorovém vyhlazování pro snížení šumu kamery. Vyhlazení lze použít i k redukci šumu okolního prostředí, které může být způsobeno například deštěm nebo sněhem zachyceným venkovní kamerou. Výstupem celého substrakčního mechanismu je model prostředí, jehož produkce bude vysvětlena v několika následujících odstavcích [11].

1.3.1 Metoda substrakce pozadí na základě průměrování snímku

Jedná se o nejjednodušší metodu vytvoření modelu prostředí využívající historii snímků uložených v zásobníku. Každý pixel výsledného modelu je roven aritmetickému průměru odpovídajících pixelů snímků ze zásobníku (1.1). Tato metoda je velmi citlivá na prahování.

$$background_{avg}(x, y) = \frac{1}{N} \sum_{i=1}^{N} frame(x, y, t - i) \tag{1.1}$$

Proměnné ve vzorci x a y určují pozici počítaného pixelu výsledného modelu a N označuje celkový počet snímků v zásobníku. Proměnná t určuje okamžik, ve kterém se algoritmus nahází a má vyjadřovat posun mezi snímky. Počet snímků by měl být volen tak, aby výsledný model prostředí odpovídal skutečné perspektivě pozadí natáčené scény

1.3.2 Metoda substrakce pozadí na základě aproximace mediánem

Tato metoda nalezne rozdíl intenzity aktuálního pixelu a mediánu hodnot některých nedávných pixelů. Využívá zásobník o velikosti N, kde N představuje počet předchozích snímků, kdy hodnoty pixelů jsou využívány pro výpočet hodnoty mediánu. Algoritmus je realizován bez odběru dílčích snímků pro vytvoření adekvátního modelu. Obecný způsob výpočtu mediánu přes celý zásobník snímků je definován rovnicí (1.2).

$$background_{med}(x,y) = median(X) = \begin{cases} X_{\frac{(N+1)}{2}}, & |X| = 2k, \quad k \epsilon N \\ \dfrac{X_{\frac{N}{2}} + X_{\left(\frac{N}{2}\right)+1}}{2}, & |X| = k, \quad k \epsilon N \end{cases} \tag{1.2}$$

Algoritmy substrakce pozadí pracují často pouze se snímky převedenými do úrovně šedi. V opačném případě by rovnice (1.2) platila pouze pro jeden z barevných kanálů snímku.

1.3.3 Metoda substrakce pozadí na základě klouzavého průměru

Tato metoda využívá Gaussovu funkci hustoty pravděpodobnosti pro vyhodnocení hodnoty intenzity obrazových bodů. Ta nalezne rozdíl intenzity aktuálního pixelu a kumulativního průměru předchozích hodnot. Metoda tedy udržuje kumulativní průměr μ_t z nedávných hodnot pixelů. Předpokládá se, že se hodnoty pozadí náhodně mění podle normálového rozdělení. To znamená, že pokud je hodnota pixelu uvnitř gaussovy křivky jedná se o pixel pozadí a pokud je hodnota mimo gaussovu křivku pak se jedná o hodnotu popředí. Pokud je daný pixel klasifikován, jako pozadí je střední hodnota přepočítána dle vzorce (1.3)

$$\mu_{t+1} = \mu_t \cdot I_t + (1 - \alpha) \cdot \mu_t \tag{1.3}$$

Kde α je časová konstanta stanovená na základě dynamiky scény a typicky se nastavuje na hodnotu 0,05; μ_t udává hodnotu předchozího průměru. Ze vzorce je patrné, že s každým pixelem, jenž nespadá do popředí, se přepočítává Gaussova křivka což má za následek úpravu výsledného modelu. Parametr α významnou měrou zasahuje do procesu generování nových hodnot pixelů, jelikož s ním lze určovat, jak rychle se nový objekt začlení do pozadí scény. Díky tomu i jisté anomálie, které by neměly vznikat, jako například objekt pohybující se nepřiměřeně nižší rychlostí oproti ostatním subjektům scény, lze touto metodou eliminovat.

1.3.4 Metoda substrakce pozadí na základě rozdílů snímků

Metoda rozdílu snímků je nejjednodušší technika pro odečtení pozadí bez zásobníku. Obraz pozadí, který neobsahuje žádné nestatické objekty, které jsou objektem zájmu, je vybrán jako referenční. Hodnota pixelu pro každou souřadnici (x, y) a popřípadě pro každý barevný kanál obrázku pozadí je odečtena od příslušné hodnoty pixelu vstupního obrazu. Je-li výsledná hodnota větší než určitá prahová hodnota pak je pixel klasifikován jako pixel popředí. V opačném případě se jedná o pixel pozadí.

1.3.5 Zhodnocení algoritmů

Vytvořením modelu prostředí se zabývá nespočet technik, a popis všech není z důvodu zaměření práce možný. Vybrány byly proto pouze metody, které jsou implementovány ve výstupní aplikaci. Jedná se o kompromis mezi kvalitou výstupního modelu a rychlosti zpracování.

Při návrhu algoritmu byla využita dvě videa, zachycující danou scénu v extrémních podmínkách jako je hustý déšť a mlha. Všechny algoritmy pracovaly s přednastaveným prahem $Ts = 15$. V obou případech vykazuje lepší výsledky metoda substrakce na základě mediánu před metodou rozdílu snímku a klouzavého průměru. Zásadní nedostatek metody rozdílu snímku je, že pokud objekt zájmu reprezentují pixely s rovnoměrnou intenzitou, pak se tyto pixely stávají součástí pozadí. Dalším problémem je, že objekty musí být neustále v pohybu. Pokud objekt zůstane nehybný po delší dobu než je perioda jednoho rámce, pak se stává součástí pozadí. Metoda substrakce pozadí na základě klouzavého průměru selhává zejména při hustém dešti, kde detekuje jednotlivé kapky jako pohyblivý objekt.

Hlavní výhodou metody rozdílu dvou snímků je jeho skromné výpočetní zatížení. Změnou prahu na $Ts = 30$ jsou dramaticky degradovány výsledky metody rozdílu snímků i klouzavého průměru, zatímco metoda mediánu stále vykazuje rozumné výsledky. Naopak snížením prahu pod úroveň implicitně dané hodnoty dojde ke zvýšení šumu v obraze na takovou úroveň, kde selhávají všechny metody. Lepší výsledky u metody mediánu jsou dány její pamětí, díky které dochází k plynulejší aktualizaci modelu prostředí. Metoda rozdílu snímků vykazovala při testech velmi malé schopnosti adaptace na změnu světelných podmínek a značnou závislost na nastavení prahové hodnoty. Výsledné rozpoznávací skóre bylo velmi nízké a tak bylo přistoupeno k vyjmutí implementace této metody z výstupní aplikace [12].

Posouzením výhod a nevýhod technik pro vytvoření modelu prostředí je uzavřena další část blokového diagramu algoritmu substrakce pozadí. Následující blok, je úzce zaměřen pouze na detekci popředí.

Detekce popředí identifikuje pixely ve snímku z videa, které nelze adekvátně vysvětlit modelem prostředí a vydává je jako binárního kandidáta masky popředí. Funkcionálně porovnává vstupní snímek videa s modelem prostředí a identifikuje pixely popředí vstupního snímku. Nejčastější metodou detekce je kontrola, zda se vstupní pixel výrazně liší od odpovídajícího odhadu pozadí (práh). Tento způsob je definovaný rovnicí (1.4)

$$|I_t(x,y) - B_t(x,y)| > Th \tag{1.4}$$

Kde I_t reprezentuje aktuální snímek a B_t je snímek modelu prostředí vytvořený v předchozí kapitole. Th představuje práh, který je třeba překonat, aby daný pixel byl klasifikován jako popředí. Prahy se většinou volí experimentálně. Dalším z populárních schémat detekce je založeno na základě normalizované statistiky definované rovnicí (1.5).

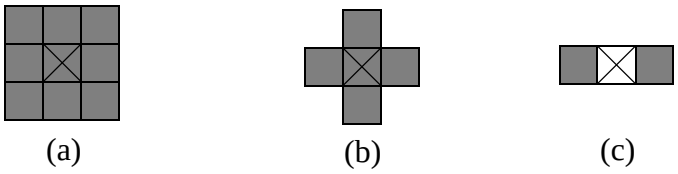
$$\frac{|I_t(x,y) - B_t(x,y) - \mu_d|}{\lambda_d} > Th \tag{1.5}$$

Kde μ_d a λ_d je průměrná a směrodatná odchylka funkce $I_t(x,y) - B_t(x,y)$, pro všechny pozice (x,y).

Poslední částí algoritmu substrakce pozadí je kontrola klíčové masky (validace). Eliminuje ty pixely, které neodpovídají skutečným objektům v pohybu a jejím výstupem je finální maska popředí. Výstup z algoritmu detekce popředí, kde se rozhoduje nezávisle na každém pixelu, bude snímek obecně velmi rušený šumem s izolovanými pixely a komponenty obrazu budou nabývat nespojitých tvarů a křivolakých hranic. V některých literaturách je validace dat definována jako proces vylepšení kandidátní masky na základě informací získaných mimo model prostředí. Výstupem validačního procesu je binární maska. Převedení kandidátní masky do binární roviny (snímek obsahuje pouze hodnoty 1 a 0) se realizuje segmentací. Pixely stupně šedi se nacházejí nad nastavenou hranicí prahu a jsou tak nastaveny na hodnotu 1. Ostatní jsou nastaveny na 0, což vytváří bílé objekty na černém podkladu nebo naopak. Proces segmentace zajišťuje snížení objemu dat a dále umožňuje aplikaci matematické morfologie, jež eliminuje šum způsobený algoritmem detekce popředí [5][6][7].

1.3.6 Matematická morfologie

Morfologická transformace je dána relací mezi obrazem s jinou typicky menší bodovou množinou, které se říká strukturní element. Strukturní element je vztažen k „lokálnímu“ počátku, který se nazývá reprezentativní bod. Typické strukturní elementy jsou na obrázku 4. Obrázek 4c ukazuje nepříliš žádoucí případ, kdy reprezentativní bod, není bodem strukturního elementu.

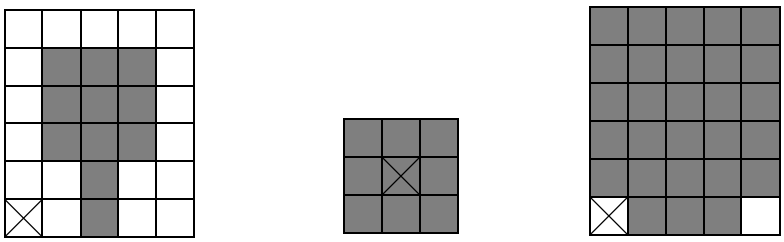


Obrázek 4: Typické strukturní elementy

Aplikace morfologické transformace na obraz je možné si představit, jako systematický posun strukturního elementu po obraze. Výsledek relace mezi obrazem a strukturním elementem se zapíše do výstupního obrazu v reprezentativním pixelu. U binárních obrazů může být výsledek relace pouze 1 nebo 0. Základními transformacemi matematické morfologie jsou dilatace a eroze. Eroze následovaná dilatací vytváří novou transformací nazývanou otevření a dilatace následovaná erozí vytváří transformaci uzavření. Výsledkem otevření i uzavření je zjednodušený obraz.

1.3.7 Dilatace

Samostatně se dilatace používá k zaplnění malých děr, a jako základ pro složitější operace. Její vlastností je zvětšování objektů. Je-li potřeba zachovat původní rozměr, musí se kombinovat s erozí, která bude popsána později. Dilatace skládá body dvou množin pomocí vektorového součtu, například $(a, b) + (c, d) = (a + c, b + d)$. Jedná se o bodovou množinu všech možných vektorových součtů pro dvojice pixelů, vždy pro jeden z množiny binárního obrazu a jeden z množiny strukturního elementu.

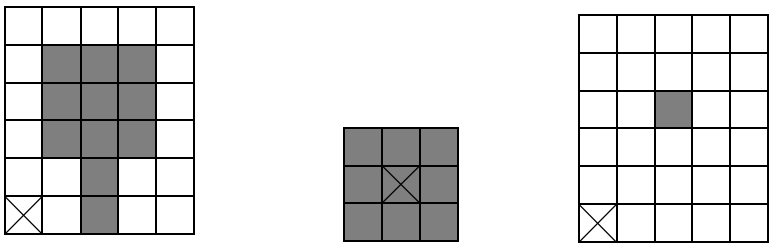


Obrázek 5: Příklad dilatačního procesu se strukturním elementem 3×3

Na obrázku 5 je vidět proces dilatace aplikovaný na jednoduchý objekt. V tomto případě byl použit *isotropický* strukturní element, který způsobuje expanzi ve všech směrech stejným způsobem. Dilataci s isotropickým strukturním elementem si lze představit jako transformaci, která změní všechny body pozadí sousedící s objektem na body objektu.

1.3.8 Eroze

Používá se ke zjednodušení struktury objektů. Samotné struktury o velikosti menší než je strukturní element využitý v erozi se ztratí úplně a složitější struktury se mohou rozdělit na více jednodušších. Eroze je duální operací k dilataci. A její princip spočívá v ověřování bodu obrazu, zda pro všechny možné body leží součet strukturního elementu a bodu obrazu v původním obraze. Pokud ano, pak do výsledného obrazu se zapíše bod popředí, v opačném případě se zapíše bod pozadí. Isotropický strukturní element o velikosti 3×3 způsobí odebrání jedné vrstvy objektu. Tento proces je zachycen na obrázku 6.



Obrázek 6: Příklad erozního procesu se strukturním elementem 3×3

Aplikace morfologických relací, na jejichž základě dochází k validaci masky popředí, je posledním krokem, který završuje celý proces substrakce pozadí. Následující krok je detekce objektu a je součástí obecného algoritmu detekce pohyblivých objektů zmíněném na začátku této kapitoly. Kandidátní maska obsahuje v ideálním případě pouze nestatické objekty dané scény, které jsou nerozlišitelné. To je dáno podstatou binárního obrazu, který zvýrazňuje objekty, ale jejich identita je dána pouze umístěním ve snímku. Detekce objektů spočívá ve vypracování postupu pro zjištění počtu objektů v obraze. Jejich tvar reprezentuje skupina pixelů (přiřazování stejného označení se nazývá homogenní region) odkazující na binární obraz [13].

1.4 Detekce objektu

Důležitou metodou pro detekci objektům, je metoda založená na procházení obrazových dat v binárním obraze. Tato část modelu popisuje algoritmus barvení oblastí známý též jako CCL (Connected Component Labeling), který získá počet objektů v binárním obraze pomocí sekvenčního označování regionů. Detekuje připojené komponenty mezi pixely v binárním obraze, který je vytvořen pomocí segmentace (prahováním) barevného nebo šedotónového obrazu. Algoritmus detekce komponent je možné aplikovat i na vícedimenzionální obraz či data. Algoritmus barvení oblastí operuje nad několika různými druhy informací integrováním rozhraní rozpoznání obrazu mezi počítač a uživatele. Tyto metody jsou rozděleny do dvou hlavních kategorií: rekurzivní a sekvenční označování spojených komponent v obraze. Rekurzivní metody se používají k nalezení homogenního regionu v binárním obraze. Vyžadují profilovaní části paměti, která je úměrná dané oblasti. Z tohoto důvodu je tato metoda vhodná zejména pro malé obrazy. Sekvenční označení regiónů je klasická nerekurzivní technika, která vyžaduje k dokončení celého procesu dva průchody o třech procesech.

1) Každému pixelu objektu v obraze se vybere dočasný zástupce na základě jeho sousedních pixelů
2) Prozatímní zástupce je nahrazen reprezentativní značkou, která je vybrána k reprezentaci všech ekvivalentních zástupců pro všechny spojené komponenty.
3) Vybraná značka je přiřazena k danému pixelu.

Běžný přístup pro výběr reprezentativního zástupce je výběr zástupce nejmenší hodnoty. Algoritmus přiřazování zástupců jednotlivých pixelům obrazu, je opět rozdělen do dvou kategorií na základě množství průchodů, které provádí [12].

Sekvenční označování regionů je proces identifikace komponent připojených prvků obrazu. V první fázi označování regiónů je obrázek skenován zleva doprava pixel po pixelu a shora dolů, kde se postupně přiřazuje prozatímní označení každému pixelu objektu v závislosti na definici okolních bodů. Tyto body jsou jednoznačně určeny maskou, která vybírá čtyři, šest nebo osm připojených komponent (pixelů). Sekvenční označování regiónů je dvouprůchodový algoritmus. V některých publikacích se nachází modifikace tohoto algoritmu, jehož úprava spočívá v odstranění druhého průchodu. Výsledkem této modifikace je zrychlení algoritmu o 5% oproti standardní verzi [36].

1.4.1 Barvení oblastí s využitím 8 sousedství

U tohoto algoritmu využíváme osm spojených komponent $N8$. Obraz o dvou dimenzích skenujeme dopředným způsobem zleva doprava a od shora dolů. Algoritmus je aplikován pouze na pixely reprezentující popředí, ostatní nejsou brány v potaz. Při nalezení pixelu popředí jsou vybrány jeho sousedé v masce pro určení vhodného označení pixelu. To znamená, že všichni čtyři sousedé $N1 = b(x, y-1)$ $N2 = (x-1, y-1)$, $N3 = b(x-1, y)$ a $N4 = b(x-1, y+1)$, se podrobují analýze, na jejímž základě je rozhodnuto o pixelu popředí. Na obrázku 7 je znázorněna maska s aktuálně vybraným pixelem a jeho sousedy, které jsou objektem zájmu.

b(x-1, y-1)	*b(x-1, y)*	*b(x-1, y+1)*
b(x, y-1)	*b(x, y)*	

Obrázek 7: Zvýraznění sousedů pixelu $b(x, y)$ u algoritmu barvení oblastí s využitím 8 sousedství

1) ***První průchod***: Každému pixelu je přiřazeno označení na základě následujících kritérií. Nalezení pixelu objektu nebo pixelu popředí $b(x, y)$ který je také nazýván jako aktuální pixel objektu. Dále je potřeba vybrat všechny sousedy pixelu dle masky na obrázku 7, na jejichž základě se dle definice (1.6) a (1.7) rozhodne o přidělení značky testovaného pixelu.

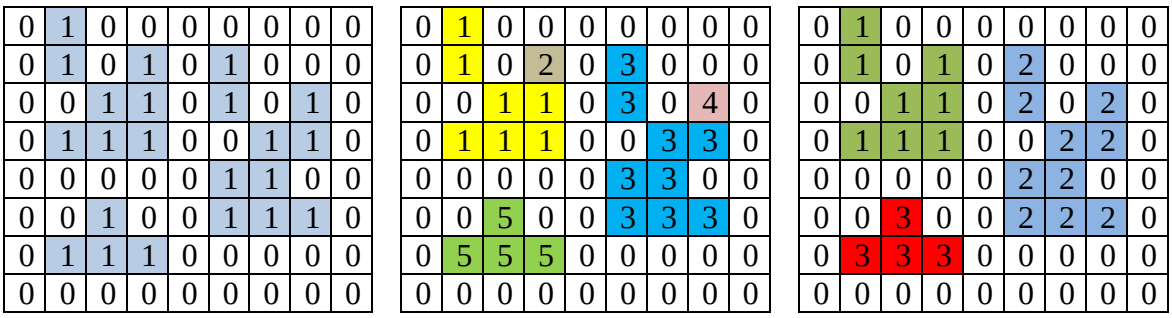
$$b(x,y) = \begin{cases} No\ Operation\ if\ P_O = 0 \\ L(L = L + 1)\ \ if\ P_O\ \neq 0\ (Min = 0 \wedge Max = 0\ ) \\ E_{min}(x,y), \qquad jinak \end{cases} \tag{1.6}$$

Kde $E_{min}(x, y)$ je minimální nenulová hodnota. L jehož inicializační hodnota je jedna a s každým krokem je inkrementován, reprezentuje označení pixelu. P_O označuje hodnotu popředí respektive hodnotu pixelu objektu. Min a Max jsou definovány následujícím vztahem (1.7).

$$\begin{aligned} Min &= \min[\{b(x-i, y-j) \mid i, j\ \in Ms\ \}] \\ Max &= \max[\{b(x-i, y-j) \mid i, j\ \in Ms\ \}] \end{aligned} \tag{1.7}$$

Kde Ms je region masky kromě pixelu objektu $b(x, y)$. $Min(.)$ a $max(.)$ jsou operátory pro výpočet minimální a maximální hodnoty sousedních pixelů v masce. Definice (1.6) jednoznačně určuje, jakým způsobem bude s nalezeným pixelem

popředí naloženo. Na obrázku 8 je vyobrazen postup identifikace regionu pomocí algoritmu využívající čtyř sousedních pixelů.



Obrázek 8: Ukázka identifikace objektu v binárním obrázku pomocí algoritmu barvení oblastí
(a) Binární obrázek, (b) První průchod, (c) Druhý průchod

Pokud je pixel objektu $b(x, y)$ zároveň pixelem pozadí pak není možné určit provizorní označení. Pokud pixel objektu skutečně náleží objektu (není pozadím) a maximální hodnota stejně tak jako minimální hodnota pixelů masky je rovna nule, pak se zvýší prozatímní označení L o jedničku a přiřadí se pixlu objektu $b(x, y)$. Je-li minimální hodnota X a maximální hodnota Y rozdílná na některém z pixelů popředí $(X \neq Y)$, pak došlo ke kolizi při přiřazování prozatímní hodnoty. V tomto případě je pixelu $b(x, y)$ přiřazena minimální hodnota X. Tato hodnota je současně zaznamenána do pole ekvivalence sdružující kolize dočasných označení. Pole je aktualizováno průběžně za chodu algoritmu. Pole ekvivalence označení pro příklad na obrázku 8 je následující [14][16].

Tabulka 1: Pole ekvivalence pro druhý průchod algoritmu

Index pixelu	0	1	2	3	4	5
Dočasné označení	0	1	1	3	3	5

2) ***Druhý průchod:*** Opět je obraz snímaný zleva doprava a odshora dolů. Při nalezení komponenty, které mají v poli shodný zástupný prvek, se pokračuje na další komponentu. Pokud se přistoupí na komponentu, jenž má v poli rozdílné dočasné označení pak je komponenta přepsána dle tabulky. Výsledkem je, že každý objekt v obraze je označen jedinečným identifikátorem jak je vidět na obrázku 8c. U obrázků se složitou konturou a vyšším rozlišením, může docházek k větvení tabulky do struktury binárního stromu. Ten je potřeba redukovat přes referenční objekty, které na sebe navazují.

1.4.2 Barvení oblastí s využitím 4 sousedství

Detekce objektů v binárním obraze si vyžaduje volbu vhodného algoritmu, protože může dojít k chybné identifikaci komponent. To způsobí navýšení či snížení počtu entit oproti reálu. Algoritmus pracuje na téměř shodném principu jako jeho modifikace zmíněná v předchozí kapitole s tím rozdílem, že analyzuje pouze dva sousední pixely. Je zde popisován z důvodu znázornění nevyhovujících výsledků na příkladu, na kterém algoritmus analyzující čtyři sousedy vykazuje dobré výsledky.

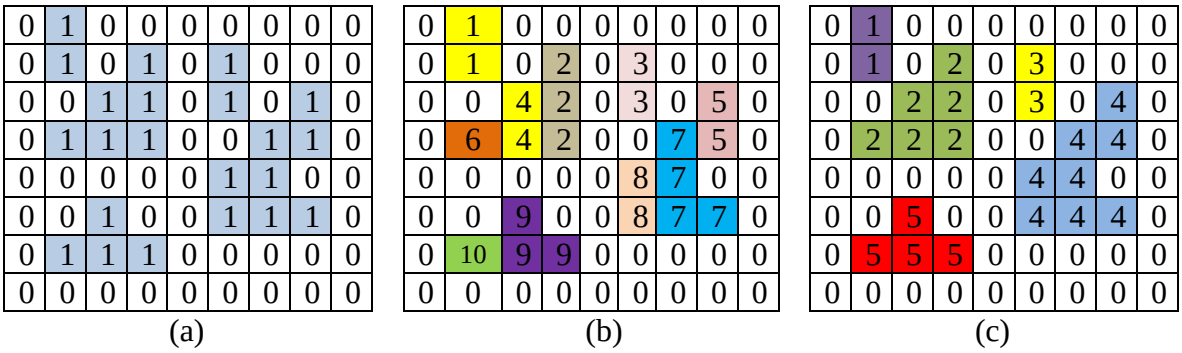
Jak již bylo řečeno, u tohoto algoritmu se analyzují pouze dva sousední pixely nacházející se nad pixelem popředí $N1 = b(x-1, y)$ a na jeho levé straně $N2 = b(x, y-1)$ jak je znázorněno na obrázku 9.

	b(x-1, y)	
b(x, y-1)	*b(x, y)*	

Obrázek 9: Zvýraznění sousedů pixelu $b(x, y)$ u algoritmu barvení oblastí s využitím 4 sousedství

1) **První průchod:** Prvním krokem je opět pří průchodu obrazem nalezení pixelu popředí $b(x, y)$, jenž je na obrázku 10a znázorněn jako regiony s hodnotou 1. Pokud existují, uloží se do mezipaměti sousední elementy dle masky na obrázku 9. Analýza obou sousedních pixelů spočívá v rozhodovacím stromu, který určí hodnotu pixelu popředí. Jsou-li hodnoty obou sousedních pixelů nastaveny na 0 čili jsou to pixely pozadí, pak je inkrementována hodnota L (implicitně 1), kterou je označen pixel popředí. Dalším stavem je, že sousední pixel nacházející se nad pixelem popředí je nastaven na hodnotu X a druhý sousední pixel je nastaven na hodnotu Y, přičemž platí, že $X \neq Y$. V tomto případě je pixel popředí označen menší hodnotou z obou sousedních pixelů. Hodnota je stejně jako u předchozího algoritmu zaznamenána do pole ekvivalence. Mají-li oba sousední pixely totožnou hodnotu, přičemž se nejedená o hodnotu pozadí, pak je pixel popředí nastaven na hodnotu jednoho ze sousedních pixelů.
2) **Druhý průchod:** Obrázek je skenován pixel po pixelu shora dolů. U každého pixelu popředí se nalezen prvek v tabulce ekvivalence na, který má být dotyčný pixel přepsán. V tomto případě bude zapotřebí učinit více kroků, protože vzhledem k masce bude i tabulka ekvivalence rozsáhlejší než v předchozím případě.

Na obrázku 10c je možné pozorovat výsledek identifikace regionu za použití masky analyzující pouze dva sousední pixely.



Obrázek 10: Rozdílný výsledek algoritmu využívající masku o dvou sousedních pixelech
(a) Binární obrázek, (b) První průchod, (c) Druhý průchod

Při pohledu na obrázky 8c a 10c je jasně patrný rozdílný výsledek řešení stejného problému. Zatímco algoritmus analyzující čtyři sousední pixely uvádí celkový počet objektů v binárním obraze **tři** u algoritmu analyzující pouze dva sousední pixely je výsledný počet objektů **pět**. Aby nedocházelo k falešné identifikaci objektů, bude ve výstupní aplikaci implementován algoritmus, který vykazuje lepší výsledky při identifikaci objektů [16].

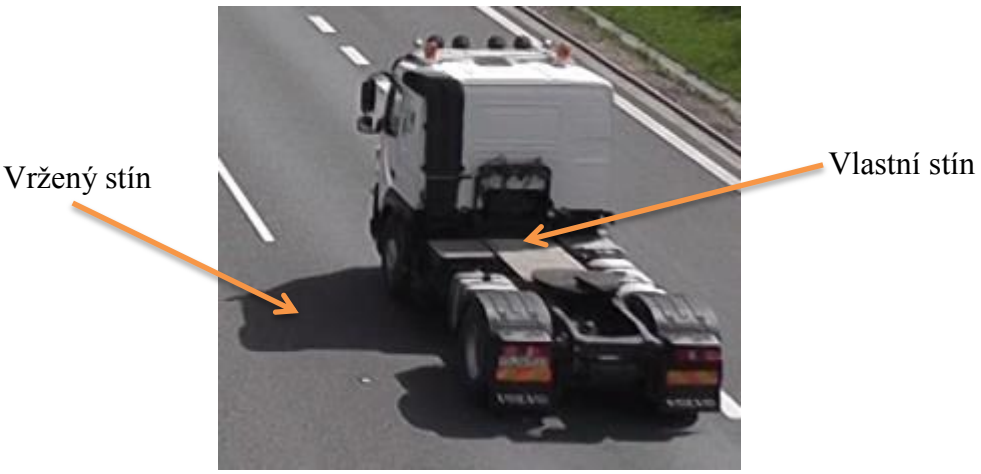
1.5 Identifikace a eliminace stínů

Identifikace a následná eliminace stínu se řadí do oblasti validace výstupních dat z procesu detekce objektu. Jedná se o jednu z nejčastějších a nejobtížnějších problematik, kterou se zpracování obrazu zabývá. Přesto je v našem případě potřeba tuto problematiku řešit alespoň na minimální úrovni, jelikož stíny zanášejí do klasifikačních algoritmů velkou chybovost. Jednou z výhod, která ulehčí celý proces eliminace je, že objekt bude klasifikován v detekované oblasti zájmu.

Z hlediska daného úhlu pohledu kamery mají stíny mnoho stejných vlastností jako vozidla, jelikož se pohybují podobným směrem, totožným vzorem a zároveň jsou značně odlišné od pozadí scény. Použité algoritmy pro detekci pohybu v obraze jsou založeny na pozadí scény respektive na modelu prostředí, a proto budou stíny detekovány jako součást vozidla. Ignorování stínů může mít za následek snížení počtu detekovaných vozidel spojením vozidel do jednoho celku při překrytí jejich stínů. Dále mohou stíny vést ke zkreslení výpočtu různých parametrů u objektů a také způsobit chybnou klasifikaci typů vozidel. Osobní automobil, který bude vrhat robustní stín,

může být klasifikován jako nákladní automobil vlivem vzrůstu jeho objemu. Lidské oko dokáže stín velmi bezpečně rozeznat, ale aby stejného úspěchu bylo dosaženo i na úrovni výpočetní techniky muselo být vynaloženo značné úsilí. Ve Wangově algoritmu [18] je navržen třístupňový proces pro odstranění stínů z popředí objektu získaného po odečtení obrazu a modelu prostředí (proces substrakce pozadí). Prvním krokem je posouzení osvětlení, ve kterém se daný jev nachází s cílem zjistit, zda obsahuje veškeré stíny na základě intenzity obrazových bodů a energie. Směr osvětlení se nachází metodou přes hranice pixelů. Plochy objektu jsou uznávány odečtením hran obrazu na popředí s hranami pozadí. Oblasti se zbývajícími hranami jsou považovány za hranice objektu. V posledním kroku je objekt obnoven použitím informací z okolí objektu a z atributů stínu.

Řešení problematiky stínů, které navrhl Wang je optimalizováno pro systémy pracující v reálném čase, s cílem poskytnou reálné výsledky pro detekci, nikoliv klasifikaci. Celé řešení se navíc opírá o řešení problému vlastního stínu a stínu vrženého. Vlastní stín je stín, který je součástí objemu automobilu, kdežto vržený stín objem automobilu zvětšuje. Tento případ je zaznamenán na obrázku 11.



Obrázek 11: Ilustrace vozidla s vlastním a vrženým stínem

Jelikož vlastní stín žádným způsobem nedegraduje algoritmy pro rozpoznávání či klasifikaci je nutné ho zachovat, aby nebyla porušena kontura vozidla. Vržený stín je objektem našeho zájmu, který chceme odstranit. Všechny tyto důvody daly podmět k vytvoření vlastního algoritmu, z části založeném na Wangově návrhu, ale principiálně bude jednodušší a pro zvýšení účinnosti bude úzce zaměřen pouze na problematiku, stínu dané scény. Princip algoritmu bude postaven na Cannyho hranovém detektoru a posuvné masce ve které se bude provádět součet obrazových elementů. Tento součet

se v každém kroku porovnává s přednastaveným prahem. Stop kritérium algoritmu bude překročení nastaveného prahu, kde nová hranice obrazu bude dána hranicí masky [18].

1.5.1 Cannyho hranový detektor

Cannyho hranový detektor využívá metodu průchodu druhé derivace nulou a v odborných literaturách je často označovaný za nejrobustnější. Využívá konvoluce Gaussiánu s obrazem a následné derivace ve směru gradientu. Významné hrany se určují prahováním s hysterezí, které zaručí souvislost hran. Některé metody, hlavně ty využívající první derivaci, nejsou schopny nalézt některé hrany, pokud je obraz zašuměný. Cannyho hranový detektor naopak vykazuje velmi malou citlivost na šum.

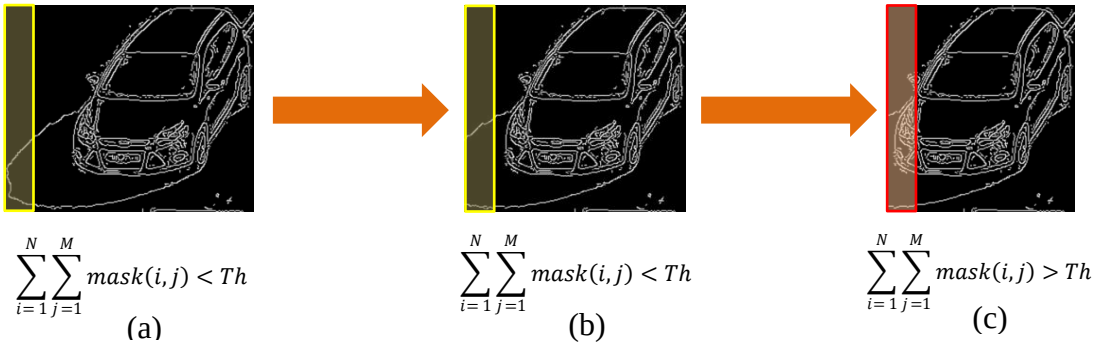
Cannyho algoritmus pro detekci hran se obvykle skládá ze čtyř kroků. Nejdříve se provede eliminace šumu Gaussovým filtrem, který je nejčastěji realizován konvoluční maskou. Dalším krokem je zjištění velikosti a směru gradientů aplikací Sobbelova operátoru. Následuje výběr lokálních maxim ze zjištěných gradientů. Posledním krokem je již zmíněné prahování, které slouží k ohodnocení významu nalezených hran, aby nedošlo k započítání jedné hrany několikrát. Doposud byly vybrány všechny hrany a to i ty hrany, které mají původ v šumu. Ty lze jednoduše odfiltrovat určením dvou prahových hodnot T_1 a T_2 se kterými se porovnávají hodnoty gradientů. Zašuměné hrany mají velmi nízké hodnoty gradientů, a proto budou odfiltrovány prahovou hodnotou T_1. Jeli hodnota gradientu vyšší než nastavený práh T_2 je bod určen jako hrana jestliže je hodnota gradientu nižší než T_1 pak se nejedná o hranu [18].

1.5.2 Návrh algoritmu eliminace stínu

Navržená metoda upřednostňuje jednoznačně odlišný přístup než obvyklé metody eliminace stínu. Místo pokusu identifikovat stín na základě intenzity pixelů se tato metoda zaměřuje na hledání hran způsobených náhlou změnou jasu. Účelem detekce hran je obecně výrazně snížit množství dat v obraze, při zachování strukturálních vlastností, které mají být použity pro další zpracování obrazu. Na obrázku 12a je možné pozorovat vozidlo převedené na hrany. Oříznutí stínu je realizováno následujícím způsobem.

1) Prvním krokem je definice masky, její šířka, a krok, o který se bude posouvat v rámci snímku.
2) Maska je posouvána o zvolený krok zleva doprava, dokud není algoritmus ukončen
3) V každém kroku je vytvořen součet všech elementů obrazu, který se porovná s nastaveným prahem.

Přesáhne-li množství elementů popředí stanovený práh, pak se algoritmus zastaví a levý okraj masky je považován za nový okraj obrazu. Jelikož nastavení ideálního prahu by bylo značně problematické i v experimentální rovině, je algoritmus vybaven schopností automaticky snižovat práh, dokud nedojde k zastavení algoritmu vlivem překročení prahu a nikoliv dosažením okraje obrazu. Tento případ je vyobrazen na snímku 12c.



Obrázek 12: Způsob posuvu masky

1.6 Rozpoznání dalších vlastností projíždějících vozidel

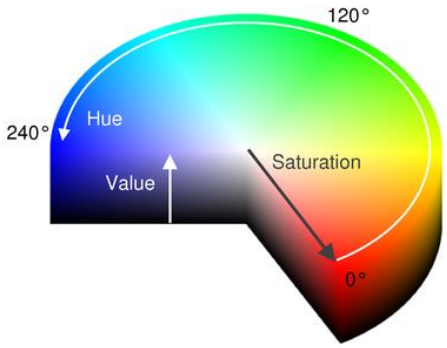
Aplikace by měla být schopna detekovat projíždějící vozidla a ty zařadit do příslušné třídy pomocí klasifikátoru. Jako dílčí funkce detekce a klasifikace vozidla je extrakce příznaků, kterými jsou rychlost a barva vozidla. Obě vlastnosti nejsou pro účel této práce významné, a proto jim nebyl věnován takový prostor. V této kapitole budou popsány algoritmy, na jejichž základě funguje extrakce barvy vozidla a výpočet jeho rychlosti. Výsledky rozpoznání jsou součástí poslední kapitoly, která hodnotí veškeré použité algoritmy.

1.6.1 Rozpoznávání barvy vozidla

U každého vozidla je zjišťována jeho barva a tvar. Rozpoznání barev je velmi složitou úlohou, kterou se počítačové vidění zabývá. Problémů na poli detekce barvy vozidla je mnoho. Jedním z takových problémů je například vlastní stín vrhaný

vozidlem, který ovlivňuje barevnou složku vozu jejím zešeřením, dále je detekce barvy závislá na místním kolísání světelných podmínek a jiných vlastnostech podnebí. Množství aspektů ovlivňující skutečnou barvu vozidla je mnoho.

První návrh algoritmu detekce barvy je založen na barevném modelu prostředí, který se odečítal od detekovaného vozidla, čímž došlo k odfiltrování rušivého prostředí a ve výsledném obrazu zůstalo na popředí pouze detekované vozidlo. Obraz reprezentovaný barevným prostorem RGB, kde jsou potřeba tři parametry k určení barvy, byl převeden do barevného prostoru HSV u kterého je k určení barvy zapotřebí pouze jeden parametr HUE (úhel) jak je vidět z obrázku 13.



Obrázek 13: Barevný prostor HSV

Tento úhel je vyextrahován z obrazu a na základě jeho hodnot je vytvořen histogram, což je grafické znázornění distribuce dat pomocí sloupcového grafu kde velikost sloupců vyjadřuje četnost sledované veličiny v daném intervalu. Barva vozidla byla určena podle nejvyššího sloupce v histogramu, protože přidružená barva se vyskytovala v obraze nejčastěji a vzhledem k odečtenému pozadí scény, by největší zástup barvy mělo obsahovat zkoumané vozidlo. Rozpoznávací skóre pro zjišťování barvy vozidel bylo nízké, jelikož se jedná o poměrně složitou úlohu. Algoritmus nebyl schopen detekovat bílá a černá vozidla, kvůli redukci parametrů, a prostor HSV se ve výsledku ukázal jako velmi omezený pro určení barvy.

Princip druhého algoritmu je postaven na pozici automobilu v obraze a řeší také problém detekce bílé a černé barvy. Jelikož jsou vozidla extrahována ze scény vždy na stejném místě, mají i stejný úhel natočení a pozici v obraze. Z obrazu se vybere staticky nastavený region, jehož pozice odpovídá umístění kapoty automobilu. Tento region je následně otestován na přítomnost bílé nebo černé barvy. Pokud region neodpovídá ani jedné testované barvě, převede se výsek do prostoru HSV, kde je opět z HUE parametru vytvořen histogram, na jehož základě je určena barva. Tento algoritmus vykazuje lepší výsledky klasifikace barvy, ale stále je značně ovlivněn prostředím.

1.6.2 Výpočet rychlosti vozidla

Základ výpočtu rychlosti vozidla je postaven na znalosti parametrů videa a příznaků extrahovaných ze snímků v části algoritmu detekce pohybu. Princip je následující. Nejprve je definována detekční a registrační linie, které budou určovat vzdálenost, jež musí vozidlo urazit. Každému vozidlu, po překročení detekční linie, bude inkrementován počet snímků, které byly zapotřebí k dosažení registrační linie. Počet snímků je klíčovým parametrem pro odhad rychlosti vozidla. Druhým důležitým parametrem je znalost skutečné vzdálenosti mezi detekční a registrační linií, která odpovídá hodnotě simulující existenci obou linii v reálném světě. Při znalosti obou těchto parametrů je možné spočítat rychlost vozidla dle vzorce (1.8)

$$v = \left(\frac{d}{f_c / f_v}\right) \cdot 3{,}6 \tag{1.8}$$

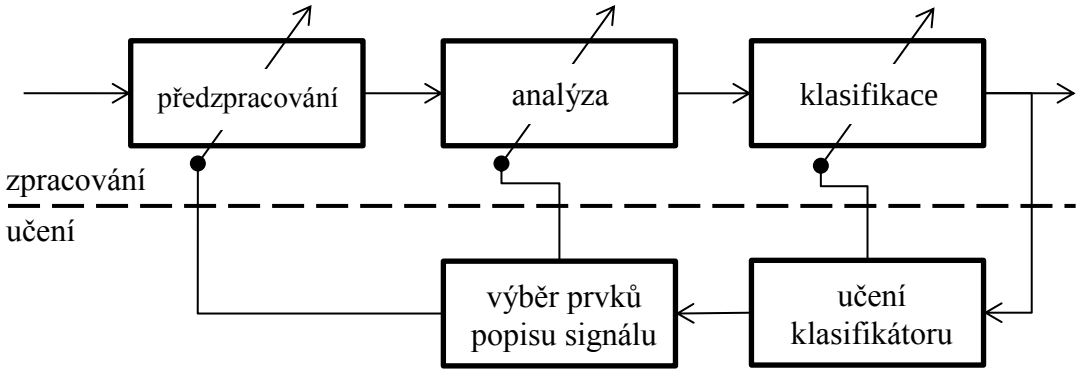
Kde v značí vypočítanou rychlost automobilu, d je reálná vzdálenost mezi detekční a registrační linií, f_c je počet snímků, které vozidlo potřebovalo k překonání dané vzdálenosti a f_v je počet snímků videa za jednu sekundu. Aby vyšel výsledek v Km/h je celý vzorec vynásoben konstantou 3,6.

Je potřeba upozornit na fakt, že daný algoritmus zanedbává lineární perspektivu. Odstranění zkreslení vlivem perspektivy si vyžaduje znalost dalších parametrů jako je výška kamery či úhel jejího natočení. Vzhledem k umístění kamery (dálniční most) nebylo možné získat některé parametry bezpečnou cestou, proto se problematika perspektivy zanedbává.

Kapitola 2

Rozpoznání a klasifikace objektů v obraze

Klasifikací rozumíme rozdělení dané skupiny objektů jevů či procesů na konečný počet dílčích skupin (podmnožin), v nichž všechny objekty, jevy či procesy mají dostatečně podobné společné informace. Klasifikaci provádíme pomocí klasifikátoru, což je algoritmus se vstupem odpovídajícím charakteru dat, popisujících analyzovaný objekt. Výstupem může být hodnota, která klasifikuje danou třídu. Na obrázku 14 je znázorněné blokové schéma pro klasifikaci založené na obrazových datech (Holčík 1992). Ze schématu je patrné, že proces klasifikace je součástí zpracování dat, které lze rozdělit do tří bloků. Předzpracováním obrazových dat se zabývá celá první kapitola. Druhým krokem je analýza obrazu, u které se zpracovávaný signál převádí na formální popis. Formální popis je často spojován také s redukcí dat, která je realizována například analýzou hlavních komponent *PCA*. Z obrazu se vybere množina elementárních vlastností popsané v předem známé formě a následně dochází k vytvoření obrazu daného signálu v příznakovém prostoru. Obraz signálu má tvar n-rozměrného vektoru, jehož složky tvoří příznaky. Výstupem z analýzy je tedy popis signálu, na který je následně aplikován proces klasifikace. Ten rozdělí signály pomocí rozhodovacího pravidla a klasifikátorů do tříd.



Obrázek 14: Schéma zpracování obrazových dat

Cílem počítačového rozpoznávání objektů je tedy klasifikovat a následně přiřadit daný objekt do jedné ze skupin, které mají být rozpoznány. Klasifikovány mohou být jednorozměrné signály (řeč) nebo i vícedimenzionální (obraz). Obecně existují dvě

hlavní třídy definující klasifikační problematiku. Většina klasifikátorů se vypořádává s takzvanou uzavřenou množinou (*close-set*), což znamená, že držíme všechny hlavní informace o třídách a jejích klasifikátorech. V otevřené množině máme pouze předběžné informace o jedné specifické třídě. Existuje, mnoho aplikací pro rozpoznávání objektů. V automatizované detekci a následné klasifikaci objektu je hlavním cílem rychlá a automatická detekce a klasifikace objektů, které jsou reprezentovány jako velké množství dat (typicky obrázky) s minimální lidskou intervencí. Další z možných aplikací pro rozpoznávání objektů v obraze je biometrika, jako je rozpoznávání obličejů, či otisků prstů pro identifikaci jedince. Většina klasifikačních algoritmů, byla navržena právě za účelem detekce osoby a verifikace obličeje pro její autentizaci.

Problému automatického rozpoznávání se vědci i univerzity po celém světě věnují již řadu let, díky čemuž vzniklo poměrně velké množství algoritmů, které mají různý pohled na danou problematiku. Rozpoznávací algoritmy se dělí do tří základních skupin. Tyto skupiny jsou založeny na studiích lidského způsobu rozpoznávání objektů.

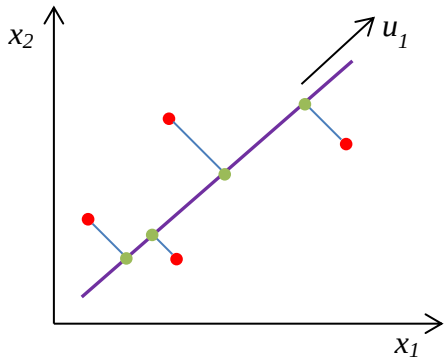
1) ***Statické modely.*** Základní princip těchto algoritmů je posuzování objektů globálně bez zaměření na jednotlivé rysy. Díky této vlastnosti patří statické algoritmy mezi nejvyhledávanější s nejefektivnější. Jejich výhodou je také rychlost a snadný převod do prostoru nejlépe reprezentujícího obrázky.
2) ***Strukturální modely.*** Oproti statickým modelům se liší zejména v analýze lokálních vlastností, které je potřeba vyextrahovat a až poté je zpracovat klasifikátorem.
3) ***Hybridní modely.*** Nejméně rozšířené algoritmy, které kombinují vlastnosti statických i strukturálních modelů. Nejvíce se blíží lidskému způsobu rozpoznávání.

Klasifikátory obvykle využívají pozorování získané z trénovací sady dat a vytvářejí co nejefektivnější predikční funkci výběrem vhodných příznaků, aby se na jejich základě dala klasifikovat testovací sada. Přesnost predikce je založena na tom, s jakou úspěšností daný klasifikátor rozděluje testovací data do skupin. Pokud obsahuje každá třída různý počet dat určených k natrénování, pak je nutné uvádět úspěšnost každé třídy zvlášť [19]. Učení klasifikátoru je jedním ze dvou základních bloků učící

fáze dat. Výstupem z tohoto bloku je návrh obecného tvaru rozhodovacího pravidla[1], případně určení jeho parametrů, pokud je rozhodovací pravidlo parametrické. Návrh obecného tvaru rozhodujícího pravidla není formalizován a závisí tak především na zkušenostech konstruktéra s danou reálnou úlohou nebo s charakterem naměřených dat. Návrh parametrů rozhodovacího pravidla následně vede na použití nějaké optimalizační úlohy. Děje se tak na základě učební neboli trénovací množiny, která obsahuje vstupní obrazy spojené s informací o předpokládané správné klasifikaci. V našem případě se jedná o uspořádanou dvojici datového popisu snímku automobilu s identifikátorem klasifikační třídy. Je-li k dispozici trénovací množina, pak hovoříme o *učení s učitelem*. V případě, že trénovací množina není k dispozici, pak klasifikátor pracuje pouze s návrhem obecného tvaru rozhodovacího pravidla.

2.1 Analýza hlavních komponent

Analýza hlavních komponent (PCA – *Principal Component Analysis*) je využita v praktické části přiložené práce. Cílem této metody je redukce dat nebo extrakce příznaků. Redukce vícerozměrných dat je snaha o vyjádření obrazu reprezentovaným příznakovým vektorem o velikosti M, který bude mít redukovanou velikost na pouze N rozměrný prostor s co nejmenší ztrátou informace. Toho lze docílit ortogonální projekcí dat do lineárního prostoru s nižší dimenzí, kde je rozptyl dat maximalizován. Tento proces je zobrazen na obrázku 15 na příkladu dvourozměrných dat. Fialová čára zde znázorňuje prostor nižší dimenze. Ortogonální projekce bodů (červené) k tomuto podprostoru maximalizuje rozptyl promítnutých zelených bodů. Ve více rozměrném prostoru by se dále volily další směry projekce u_n ve kterých data vykazují maximální variabilitu [20].



Obrázek 15: Princip nalezení prostoru nižší dimenze

[1] Matematická operace, kterou realizuje klasifikátor

Jednotlivé obrazy projíždějících automobilů jsou popsány obrovským množstvím různých proměnných, kde každá proměnná tvoří jeden rozměr v příznakovém prostoru. To značně komplikuje klasifikační proces. Pro představu by se klasifikátor musel vypořádat s prostorem, který, může mít i tisíce rozměrů. Klasifikace na úrovni metody PCA se skládá z několika kroků. Nejdříve je třeba vytvoření podprostoru PCA a do tohoto prostoru promítnout trénovací sady dat. Dalším krokem je samotná klasifikace.

2.1.1 Vytvoření základního prostoru PCA

Předpokládejme, že máme k dispozici databázi trénovacích dat o velikosti N v podobě obrázků různých automobilů s výškou H a délkou W, které mají rozměr $d = W \times H$. V první fázi se nedříve převedou všechny obrázky z trénovací sady na sloupcové vektory x_i o rozměru $d \times 1$. Sloupcový vektor je možné vytvořit přímo skládáním jednotlivých členů matice obrazu na sebe, nebo nepřímo skládáním celých řádků matice za sebe a následným transponováním výsledného vektoru. Tyto vektory pak tvoří sloupce matice X (2.1) o rozměrech $d \times N$, která představuje základní prostor PCA.

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{d1} & \cdots & x_{dN} \end{pmatrix} \tag{2.1}$$

Dalším krok, který je nedílnou součástí řešení pro nalezení hlavních komponent, které minimalizují střední kvadratickou chybu aproximace dat, je výpočet průměrného vektoru wp z řádků datové matice X. Tento vektor je dán vztahem (2.2)

$$wp(i,1) = \frac{1}{N}\sum_{j=0}^{N} x_{ij}\,, \qquad i = 0, \dots, m \tag{2.2}$$

Po vypočítání průměru se odečtou tyto hodnoty od každého sloupce matice X, čímž vznikne vycentrovaná matice W, kde jednotlivé vzory mají nulový součet. Nyní se dostáváme k jádru metody PCA, jejím hlavním účelem je najít vlastní čísla a vektory kovarianční matice $C = W \cdot W^T$. Kovarianční matice C má v tomto případě rozměr $d \times d$ a zjišťování d vlastních čísel by zabralo velké množství výpočetního času. Pro představu, velikost trénovacích vzorů je v praktické části stanovena na 128×128 pixelů. To znamená, že velikost kovarianční matice by byla $16\,384 \times 16\,384$. Aby se vyhnulo výpočtu vlastních čísel takto enormních matic, zavedla se metoda, která

využívá značného rozdílu trénovacích dat a počtu pixelů jednoho vzoru, nazývaná také jako *snap-shot*. Pokud by nastala opačná situace, že počet pixelů vzoru je menší než celkový počet vzorů pro natrénování, jak tomu bývá u velkých databází, pak se ve výjimečných případech stává, že výpočet vlastních čísel matice je pomalejší než standardní metodou. U metody *snap-shot* se vlastní čísla a vektory vypočítávají z matice S o velikosti $N \times N$, která vznikne jako $S = W^T \cdot W$, pak tedy platí rovnice (2.3).

$$W^T \cdot W v_n = \lambda_n \cdot v_n, \qquad n = 1,2 \dots N \tag{2.3}$$

Zde označuje λ_n vlastní čísla (*eigenvalues*) matice zatímco vlastní vektory (*eigenvectors*) jsou označeny v_n Vektory původní kovarianční matice C lze dopočítat vynásobením matice W s vypočítaným vlastním vektorem v_n dle (2.4)

$$c_n = v_n \cdot W \tag{2.4}$$

Kde c_n je vlastní vektor na pozici n v kovarianční matici C a v_n je vlastní vektor na stejné pozici v kovarianční matici S. Vypočítané vlastní vektory c_n je potřeba ještě seřadit sestupně dle vlastních čísel a normalizovat podle rovnice (2.5) na jednotnou velikost.

$$c_n = \frac{c_n}{\|c_n\|} \tag{2.5}$$

Tento proces poukazuje na fakt, že vlastní vektory s největšími vlastními čísly jsou hlavní komponenty souboru dat. Komponenty nižšího významu lze zanedbat, i když tím dojde ke ztrátě informace. Tato újma na datech je, ale tak malá, že se ve výsledku projeví jen nepatrně. Výhody redukce dle vlastních čísel se značně projeví zejména při uložení struktury do souboru, pro rychlejší klasifikaci. Složením všech vektorů c_n do matice je získaná hledaná matice vlastních vektorů E o velikosti $d \times N$ která reprezentuje bázi hledaného prostoru PCA, jinak nazývaná jako *eigenspace*.

Posledním krokem, kterým se uzavírá ta část algoritmu PCA, která trénuje data je projekce známých vektorů do vlastního prostoru. Projekce je prováděna dle (2.6), čímž vznikne matice PI o rozměrech $N \times N$, jejíž každý sloupec obsahuje vektor reprezentující jeden z trénovaných obrázků.

$$PI = E^T \cdot W \tag{2.6}$$

2.1.2 Rozpoznávání metodou PCA

Po natrénování všech vzorů je možné aplikovat rozpoznávací části PCA algoritmu, která je schopna přiřadit každému vzorku svou třídu na základě naučených dat. První část rozpoznávání je shodná s kroky projekce trénovací sady obrázků. Nejdříve je od neznámého obrázku y odečten průměrný obraz wp natrénované množiny a následně je vektor transformován do charakteristického prostoru E čímž vznikne vektor PT. Tento proces projekce je zaznamenán vzorcem (2.7)

$$PT = E^T \cdot (y - wp) \tag{2.7}$$

V prostoru PT se momentálně nachází několik reprezentativních bodů natrénovaných dat. Klasifikátor zařadí zkoumaný obraz y do té třídy, jejíž reprezentativní bod má od projekce neznámého obrázku y nejmenší vzdálenost. Vzdálenost je hodnota, kterou lze považovat za míru podobnosti. Čím je vzdálenost mezi dvěma objekty vyšší, tím méně jsou si podobné [21] [22].

2.1.3 Euklidova metrika

Jedná se o metriku s nejnázornější geometrickou interpretací, která je definován vztahem (2.8). Kvadrát rozdílu souřadnic znamená, že je u této metriky kladen velký důraz na větší rozdíly mezi souřadnicemi než v lineárním případě.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.8}$$

Pro snížení náročnosti výpočtu lze ze vzorce vyjmout odmocninu, která je výpočetně na celém vztahu nejnáročnější. Pak se bude jednat o takzvanou kvadratickou Euklidovu vzdálenost, která nesplňuje trojúhelníkovou nerovnost. Klasifikace založena na relativní vzdálenosti dvou hodnot tuto možnost připouští, i když vztah by již nebyl pravou metrikou.

2.1.4 Hammingova metrika

Hammingova metrika, která je spíše známá pod anglickým ekvivalentem *city-block* (vzdálenost v městských blocích) navozuje představu uražené vzdálenosti v automobilu z místa A do místa B v pravoúhle zastavěném městském prostředí. Jeho vztah je následující (2.9)

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.9}$$

Hammingova metrika je vytvořena linearizací Euklidovy metriky což má za následek snížení výpočetní náročnosti vůči Euklidově metrice. Ze vzorce je patrné, že zde chybí mocnina rozdílu bodů což má za následek snížení významu členů s vyšším rozdílem na úroveň ostatních členů. Aby byl zachován kladný výsledek, rozdíl bodů musí být v absolutní hodnotě.

2.1.5 Minkovského metrika

Minkovského metrika (šachovnicová vzdálenost) zobecňuje jak Euklidovu tak Hammingovu metriku. Druhá odmocnina je zde nahrazena obecnou odmocninou, jejíž velikost udává váhu rozdílu dílčích souřadnic obou obrazů.

$$d(x,y) = \sqrt[m]{\sum_{i=1}^{n} |x_i - y_i|^m} \tag{2.10}$$

2.1.6 Metrika kosinové podobnosti

Ze skalárního součinu vektorů je možné odvodit metriku vzdálenosti. Na tomto principu je založena metrika kosinové podobnosti, která je definována vztahem (2.11)

$$d(x,y) = \frac{x^T \cdot y}{\|x\| \cdot \|y\|} \tag{2.11}$$

Ze vzorce je patrný předpoklad stejné délky obou vektorů, neboli že jsou vektory takzvaně normovány. Výsledná hodnota $d(x,y)$ je rovna kosinu úhlu mezi oběma vektory

2.1.7 Mahalanobisova metrika

Velmi známou a ve statistické analýze používanou metrikou je takzvaná Mahalanobisova metrika. Všechny dosud uvedené metriky neuvažují závislost mezi jednotlivými body v prostoru. Pokud zahrneme do vztahu pro výpočet vzdálenosti i závislosti mezi body vyjádřené kovarianční maticí, pak dostaneme právě Mahalanobisovu metriku. Ta je definována vztahem (2.12)

$$d(x,y) = \sqrt{(x-y)^T \cdot C^{-1} \cdot (x-y)} \tag{2.12}$$

Použitím této metriky je potlačen vliv rozdílu ve variabilitě proměnných na výsledky což není vždy žádoucí. Pokud mají proměnné párové korelační koeficienty nulové a proměnné vstupující do výpočtu jsou převedeny na normovaný tvar, pak Mahalanobisova vzdálenost odpovídá čtverci euklidovské vzdálenosti [13].

2.2 MACE

Dalším algoritmem pro automatickou klasifikaci objektů je MACE (Minimum Average Correlation Energy). Korelační filtry byly úspěšně aplikovány na problematiku automatického cíleného rozpoznávání. Základní korelační filtr je MSF (Matched Spatial Filter) jehož impulzní odezva je převrácená verze referenčního snímku. Zatímco MFS filtr funguje dobře pro detekci referenčních snímků poškozených aditivním bílým šumem, má velmi špatné výsledky pokud se objeví referenční snímek, který byl nějakým způsobem zdeformován, jako například rotací či změnou měřítka. Z tohoto důvodu je pro úspěšnou detekci mít několik MFS filtrů jenž detekují vždy jiný vzhled objektu což je velmi neatraktivní pro praktické rozpoznávání. Hester a Casasent tento problém řeší zavedením filtru syntetické diskriminační funkce (SDF - Synthetic Discriminant Function). SDF filtr je lineární kombinací filtru MFS, který omezuje výstupní hodnoty na počátku korelační roviny. Algoritmus využívá lineární kombinace trénovacích obrázků. Pokud trénovací obrázky obsahují širokou škálu různých deformací pak je možné s tímto filtrem dosáhnout dobrých výsledků. Nevýhodou tohoto filtru je neostrý vrchol (*peek*), generovaný výstupem korelace, což způsobuje jeho složitou lokalizaci. Kvůli výše zmíněným důvodům byl vytvořen algoritmus MACE, který řeší neostrost generovaného vrcholu při detekci objektu [23][24].

2.2.1 Design filtru MACE

Pří návrhu filtru MACE, se stejně jako u algoritmu PCA, uvažuje dvourozměrné matice (obrázek) převedený na sloupcový vektor o velikost $d\ x\ 1$, kde d určuje celkový počet pixelů v obrázku. Vektor obrázku označíme jako x_i . Filtr MACE je lépe formulován ve frekvenčním spektru do, kterého převedeme vektory obrázků x_i pomocí Diskrétní Fourierovi transformace (DFT). Takto převedené vektory označíme X_i . Tímto způsobem převedeme všechny obrázky z trénovací množiny, čímž vznikne matice X definovaná jako (2.13). Sloupce této matice jsou lexikograficky seřazeny.

$$X = [X_1, X_{2,\ldots,} X_N] \tag{2.13}$$

Velikost matice X je $d \times N$ kde N je počet obrázků v trénovací množině. Nechť je vektor h filtr v běžném spektru reprezentovaný jeho Fourierovou transformací označenou H. Nás momentálně zajímá korelace mezi vstupním obrázkem, který chceme zařadit do třídy a filtrem. Korelace i-tého obrázku sekvence $x_i(n)$ s filtrem $h(n)$ je definován jako:

$$g_i(n) = x_i \otimes h(n) \tag{2.14}$$

Podle Parsevalova teorému, může být korelační energie i-tého obrázku zapsána kvadratickou formou jako (2.15)

$$E_i = H^H D_i H \tag{2.14}$$

Kde D_i je diagonální matice o velikosti $d \times d$, přičemž její elementy jsou umocněná magnituda asociativního elementu X_i, což je energie spektra $x_i(n)$ a horní index H značí Hermitovskou transpozici. Hermitovská transpozice, má význam transpozice matice a nahrazením prvků matice jejich komplexně sdruženými verzemi.

Cílem algoritmu MACE je minimalizování průměrné korelační energie korelačních vstupů, při současném splnění podmínky omezení maximální hodnoty v počátku na předem definovanou hodnotu vektorem c. Hodnota korelační energie v počátku může být zapsána jako (2.15)

$$g_i(0) = X_i^H H = c_i \tag{2.15}$$

Pro N trénovacích obrázků kde c_i je uživatelem definována výstupní hodnota korelace v počátku (obvykle 1) pro i-tý obrázek. Potom průměrná energie všech trénovacích obrázků je vyjádřena jako (2.16).

$$E_{avg} = H^H D H \tag{2.16}$$

Průměrná energie se spočítá stejně jako energie obrázků ze vzorce (2.14) s tím rozdílem, že je nejdříve potřeba spočítat průměrné hodnoty matice D dle (2.17)

$$D = \left(\frac{1}{N} \sum_{i=0}^{N} D_i \right) \tag{2.17}$$

Návrh filtru MACE, spočívá v minimalizaci E_{avg} při současném splnění omezení $X^H H = c$ kde c je N dimenzionální vektor. Tento optimalizační problém lze řešit použitím Lagrangeových multiplikátorů.

$$H = D^{-1}X(X^{+}D^{-1}X)^{-1}c \tag{2.17}$$

Z výsledného vzorce (2.17) je patrné, že k navržení filtru MACE je potřeba dvou inverzních matic, což by mohlo být výpočetně náročné. Matice D obsahuje prvky pouze na své diagonále a její inverzí podoba D^{-1} je tedy realizovatelná jednoduchým invertováním jejich hodnot.

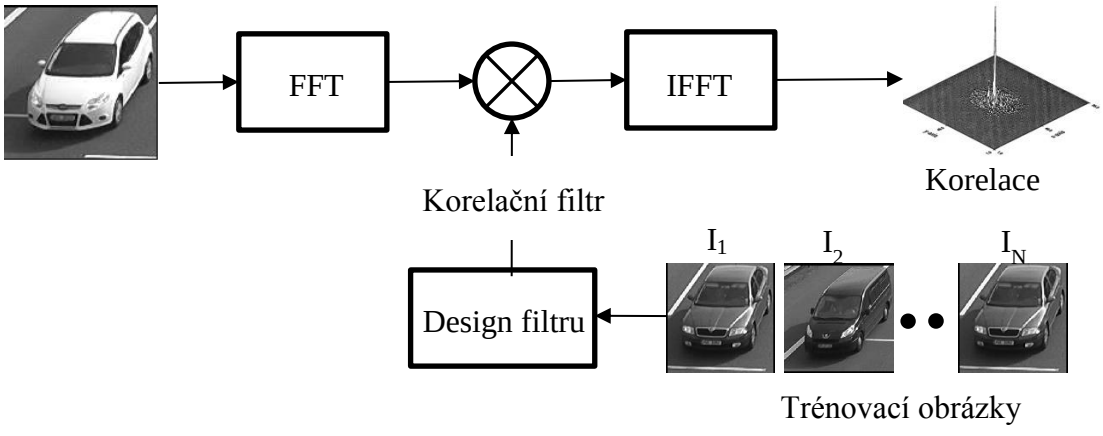
Filtr OTSDF (Optimal Trade-off Synthetic Discriminant) je dobře známý korelační filtr, který je navržen pro překonání špatné generalizace filtru MACE, když je na vstupu přítomen šum. OTSDF filtr je definován vzorcem (2.18)

$$H = T^{-1}X(X^{H}T^{-1}X)^{-1}c \tag{2.18}$$

Kde $T = \alpha D + \sqrt{1-\alpha^2}C$ a $0 \leq \alpha \leq 1$. Matice D je diagonální matice z návrhu MACE filtru a C je diagonální matice obsahující energii vstupního šumu jako spektrální hustotu výkonu. Výsledkem je mnohonásobně nižší zašumění ve výsledné korelaci ale také mnohem menší ostrost vrcholu [25].

2.2.2 Rozpoznání metodou MACE

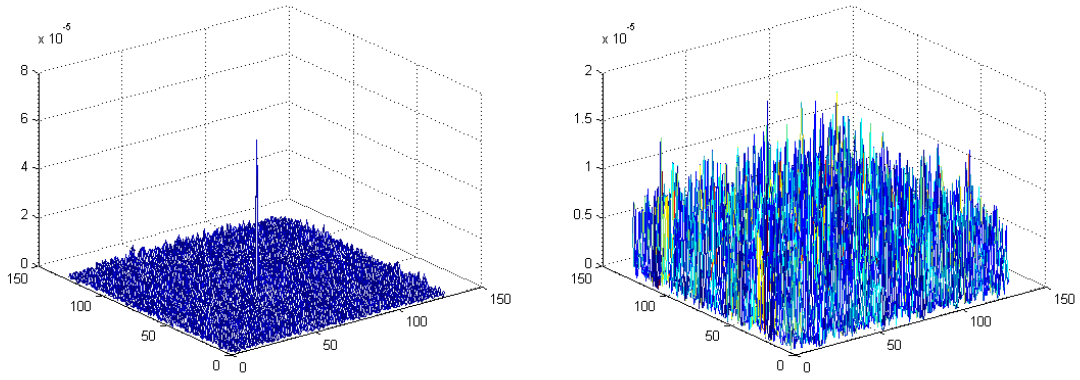
Rozpoznávání objektů se provádí pomocí vzájemné korelace (*cross-correlation*) vstupního obrazu se syntetizovanou šablonou nebo filtrem a zpracování výsledného výstupu. Obrázek 16 schematicky znázorňuje, jak je vzájemná korelace provádí účinně za použití rychlé Fourierovy transformace (FFT).



Obrázek 16: Blokový diagram korelačního procesu

Vstupní obrázek je nejprve převeden do frekvenční oblasti a přetransformován na sloupcový vektor. Tento vektor je prvkově vynásoben (*element-wise*) s navrženým filtrem, který je natrénován s určitou množinou obrázků patřící do stejné třídy. Ve

výstupu korelace se hledají již zmíněné vrcholy a relativní výška těchto vrcholů se používá k určení, zda je objekt zájmu přítomen či nikoliv. Polohy vrcholů indikují polohu objektu. Předpokládá se, že je-li testovaný obrázek v souladu se zkušebními obrázky daného filtru, pak korelační výstup obsahuje ostrý vrchol. V opačném případě obsahuje korelační výstup pouze náhodný šum, bez významného vrcholu. Tento šum indikuje falešnost testovaného obrázku, který pravděpodobně bude patřit do jiné třídy.



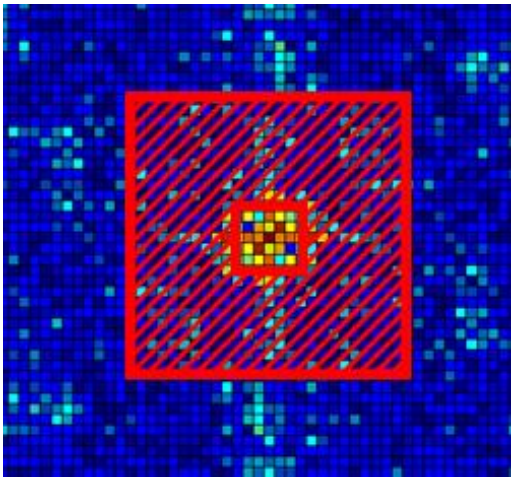
Obrázek 17: Vlevo – korelace filtru a obrázku ve stejné třídě, Vpravo – korelace filtru a obrázku v různé třídě

Parametr PSR (Peak to Side Lobe Radio) je metrika sloužící k určení, zda testovací obrázek patří do autentické třídy či nikoli. Na základě tohoto parametru lze přijmout rozhodnutí o korelaci neznámého obrázku a filtru. PSR je dobrým měřítkem pro vyhodnocení ve srovnání s korelačním vrcholem, protože je invariantní vůči jednotným změnám osvětlení ve zkušebních obrázcích. Parametr je definován takto:

$$PSR = \frac{p - \mu}{\sigma} \tag{2.19}$$

Kde p je hodnota nejvyššího vrcholu dále μ je okolí vrcholu a σ je standardní odchylka. PSR měří ostrost vrcholu korelačního výstupu což je přesně to, co se filtry typu MACE snaží maximalizovat. Proto čím vyšší je PSR, tím je pravděpodobnější, že zkušební obraz patří do dané třídy. Rozhodovací pravidlo může být vedeno přednastaveným prahem, jehož velikost bude kritérium zařazení obrázku do třídy. Způsob, vykazující lepší výsledky při praktické klasifikaci je založen na vzájemném porovnání jednotlivých PSR různých filtru a filtr s nejvyšší hodnotou určí třídu neznámému obrázku. V mnoha aplikacích, kde se využívají filtry MACE, je pro určení třídy zkoumána pouze hodnota korelačního vrcholu, ale je třeba si uvědomit, že rozhodnutí o autentizaci snímku se, kterým nakládáme (při použití PSR metriky) není založeno na jedné projekci, ale na mnohačetné projekci.

Okolí vrcholu je typicky voleno způsobem, znázorněným na obrázku 18, kdy je z korelační roviny vybrána čtvercová oblast o velikosti 20×20 pixelů *(sidelobe region*) ze které se vypočítá standardní odchylka σ a průměr μ (vyjma centrální masky 5×5) [6][26]. V některých literaturách je uváděn region o velikosti 25×25 ze, kterého se vyjímá čtverec 5×5. To pro jak velkou oblast je vhodné počítat parametr PSR je dáno zejména rozlišením trénovacích obrazů.



Obrázek 18: PSR region (výstup MACE filtru pohled shora)

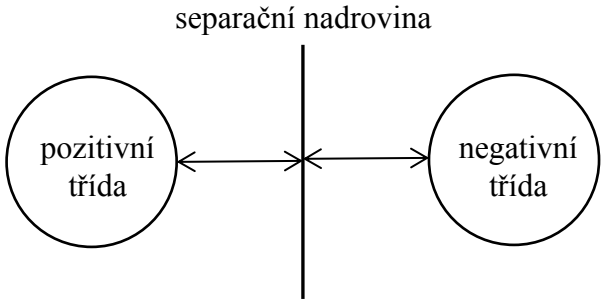
2.3 Support Vector Machine

Support Vactor Machine (SVM) je technika strojového učení původně vyvinuta Vladimírem Vapnikem určená pro řízenou klasifikaci. SVM jsou lineární klasifikátory založené na konceptu rozdělujících rovin, které definují rozhodovací hranice. Rozhodující rovina je rovina oddělující od sebe soubor objektů patřící do různých členských tříd. Je potřeba poznamenat, že metoda SVM je schopna konkurovat i aktuálně používaným metodám, jako jsou neuronové sítě, při řešení problému rozpoznání vzoru. SVM je schopné se naučit jejich třídu přes trénovací set definovaný rovnicí (2.20)

$$\{\vec{x}_i, y_i\}, \vec{x}_i \in R^n, y_i \in \{-1,1\}, i = 1 \dots l \tag{2.20}$$

Každá instance l trénovacích dat obsahuje n-dimenzionální vektor $\vec{x}$ který popisuje její vlastnosti a značku y, která rozděluje instance do dvou kategorií 1 (pozitivní) nebo -1 (negativní). V případě dostatečně velké tréninkové sady je metoda SVM schopna rozdělit dříve neviděné vzory (instance dat) s nedefinovanou značkou, do jedné ze dvou kategorií. V případě základní lineární klasifikace vytvoří SVM oddělovací nadrovinu,

která se nachází v potencionálně transformovaném vstupním prostoru. Díky binární možnosti rozhodování, rozdělí nadrovina pozitivní a negativní vzorky tak, aby vzdálenost mezi příslušnými třídami od nadroviny byla maximální. Tento přístup lze hodnotit i z geometrického hlediska. SVM se snaží o vytvoření povrchu, který půlí prostor R^n tak, aby všechny instance, které patří do pozitivní třídy, byly na jedné straně plochy a všechny instance patřící do záporné třídy byly uvedeny na straně druhé, jak je znázorněno na obrázku 19. I když tento přístup není nový v klasifikační oblasti, SVM je odlišné při implementaci. Aby se dosáhlo maximálního rozpětí mezi třídami, a rozhodovacím povrchem musíme definovat konvexní obal pro danou třídu a maximalizovat rozpětí ve vztahu k tomuto obalu. To proto, že nejbližší přístup určité třídy k rozhodovacímu povrchu nemusí být vždy ve specifickém místě, ale v lineární kombinaci bodů [27] [28].



Obrázek 19: Definice separační (rozhodovací) nadroviny a její rozdělení dvou tříd

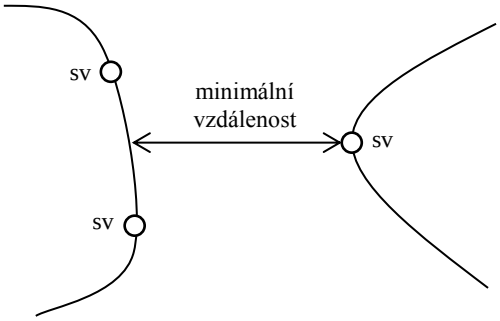
Formálně je konvexní obal definovaný jako soubor bodů S v n-rozměrech kde průnik všech konvexních množin obsahuje S [31]. Pro N bodů $p_1, \dots, p_n$ je konvexní obal C dán následující definicí (2.21).

$$C \equiv \left\{ \sum_{j=1}^{N} \lambda_j p_i : \lambda_j > 0 \; pro \; všechna \; j \; a \; \sum_{j=0}^{N} \lambda_j = 1 \right\} \tag{2.21}$$

Jak již bylo uvedeno výše SVM jsou lineární klasifikátory, které sestrojí rozhodovací plochy (nadroviny) mezi konvexní obaly tříd. Nicméně použitím kernel funkcí umožňuje SVM nalézt nadroviny v rozšířeném prostoru, který je ekvivalentní pro nalezení nelineární oddělovací roviny v původním prostoru. To umožňuje nelineární klasifikaci. Při zvažování nelineární klasifikace je výsledný SVM algoritmus formálně podobný, kromě faktu že každý skalární součin je nahrazen nelineární kernel funkcí.

2.3.1 Lineárně oddělitelná data

Lineárně oddělitelná data znamená, že se konvexní obaly tříd nepřekrývají, takže skupiny mohou být lineárně odděleny. V případě že jsou třídy lineárně oddělitelné je potřeba najít nadrovinu která maximalizuje vzdálenost mezi konvexními obaly. Jak již bylo řečeno výše, nelze použít body definovaných tříd jako potencionálně největší vzdálenost dvou tříd, neboli lokaci kde bude nadrovina s maximálním rozpětím vytvořena. Minimální vzdálenost může být u konvexního obalu mezi bodem a lineární kombinací bodů definující obal, jak to ukazuje obrázek 20.

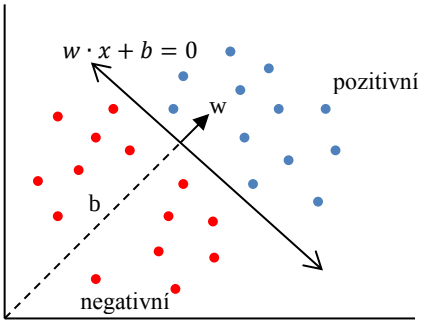


Obrázek 20: Minimální vzdálenost dvou konvexních obalů

To vede k nadrovině, která definuje rozhodovací plochu pro každou instanci dat $(\vec{x})$, určenou vzorcem (2.22)

$$\vec{w} \cdot \vec{x} + b = 0, \qquad \vec{w} \in R^n, \qquad b \in R \tag{2.22}$$

Kde $\vec{w}$ je normála nadroviny čímž udává její orientaci a b je úměrná vzdálenosti originálu daná $(\|b\|/\|\vec{w}\|)$. Vzdálenost mezi konvexními obaly tříd se nazývá „rozerva“, která odděluje nadroviny a postupem maximalizuje jejich odstup. Tento postup vede k zobecnění SVM, kde instance vybrané z obou tříd, pozitivní i negativní správně definují pozici a orientaci zvolené oddělující nadroviny. Výsledek lineárního oddělení dvou tříd je znázorněn na obrázku 21.



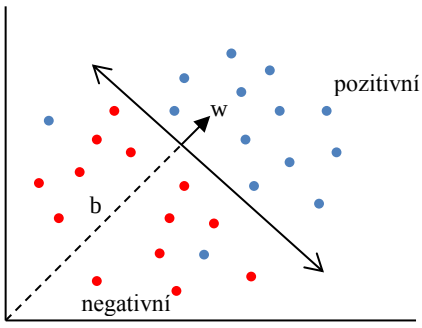
Obrázek 21: Lineárně oddělená třídy

2.3.2 Nelineárně oddělitelná data

Aby se v reálném prostředí vyskytovala data, jež lze oddělit lineárním způsobem je spíše nepravděpodobné. V případech kdy data nejsou oddělitelná lineární cestou, se konvexních obaly příslušných třídy překrývají. Z tohoto důvodu musíme definovat nadrovinu, která maximalizuje rozpětí u minimální vzdálenosti mezi konvexními obaly. Z toho vyplývá, že konvexní obal každé třídy musí být redukován na jejich těžiště, dokud není možné třídy oddělit. Těžiště je definováno jako průměr všech bodů, ve třídě, což znamená, že se bude nacházet uprostřed dané třídy. Při takto rozsáhlé redukci je nevyhnutelné, že některá data skončí na špatné straně povrchu rozhodovací nadroviny. S ohledem na tuto skutečnost se SVM současně snaží minimalizovat chybné klasifikace při maximalizaci rozpětí. Aby bylo možné definovat tento dvojí cíl je zaveden penalizační operátor za chybné klasifikace. Z rovnice SVM se tedy stává (2.23).

$$\begin{aligned} y_i(\vec{w} \cdot \vec{x}_i + b) &\geq 1 - \xi_i \\ kde\ \xi_i &\geq 0 \\ a\ i &= 1, \dots, l \end{aligned} \tag{2.23}$$

Ve výše uvedeném vzorci představuje ξ_i celkovou penalizaci pro jednotlivé chybné klasifikace a $y_i \in \{-1, 1\}$. Bez ohledu na to jestli jsou data oddělitelná či nikoliv jsou datové body klasifikovány v závislosti na jaké straně nadroviny se nacházejí. Nelineární separace tříd je k vidění na obrázku 22 [28][30].



Obrázek 22: Nelineární separace tříd

Modré body na obrázku nacházející se v oblasti negativních bodů jsou špatně klasifikovaná data vlivem redukce.

2.3.3 Nelineární SVM

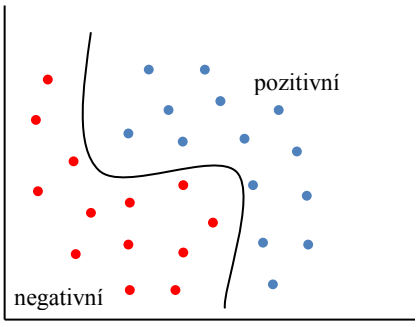
Metoda *Support Vektor Machine* není omezena pouze na čistě lineární aplikace. Jednou z výhod SVM je jeho schopnost generalizovat postup s cílem umístit rozhodovací nadrovinu jako nelineární funkci z trénovacích dat. Toho je dosaženo prostřednictvím mapování datových bodů ve vstupním prostoru do vícedimenzionálního prostoru.

$$F = \{\phi(\vec{x}): \vec{x} \in R^n\}, \qquad \phi: R^n \to R^m, \qquad m > n \tag{2.24}$$

Mapování jednoduše přidá další atribut k datům, která jsou nelineární funkcí atributů původních dat. Klasifikace pak může být realizována na základě stávajících lineárních algoritmů, na rozšířenou sadu dat v prostoru, což vytváří nelineární funkce v původním vstupním prostoru. Požadovaný výpočet klasifikace je dosažen pomocí kernel funkcí. Při použití kernel funkcí se předchozí algoritmus obsahující výpočet skalárního součinu modifikuje substitucí kernel funkce za skalární součin. To je možné díky Mercerově podmínce, která uvádí, že každý pozitivní *semi-definitní* kernel může být vyjádřen jako skalární součin ve vícerozměrném prostoru.

$$F(x, y) = \phi(x) \cdot \phi(y) \tag{2.25}$$

Kernel funkce mohou být v SMV využity vzhledem k tomu, že vstupní data jsou využívána pouze k výpočtu skalárního součinu $\overrightarrow{x_i} \cdot \overrightarrow{x_j}$ nebo v případě že byla data mapována do prostoru $\phi(x) \cdot \phi(y)$. Nelineární rozhodovací povrch je znázorněn na obrázku 23 [30].



Obrázek 23: Nelineární rozhodovací plocha

Kapitola 3

Návrh systému pro detekci a rozpoznávání vozidel

Jedním z cílů této diplomové práce je navrhnout a implementovat funkční aplikaci, která by byla schopna automaticky detekovat a rozpoznat pohybující se vozidlo. V této kapitole budou popsány jednotlivé fáze realizace softwarového řešení problematiky detekce a rozpoznán. Celý návrh je doplněn modely o diagramy tříd pro lepší orientaci při popisu funkčních bloků. Pracovně byla tato aplikace nazvána **VehicleClassification**. Název má evokovat výstup této práce, protože rozpoznání vozidla bude postaveno na principu zařazení objektu do některé ze tříd. Aplikace je napsána v jazyce *C#* což je jazyk vhodný pro vytváření desktopových aplikací cílené na platformu Windows. Spojuje silné vlastnosti jazyka *C* a objektově orientovaného jazyka Java. Použité vývojové prostředí je *MS Visual Studio 2012*.

3.1 Knihovna EmguCV

Zpracování obrazu na úrovni detekce a klasifikace není triviální problém a proto bylo třeba před započetím implementace zvolit vhodnou knihovnu, značně urychlující celkový vývoj. Při analýze výběru knihoven pro zpracování obrazu v jazyce C# bylo nakonec přistoupeno k otevřené knihovně EmguCV, obsahující jako jediná značný počet metodik pro analýzu videa. EmguCV je multiplatformní .NET wrapper pro knihovnu OpenCV jenž je navržena pro práci s kompilovanými jazyky jako je C/C++. Technicky se jedná o sadu dynamicky linkovaných knihoven, vytvářející most mezi funkcemi implementovanými v knihovnách OpenCV. Poskytuje také několik grafických komponent.

Knihovna OpenCV (Open Source Computer Vision Library) byla vytvořena firmou Intel a kromě volně dostupné verzi, je k dispozici i verze pro komerční účely. Knihovna obsahuje rozsáhlou kolekci funkcí a tříd sepsaných v jazyce C, implementující různé algoritmy pro zpracování obrazu. Rovněž jsou přítomny primitivní funkce určené například pro filtrování nebo statické funkce nad obrázky. Hlavním důvodem proč byla tato knihovna zvolena, respektive její wrapper, bylo její zaměření na funkce vyšší úrovně, především na analýzu, segmentaci a klasifikaci objektů. Kvůli přítomnosti velkého množství algoritmů zajišťující práci s obrazem, je

knihovna rozdělena do několika částí, které se zaměřují na řešení vždy určité problematiky. Zde je několik kategorií zajímavých pro tuto práci:

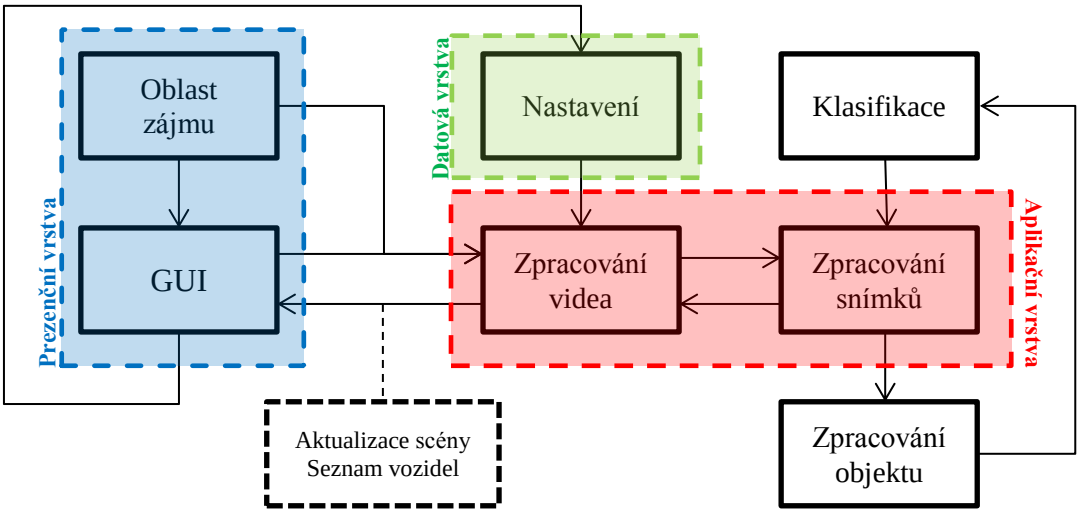
1) ***Structure*** - Zde jsou definovány obrazové struktury a třídy pro jejich prezentaci.
2) ***CV*** - Kategorie zodpovědná pro zpracování obrazu na nízké úrovni (filtry,…).
3) ***MML*** – Implementace strojového učení.
4) ***FFmpeg*** – Slouží pro práci s videem.

Aplikace by měla být vyvíjená včetně grafického rozhraní, aby bylo možné kontrolovat nastavení některých algoritmů. Zde se nabízí grafické komponenty, jež jsou součástí balíku knihoven a umožňují efektivnější zobrazování zpracovávaných snímků videa. [35]

3.2 Objektově orientovaná analýza

Cílem objektově orientované analýzy (OOA) je vytvořit konceptuální model informací, jenž patří do dané oblastí zkoumání. Nebere se zde v potaz jakákoliv omezení či problematika implementace.

Zpracovávání obrazu je značně závislé na nastavení jednotlivých algoritmů a před uvedením do provozu je v průběhu vývoje třeba aplikaci testovat a ladit. Jelikož podstata práce úzce souvisí s visuální problematikou, není v tomto případě možné vyvíjet pouze konzolovou aplikaci, zobrazující výsledky. Nicméně návrh musí být natolik flexibilní, aby vizualizace scény nezpomalovala proces přehrávání a detekce probíhala v reálném čase. Blokové schéma návrhu aplikace se znázorněnými vrstvami aplikace je zachyceno na obrázku 21.



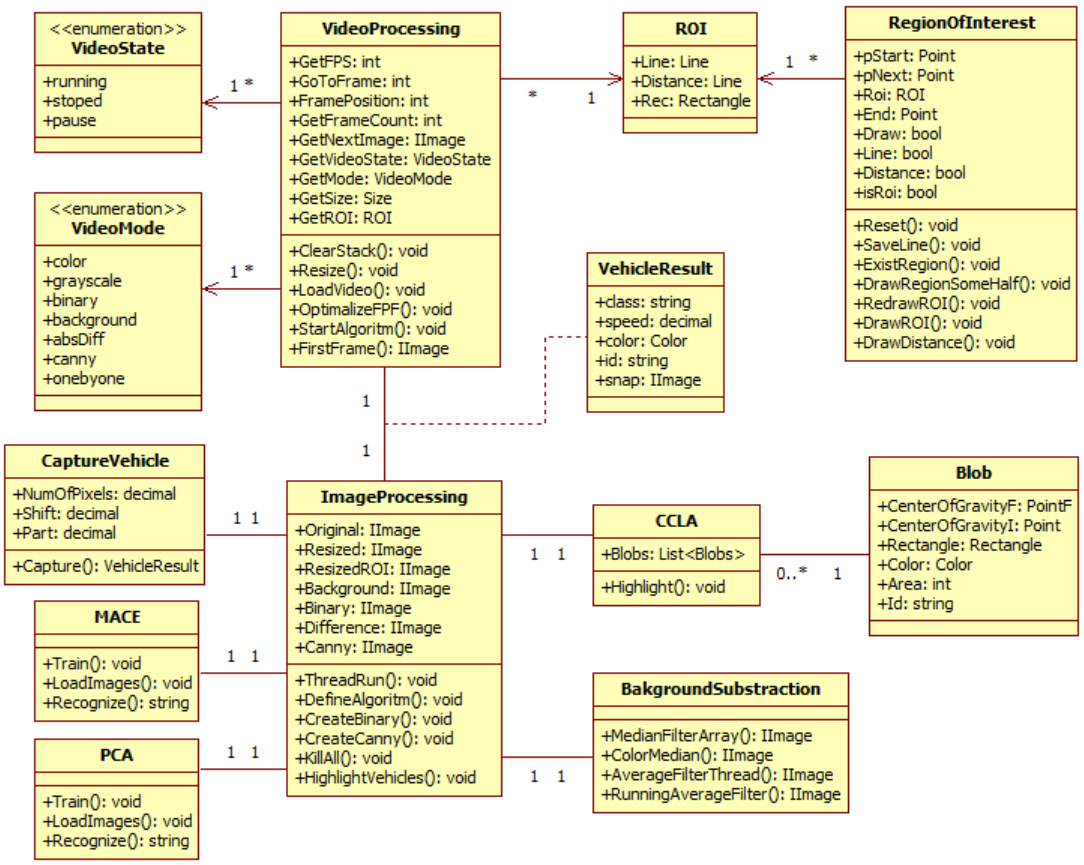
Obrázek 24: Blokový návrh aplikace se znázorněnými vrstvami

Jedná se o velmi hrubý návrh, ve kterém nejsou zahrnuty všechny části, aplikace potřebné ke své autonomní činnosti, ale pouze hlavní bloky, nesoucí celkovou funkcionalitu. Návrh stojí pouze na třech základních blocích. Jsou jimi *GUI*, pro visuální interakci s uživatelem, dále pak blok *Zpracování videa* zajišťující separaci video sekvence na jednotlivé snímky a také ovládání videa (play, pause, stop a rewind). Posledním blokem je *Zpracování snímků*. Tento blok v sobě koncentruje všechny algoritmy pro zpracování obrazu, které aplikuje na snímky separované předchozím vrstvou. Zpracování obrazu je realizováno triviálními funkcemi nad obrázky z knihovny EmguCV, ale klasifikace objektu a jeho separace od prostředí je už netriviální úkol, závislí na dvou oddělených částech diagramu. *Klasifikace* je oddělena od *Zpracování obrazu* zcela úmyslně. Bude obsahovat i algoritmy pro strojové učení (Machine Learning). Ty je třeba kvůli jejich náročnosti spouštět na separovaném vlákně. Stejně tak i *Zpracování objektu* bude obsahovat algoritmy vyšší úrovně a jejich sjednocení se základními funkcemi by nebylo ideální z hlediska transakcí do vyšších vrstev aplikace.

3.3 Objektově orientovaný design

Objektově orientovaný design (OOD) přetváří konceptuální model vytvořený během analýzy na systém implementačních tříd a rozhraní. Na tomto místě se berou v úvahu omezení způsobená například danou architekturou, která byla při analýze vynechána. Cílem tohoto návrhu je popis jak má být systém vytvořen.

Podoba objektového návrhu je obdobná blokového návrhu z předchozí kapitoly. Je zde kladen důraz především na rozhraní jednotlivých bloků tedy jejich závislostí na sobě. Popis objektového návrhu je rozdělen do dvou částí – první je návrh pro realizaci obecného algoritmu pro rozpoznávání a klasifikaci automobilů a druhý specifikuje jednotlivé části algoritmu s podrobným popisem problémů, které při implementaci vznikly. Základem modulů algoritmů je rozhraní *IAlgoritms*, sloužící jako obecné rozhraní pro třídy, zpracovávající obraz. Definuje metody a vlastnosti, volané při průchodu snímku kaskádou algoritmů. Tím dojde k požadovanému zpracování a snímek se vrací reprezentační vrstvě k jeho zobrazení. Výhodou použití rozhraní je jednoznačně komplexnost algoritmů, které se mohou svou strukturou značně lišit, ale přesto budou skryty za jednoduchým rozhraním. Použití knihovny EmguCV zavádí do objektového designu rozhraní *IImage*, zobecňující snímky aplikace procházející sadou algoritmů. Není tak třeba zavádět abstraktní třídu pro snímky, jejímž účelem by bylo odstraňování implementační závislosti napříč algoritmy.



Obrázek 25: Diagram tříd jádra aplikace

Třída *VideoProcessing* je jedinou třídou, schopnou komunikovat s prezenční vrstvou aplikace na principu, jenž je specifikován návrhovým vzorem *command* ze skupiny návrhových vzorů zabývající se chováním systému. Tento návrhový vzor umožňuje GUI vytvářet požadavky dle dohodnuté struktury. Cílem bylo oddělit objekty, které vyvolávají požadavky od objektu, který zná, jak daný úkol zpracovat. Tato třída též implementuje metody pro čtení videa ze souboru a jeho ovládání pomocí knihovny EmguCV. Tím je vybudován základ pro manipulaci se snímky videa z předchozí vrstvy. Manipulace s videem je ovlivňována enumeračním uchováváním stavu, ve které se daná obrazová sekvence nachází. Tímto návrhem je docíleno oddělení závislosti objektů algoritmu od grafického prostředí.

Důležitým parametrem je *ROI* což je třída uchovávající informace o nastavené oblasti zájmu. Ta není k manipulaci s videem nijak zapotřebí a je dále předávána nižším vrstvám. Ty jí využívají pro zmenšení pracovní plochy algoritmům zpracovávající snímky z videa. S třídou ROI velmi úzce souvisí třída *RegionOfInterest*. Ta pro lepší vizualizaci přímo interaguje s komponentou zobrazující snímky videa, aby byla plocha

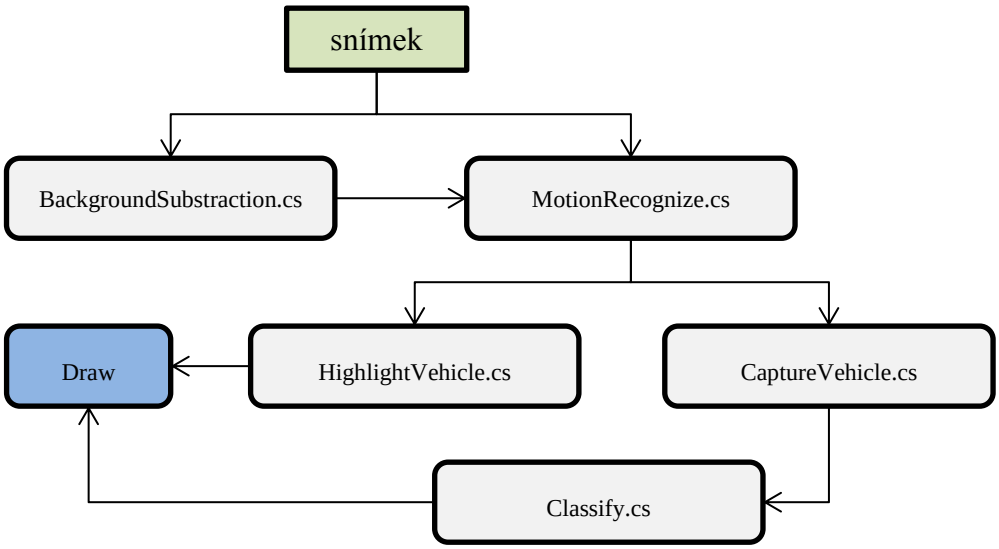
zpracování pro uživatele naprosto zjevná. Třída *RegionOfInterest* má za úkol jednak vykreslovat všechny hraniční čáry vyznačující region na který jsou aplikovány dané algoritmy, tak uchovávat souřadnice těchto hraničních čar, aby bylo možné kdykoliv danou sekvenci zrekonstruovat. Video sekvence respektive jednotlivé snímky z videa jsou předávány nižší vrstvě *ImageProcessing*, implementující řadu algoritmů na sobě závislých, jejímž posláním je prakticky celý proces detekce automobilu a jeho klasifikace. Tato třída zaobaluje a abstrahuje celý algoritmus dle návrhového vzoru fasáda (*Facade Pattern*) a výsledky zpracování předává zpět vyšším vrstvám přes svou asociativní třídu *VehicleResult*. Třída *Blob* existuje v návrhu jako mezistupeň pro obrazovou a objektovou reprezentací objektu. Lze tvrdit, že obecně reprezentuje pohyb v obraze jako korelaci mezi jednotlivými pixely a geometricky-obrazovým významem množiny.

Některé třídy jsou výpočetně více náročné, protože provádí složité manipulace s obrázkem a proto jsou provozovány na samostatném vlákně. Knihovna EmguCV disponuje asynchronními metodami, nabízející programátorovi prostředky, u kterých není třeba zpracovávat složitou synchronizaci vláken, mezi dělením videa na snímky a jeho přehráváním. Problémem u tohoto návrhu je jakým způsobem přenést výsledek rozpoznávání do prezenční vrstvy aniž by bylo třeba předávat parametry vyšším vrstvám. Tento druh předávání zpráv se řeší přes služby, jež mají implementovány kontejnery umožňující účastníkům relace publikovat data nezávisle na vrstvě. Toto oddělení je důležité v modularizovaných aplikacích. V jazyce C# se tyto kontejnery nachází v kompozitní knihovně a umožňují vyhledat konkrétní základ události i pro více účastníků sezení. Více v kapitole **Implementace.** Diagram užití, by neměl chybět u žádného návrhu aplikace, zde nebude uveden, protože je triviální. Uživatel je vázán grafickým prostředím aplikace, které je intuitivní a zabraňuje uživateli v procesech neslučitelných s primárním účelem aplikace.

3.4 Implementace algoritmu detekce a rozpoznání vozidel

Implementace je zde, aby ukázala rozdíl mezi teoretickou průpravou a praktickou implementací. První část této práce se zabývá předzpracováním obrazu, druhá část pak teoreticky provádí čtenáře přes klasifikační algoritmy. V této část se budu detailněji zabývat detekcí a klasifikací v implementační rovině. V blokovém schématu z kapitoly o objektové analýze se jedná o blok *Zpracování snímků* nacházející

se v aplikační vrstvě a v objektovém modelu je jedná o třídu `ImageProcessing`. Blokové schéma na obrázku 26 znázorňuje přes jaké třídy a jakými úpravami musí snímek projít.



Obrázek 26: Blokový diagram algoritmu detekce a klasifikace z hlediska tříd

3.4.1 Objekt pro práci s obrazovými daty

Ačkoli je možné vytvořit obrazový objekt voláním metody `cvCreateImage` ze třídy `CvInvoke`, jenž obsluhuje přímé spojení s metodami v knihovně OpenCV, je vhodnější vytvořit objekt obrázku přes konstrukci `Image<TColor, TDepth>`. Tato konstrukce zahrnuje několik výhod.

1) Paměť je automaticky uvolňována *garbage collectorem*
2) Třída obsahuje několik pokročilých metod, které nejsou v OpenCV k dispozici jako například generické operace nebo konverze na bitmapu.
3) Třídu lze přezkoumat ve *vizualizaci debugeru*.

Prvním generickým parametrem třídy `Image.cs` je upřesnění barvy typu obrázku. Podporované jsou všechny známé barevné prostory (Rgb, Hsv, Lab, atd). Druhým generickým parametrem je hloubka což upřesňuje, jakým způsobem budou v aplikaci reprezentovány jednotlivé pixely obrázku (Byte, Double, Int16/32, atd.).

Používáním této struktury vznikají jisté problémy, které je třeba anulovat nebo se jim přizpůsobit. Prvním takovým problémem, který se jeví, jako triviální jsou otočené souřadnice obrázku. Obecně se udává jako první parametr šířka obrázku a následně výška. Pole uchovávající hodnotu pixelů, má tyto dva parametry otočené a prvním prvkem je tedy výška. Algoritmy, jejichž princip je závislý na procházení

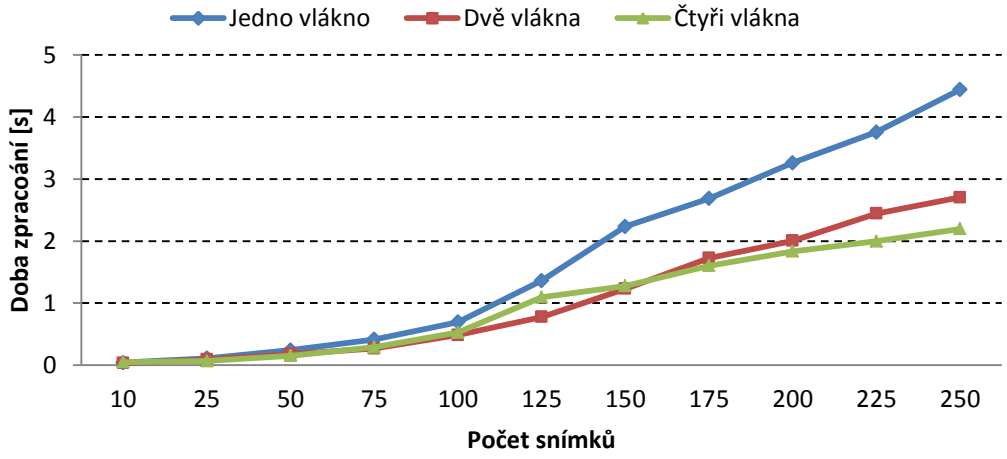
dvourozměrného pole do šířky nebudou efektivně plnit svůj účel, protože budou daný segment procházet do hloubky.

3.4.2 BackgroundSubstraction.cs

Základem této třídy jsou metody pro substrakci pozadí popsané v kapitole 1.2. Každý snímek procházející kaskádou algoritmů je nejprve použit pro vytvoření modelu prostředí. Objekt reprezentující aktuální snímek (definován jako *Image*) je uložen do generické kolekce o *N* prvních. Kolekce je implementována jako fronta, aby byl model prostředí variabilní vůči změně scény. Veškeré úkony jsou realizovány na stejné vrstvě aplikace, aby došlo k úspoře výpočetního času. Celá kolekce je následně předána uživatelem zvolené metodě. Konstrukce metod je přibližně stejná. Vždy je třeba postupně projít jednotlivé obrazové elementy, které slouží dle algoritmu k dalšímu zpracování.

Implementace substrakčních metod provázela řada neduhů citlivě zpomalující celý proces. Jedním z těchto problémů bylo dynamické načítání nastavení z *XML (Extensible Markup Language)* parametrů uložených ve struktuře aplikace. Aby byla reakce algoritmu na uživatelův podmět zanedbatelná, probíhala aktualizace proměnných závislých na nastavení při načítání každého obrazového bodu. Načtení hodnoty z *XML* souboru se jevilo jako triviální proces, ale ve výsledku byl celý algoritmus zpožděn až pětinásobně oproti načítání nastavení pouze při aplikaci vybrané metody. Tato skutečnost byla objevena při využití analyzačního software třetí strany *ANTS Performance profiler*, který měří procesorový čas u každého řádku kódu. Přistupování k obrazovým elementům je implicitně prováděno pomocí ukazatelů, díky čemuž jsou operace nad objekty `Image` velmi rychlé. Substrakce pozadí je pro celý proces klasifikace klíčovým prvkem a proto bylo potřeba optimalizovat proces tak, aby probíhal v co nejkratším možném čase. Toho bylo docíleno zejména zpracováním jednoho snímku více vlákny, jejichž výsledky jsou spojeny synchronizačním primitivem *Bariéra*. Testovány byly varianty za použití jedno, dvou a čtyř vláken. Výsledky jsou promítnuty v grafu 1. Zaznamenána je pouze metoda generující pozadí pomocí průměru, protože ostatní metody jsou principiálně velmi podobné, takže výsledek vyplívající z grafu by se lišit pouze výjimečně. Referenční snímky mají rozměr 634×340 pixelů. Varianta obsluhy obrazu jedním vláknem byla zjevně nejpomalejší, jelikož se prochází celá struktura obrázku po jednotlivých elementech. Rozdělení obrazu na dvě poloviny kde každé vlákno obstarává jednu z těchto částí, se později

ukázalo jako nejstabilnější optimalizační zpracování snímku, protože dosahuje lepších výsledků než metoda s jedním vláknem a zároveň režie obsluhy vláken není tak náročná jako v případě použití čtyř vláken. Použití čtyř vláken je vázáno obsluhou pouze jednoho z kvadrantů obrázku. Tato metoda dosahuje lepších výsledků než oba předchozí způsoby, ale průběžné testování ukázalo vysokou míru nestability při dlouhodobém chodu aplikace.



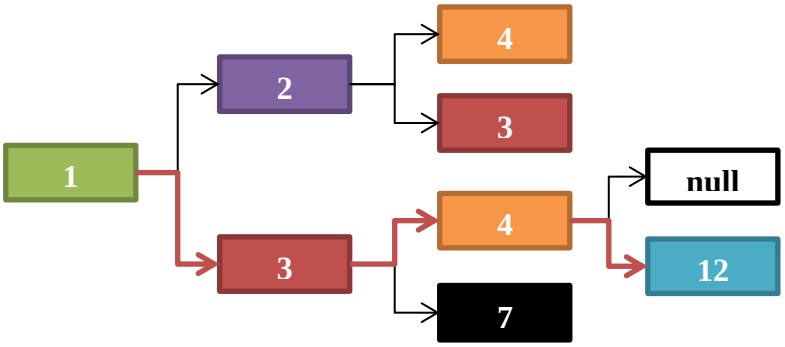
Graf 1: Doba zpracování sady snímků v závislosti na počtu použitých vláken

3.4.3 MotionRecognize.cs

Tento modul respektive třída zajišťuje detekci pohybu ve scéně. Použité řešení je kombinací několika přístupů. Snímek je referenčně odeslán metodě pro vytvoření absolutní diference, vznikající odečtením modelu prostředí s aktuálním snímkem. Absolutní diference zobrazuje pouze objekty, které nejsou součástí statické scény v našem případě automobily. Při prahování vzniká velké množství mylných detekcí, způsobených nedokonalým modelem prostředí. Nabývají charakter necelistvých oblastí a jsou ze snímku odebrány pomocí morfologické metody *otevření*. To je zahrnuto v základních operacích nad obrázky v knihovně EmguCV. Následně je aplikován algoritmus barvení oblastí s maskou zahrnující do algoritmu šest okolních bodů. Jeho rychlost je přímo úměrná velikosti přepisovací tabulky tedy druhým průchodem přes obrazové elementy viz kapitola 1.4.

Aplikací algoritmu barvení oblastí vzniká stejný problém jako u substrakce pozadí. Jeho implementace nesmí zabírat mnoho procesorového času, aby nedošlo ke zpomalování algoritmu. U sekvencí, kdy se v jednom časovém okamžiku vyskytují ve scéně několik pohyblivých objektů se složitou konturou, dochází k velkému nárůstu obsahu tabulky ekvivalence barev, kterou je třeba redukovat. Bez této redukce není

algoritmus zcela efektivní. Redukce přepisu barev je založena na jejich vzájemné závislosti. Závislostí barev je myšlen ukazatel v paměti na další barvu v tabulce ekvivalence určující jejího následníka. Obrázek 27 vyobrazuje příklad vazeb mezi barvami.



Obrázek 27: Schéma vazeb mezi barvami u algoritmu barvení oblastí

Reference na barevnou složku vyskytující se v tabulce na prvním místě odkazuje na referenci barvy následující. Algoritmus barvení oblastí nepřipouští kolizi s více jak dvěma barvami proto celá kolekce referenčních odkazů nabývá struktury binárního stromu. Redukce spočívá v prohledání binárního stromu do hloubky a vytvoření nové reference, jejíž součástí je pouze počáteční a koncový prvek. Průběžné testy prokázaly, že touto úpravou tabulky ekvivalence barev je možné snížit dobu zpracování snímku z několika sekund na několik milisekund[2]. Dalším významným urychlením algoritmu je použití ukazatelů u přístupu k obrazovému objektu kombinovaný s nezabezpečený kontextem definovaným klíčovým slovem *unsafe*. [15]

3.4.4 HighlightVehicle.cs

Oddělení funkční složky od algoritmicky zaměřeného jádra aplikace, si vyžádalo návrh rozhraní, jehož úkolem je nalezený objekt zvýraznit, čímž pro uživatele vzniká ilustrativní a jasná představa o tom, co se v aktuální scéně právě zpracovává. Na základně dat z vrstvy detekující pohyb vozila, jsou vypočítány parametry, jako jsou těžiště, a obdélníkový obal, které slouží jako objekt zájmu (zvýraznění). Třída `HighlightVehicle.cs` se nachází na stejné úrovni návrhu jako třída určená pro detekci pohybu, tedy na samém dně. S tímto přístupem přichází komplikace v podobě publikace dat, protože standardní předávání upraveného snímku přes delegáty by muselo procházet několika vrstvami modelu, což by značně zpomalovalo celý proces. Problém

[2] http://stackoverflow.com/questions/19618281/how-to-speed-up-connected-component-labeling-second-pass/19708793#19708793

je možné řešit referenčním předáváním objektů. Po upravení snímku je vyvolána událost, jež zapříčiní předání snímku přes delegáta prezenční vrstvě. Není tak porušena základní myšlenka oddělení algoritmického jádra od prezenční vrstvy aplikace. Snímek nikdy neopustí modul, ve kterém byl vytvořen a referenční předání je natolik pružné, že vyhovuje značným nárokům na optimalizaci procesu při detekování pohyblivého objektu.

3.4.5 CaptureVehicle.cs

Algoritmus pro oddělení pohyblivého objektu (vozidla) od scény zajišťuje třída `CaptureVehicle`. Dále v sobě implementuje funkci detekce a eliminace stínů, což přirozeně souvisí s oblastí oddělenou od scény. Před zásahem do obrázku je třeba vyřešit komplikaci vícenásobné detekce stejného automobilu. Detekce pohybu v obraze implementuje pomocnou funkci, která přiřadí každému nalezenému objetu jednoznačný identifikátor. Přes tento identifikátor je možné zpřesnit metodu zabývající se výřezem objektu. Nicméně testováním nastaly situace, kde se identifikátor objektu projevil jako nedostatečný kontrolní mechanismus. Výsledné použité řešení je kombinací jednoznačného identifikátoru a algoritmu zavádějící detekční a registrační linii. Tyto dvě linie jsou od sebe vzdálené tak, aby je vozidlo překonalo během několika snímků. To znamená, že pokud se vozidlo nachází mezi detekční a registrační linií a zároveň podle identifikátoru nebylo zaznamenáno, pak se vyjme ze scény a jeho identifikátor je zaznamenán do seznamu již vyřazených objektů. Tento přístup zaručuje vybrání vždy pouze jednoho snímku každého automobilu. Testování prokázalo vyšší úspěšnost algoritmu s definovanou oblastí, ale i tento návrh má nedostatky. U vozidla pohybující se vysokou rychlostí, může dojít k minutí oblasti detekce což má za následek smazání vozidla ze seznamu detekovaných objektů. Na tento problém lze reagovat zvětšením oblasti detekce. Vyjmutí automobilu je realizováno pomocí obdélníkového obalu zmíněného v předchozí kapitole.

V některých případech bylo detekované vozidlo vyseparováno z obrazu včetně stínu, což je pro klasifikační algoritmus zásadní nedostatek. Eliminace stínu je založena na principu popsaného v kapitole 1.5 tedy posuvné masky. Vybraný snímek je převeden na hrany pomocí Cannyho hranového detektoru čímž je získán snímek hran a ten je následně odeslán ke zpracování rekurzivní metodě. Rekurzivní metody mají nevýhodu v jednoduchém zacyklení při nevyhovující podmínce ukončení cyklu. Z tohoto důvodu je při každé iteraci testován počet průchodů algoritmem, aby nedošlo k vyvolání

výjimky vlivem přítomnosti nekonečného cyklu. Metoda je do jisté míry parametrizována. Uživatel si tak může zvolit mezi standardním zobrazením detekovaného vozidla a mezi zobrazením vozidla bez stínu. Také je možné nastavit parametry masky jako je šířka masky, minimální počet obrazových elementů a také posun masky, což umožňuje uživateli variabilnější přizpůsobení dané scéně.

Aby bylo možné efektivně ladit parametry algoritmu pro detekci vozidla, musí být aplikována zpětná vazba od uživatele jako reakce na výsledky algoritmu, zobrazené prezenční vrstvou. Zde nastává problém, jakým vhodným způsobem lze přesunout generický objekt vyšším rovinám aplikace. Stejný nebo podobný problém je často řešen návrhovým vzorem `EvenAggregator`. Ten umožňuje všem komponentám v kompozitní aplikaci komunikovat volně vázaným způsobem. Implementuje *publish/subscribe* mechanismus událostí, který je daný třídou, používající tento mechanismus publikace dat. Díky tomuto návrhu je minimalizována závislost mezi moduly. Moduly detekující události (*subscribe*), nepotřebují vědět nic o tom, jaké události budou generovány. Jediné co potřebují znát je jejich třída přihlášená k odběru. `EventAggregator` je vhodné využít všude tam kde je velký počet objektů potencionálně generující nějakou událost.

3.4.6 Classify.cs

Třída, u které končí celý proces klasifikace je nazvána `Classify`. Jako vstupní parametr je požadován upravený obrázek vozidla vybraný ze scény s eliminovaným stínem. Dílčí úprava obrázku předchází procesu klasifikace, a tudíž není při vstupu do klasifikační části implementována žádná kontrola vstupu. Druhý parametr slouží pro výběr klasifikačního algoritmu uživatelem a je načítán z XML souboru s výchozím nastavením aplikace. Použitá knihovna EmguCV již implementuje některé klasifikační algoritmy včetně používaného PCA. Před začátkem implementace daných metod byly tyto již hotové algoritmy testovány na menší databázi obrázků. Později se ukázalo, že jsou příliš vázané na metody a třídy používané v kontextu s touto problematikou. Dalším problém se projevil při trénování většího počtu obrázků, kde daná metoda selhala s odkazem na výjimku o nedostatku paměti. Ta byla způsobena zejména výpočtem matice D u PCA algoritmu, protože při použití datového typu *Double* a rozlišení trénovacích obrázků 128×128 pixelů zabírala přibližně 2 GB paměti, což je v jazyce C# respective v jeho vývojovém prostředí maximum možného využití paměti. Tento handicap lze od .NET 4.5 obejít nastavením proměnné

gcAllowVeryLargeObjects v konfiguračním souboru aplikace [32], avšak, tímto způsobem je problém řešitelný pouze při použití 64bit operačního systému, ostatní platformy parametr ignorují. Dalším přípustným řešením je použít datový typ *Float* jehož náročnost na paměť je oproti *Double* poloviční. Všechny operace nad maticemi jsou implementovány podporující pouze datový typ *Double*, který výrazně zpřesňuje všechny výsledky a snižuje možnost výskytu chyby vlivem zaokrouhlování. Mnoho kompilátorů se staví k datovému typu *Float* jako non-strict což způsobuje vyšší rychlost při nakládání s operacemi vyžadující tento datový typ, ale výsledky jsou vzhledem k úloze pochybné. [33] Problém nedostatku paměti lze řešit účinněji uvědoměním, že matice *D* je diagonální maticí. Konverzí její struktury na vektor nám zredukuju vytížení paměti z 2 GB na přibližně 130 KB. Díky této redukci je možné aplikovat jako trénovací vzory s mnohem vyšším rozlišením než je zadáno implicitně algoritmem. PCA algoritmus implementovaný v EmguCV knihovně nepodporuje tento přístup k řešení problému.

Důvody zmíněné v předchozím odstavci daly za vznik implementaci vlastního algoritmu PCA využívající již zmíněnou redukci matice *D* na vektor s tím rozdílem, že metody pracující s tímto vektorem jsou upraveny tak, aby bylo nadále k vektoru přistupováno jako k matici. Matice, které jsou potřeba při výpočtech u klasifikačních algoritmů, byly reprezentovány třídou `Matrix` která je poskytnuta knihovnou *EmguCV* a implementuje mnoho metod pro práci s jejich strukturou.

Algoritmus MACE, který je možné zvolit jako klasifikační algoritmus místo PCA, není v knihovně EmguCV k dispozici, ale jeho implementace je triviální. Jediný problém, který se objevil v průběhu návrhu algoritmu je Diskrétní Fourierovi Transformace (DFT). DFT je součástí algoritmů implementovaných v OpenCV nikoliv v EmguCV což díky konstrukci `CvInvoke` netvoří překážku, ale výsledek této transformace bylo pole, kde sekundární prvky představovaly imaginární jednotku. Výsledkem celého procesu byl velmi znepokojivý obraz korelace, nevypovídající žádnou informaci o třídě neznámého obrázku. Oprava konstrukce volání DFT byla značně komplikována konstrukční definitivou jazyka C++ jenž se mísilo se strukturou C#. Oba implementované algoritmy PCA a MACE obsahují i metody pro případné natrénování jiných vzorů.

Kapitola 4

Výsledky rozpoznávání

Aplikace byla testována na různých videosekvencích za účelem nastavení optimálních parametrů, pro automatickou detekci projíždějících vozidel v obraze. Aby se zabránilo nežádoucímu efektu, kdy klasifikační algoritmy fungují správně pouze na konkrétním záznamu a na ostatních fungují s vysokou chybovostí, jsou testovací sekvence různé od těch, ze kterých byly extrahovány vzory určené k trénování klasifikátorů. Klasifikátory jsou schopny rozdělit projíždějící vozidla do následujících kategorií:

1) Osobní automobily a motocykly
2) Dodávky, pickupy a malé karavany
3) Malé nákladní vozy a robustní karavany
4) Kamiony a autobusy

4.1 Trénovací databáze dat

Výsledky použitých klasifikačních postupů jsou závislé na množině trénovacích dat. Celkem bylo vytvořeno přes 50 videosekvencí v HD rozlišení. Z těchto nahrávek se vybralo 15 videosekvencí s optimální hustotou provozu, které se dále dělí na šest testovacích nahrávek po 30 minutách a dvě nahrávky po 45 minutách zachycující značnou změnu iluminace ve scéně. Další tři nahrávky zachycují extrémní podmínky jako je déšť a soumrak. Zbytek nahrávek slouží pro extrakci vzorů (automobilů) z videa, na kterých jsou natrénovány klasifikátory. Výsledná trénovací množina obsahuje v součtu přes jeden tisíc vozidel v různých světelných podmínkách. Automobily jsou zachyceny ze dvou úhlů, které se liší pouze o několik stupňů. Úhel je dán jízdním pruhem, ve kterém se vozidlo nacházelo při jeho zaznamenání a následném vyjmutí ze scény.

Před započetím trénovací sekvence bylo nejprve nutné sjednotit vlastnosti všech trénovacích vzorů pro zvýšení úspěšnosti při rozpoznávání. První vlastností, kterou bylo potřeba sjednotit je rozlišení jednotlivých snímků. Vozidla se v rámci své váhové kategorie příliš neliší, ale mimo své váhové kategorie je rozdíl značný. Srovnáme-li kontury osobního automobilu a nákladního vozidla je zřejmé, že výsledný vzor obou

vozidel se bude lišit svým tvarem a rozlišením. Zatímco osobní automobil bude spíše nabývat tvaru čtyřúhelníku a nízkého rozlišení, nákladní vozidlo (kamion) bude mít tvar obdélníkový a jeho rozlišení bude převyšovat osobní automobil. Normalizace rozlišení může být aplikována dvěma způsoby. Prvním způsobem je obrázky deformovat geometrickou transformací (obrázek 28a) a druhý způsob je vytvořit ze snímků čtverec doplněním chybějících dat (obrázek 28c). Nezávazné pokusy řešit tuto problematiku ukázaly lepší výsledky u druhé možnosti, a proto byly všechny trénovací snímky transformovány do čtvercového tvaru doplněním chybějících dat. Zbytek normalizačního procesu původně spočíval v převedení vzorů Cannyho hranovým detektorem na obrazy se zvýrazněnými konturami a snížení rozlišení na definovaných 128×128 pixelů. Výsledky klasifikace pak byly ovlivněny rozdílem prahových hodnot Cannyho detektoru hran u vozidel, jež jsou součástí trénovacího procesu a neznámých vozidel což způsobovalo degradaci rozhodovacího procesu a nižší rozpoznávací skóre klasifikátorů. Proto bylo od této modifikace vzoru opuštěno. Výsledek normalizace je možné pozorovat a obrázku 28.

(a) (b) (c)

Obrázek 28: Proces normalizace ilustrovaný na trénovacím snímku nákladního vozidla
(a) geometrická transformace, (b) originál, (c) doplnění prostoru nulami

Změna rozlišení je realizováno metodou nejbližšího souseda, která je z nabízených metod knihovnou EmguCV nejrychlejší a zpracování velkého počtu snímků tak není omezeno složitou interpolací.

4.2 Počet automobilů ve scéně

Testování klasifikačních algoritmů, předchází testování úspěšnosti detekčních algoritmů. Tabulka 2 ukazuje výsledky hodnocení systému pro offline testy na

zmíněných videosekvencí včetně ručně pozorovaných výsledků (reálná data) a srovnání mezi nimi. Detekce probíhala v obou pruzích zároveň na základě detekční linie. Vzhledem k umístění kamery se ve scéně nevyskytují vržené stíny s tendencí odklonění se do jiných drah. Prahové hodnoty byly shodné pro všechny uskutečněné testy.

Tabulka 2: Úspěšnost detekce automobilů v obraze

Videosekvence	Počet vozidel	Detekováno	Úspěšnost [%]	Chyba [%]
V01_30	172	166	96,51	3,49
V02_30	214	205	96,26	3,74
V03_30	220	215	97,73	2,27
V04_30	190	187	98,42	1,58
V05_30	199	184	92,46	7,54
V06_43	329	321	97,56	2,44
V07_45	340	332	97,65	2,35
Průměrné hodnoty [%]			**96,65**	**3,34**
E01_nightfall_rain	169	186	90,86	9,14
E02_rain_day	161	154	95,65	4,35
E03_rain_dark	41	40	97,56	2,44

Z tabulky je patrné, že algoritmy pro detekci pohybu v obraze na základě modelu prostředí mají poměrně vysokou úspěšnost. Model prostředí byl ve všech případech aktualizován metodou mediánu. Ostatní metody nejsou v tabulce zahrnuty, jelikož jejich výsledky byly shodné. Chybné detekce jsou vždy založeny na jednotné situaci. Pokud od sebe vozidla nemají postačující distanci, pak vlivem validačního procesu, kdy je aplikována relace dilatace, dojde ke sjednocení několika entit do jednoho celku, čímž klesá počet detekovaných vozidel.

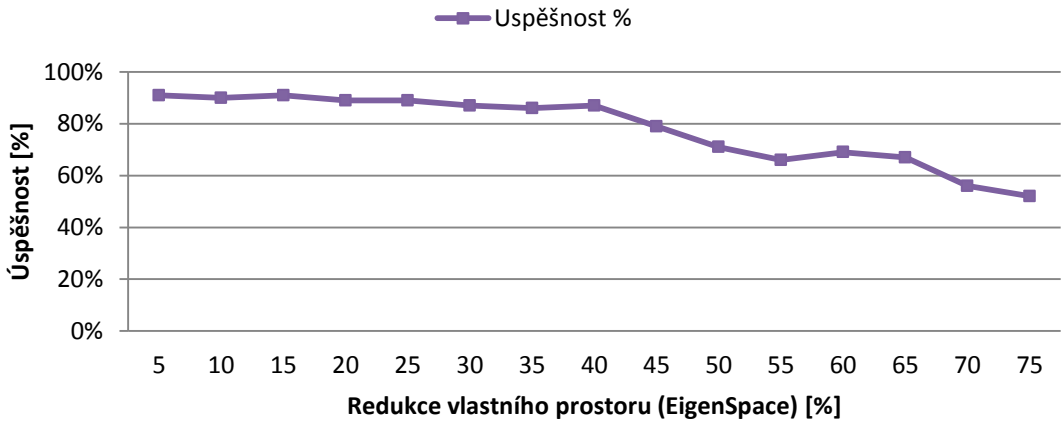
Detekce pohybu za ztížených podmínek (videa s počátečním znakem E) mají též poměrně vysoké rozpoznávací skóre. U těchto videí se vyskytují dva základní problémy. Prvním problémem jsou přední světlomety vozidel, které mají vysoký podíl na výskytu různých reflexních značek. Tyto odlesky mají tendenci kopírovat dráhu vozu, čímž oproti standardním podmínkám způsobují detekci klamných entit. Tento případ byl zaznamenán u videa s názvem **E01_nightfall_rain,** kde je snímána scéna s deštěm a v průběhu natáčení dochází ke stmívání, což umocňuje zmíněnou reflexi. Ostatní videa se ztíženým rozpoznáváním mají úspěšnost nijak nevybočující z průměrných výsledků.

4.3 Klasifikační algoritmy

Stěžejní částí této práce je detekovaná vozidla klasifikovat, což je velmi komplexní proces zpracování informací zakončený rozhodnutím o zařazení neznámého objektu do jedné z předem definovaných tříd [34]. Prvních z testovaných algoritmů je algoritmus PCA.

4.3.1 Výsledky klasifikace algoritmu PCA

Úspěšnost klasifikace algoritmem PCA je podřízené správně zvolenému vlastnímu prostoru, do kterého jsou promítnuty příznaky neznámého obrazu (viz kapitola 2.1). Procesu klasifikace touto metodou předchází série testů, která určí nezávislé komponenty, jež jsou využity k redukci dimenze dat. Redukce dimenze zaručí snížení datového obsahu vlastního prostoru a tím i zrychlí celkový proces klasifikace. Série testů má odhalit o kolik je možné zredukovat dimenzi dat, aniž by byla ovlivněna činnost klasifikátoru. Experiment byl uskutečněn na sérii neznámých obrázků extrahovaných z videosekvence určené právě pro účel redukce prostoru. Postupně bylo testováno přes sto snímků vozidel v různých světelných podmínkách s různou redukcí prostoru. Výsledky rozpoznávání jsou zachycena v grafu 2.



Graf 2: Závislost úspěšnosti rozpoznávání vzoru na redukci vlastního prostoru

Z grafu je patrné, že vlastní prostor algoritmu PCA je možné zredukovat až o 40% se zachováním rozpoznávací schopnosti pohybující se kolem 85%. Pro naše účely zvolíme redukci prostoru o 15%, abychom setrvali nad hranicí 90% rozpoznávacího skóre. Následující tabulka 3 zachycuje úspěšnost rozpoznávání na celé množině testovacích videí. Uvádí výčet detekovaných automobilů a jejich rozpoznání metodou PCA, pro přehled nad problematickými třídami.

Tabulka 3: Detailní přehled klasifikačních tříd s úspěšností rozpoznávání metodou PCA

Vozidlo	Videosekvence						
	V01_30	**V02_30**	**V03_30**	**V04_30**	**V05_30**	**V06_43**	**V07_45**
Osobní	80	92	102	90	86	221	234
Užitkové	11	19	20	16	19	47	31
Nákladní	8	10	12	9	12	7	13
Kamiony	67	85	81	72	67	39	54
∑	**166**	**206**	**215**	**187**	**184**	**314**	**332**
Úspěšnost rozpoznávání jednotlivých tříd metodou PCA							
Osobní	48	78	78	82	78	216	196
Přepravní	7	17	15	10	12	38	24
Nákladní	6	3	9	4	8	5	8
Kamiony	65	84	77	67	62	34	48
∑	**126**	**182**	**179**	**163**	**160**	**293**	**276**
Úspěšnost	**75,90%**	**88,35%**	**83,26%**	**87,16%**	**86,95%**	**93,31%**	**83,13%**

Algoritmus PCA vykazoval ve všech testech vysokou úspěšnost rozpoznávání, které bylo realizováno pomocí Euklidovy normy. Z tabulky 3 je parné, že nejlépe jsou rozpoznávány vozidla kamionové dopravy, což je způsobeno jejich rozdílnou strukturou oproti osobním vozům. Oproti tomu nejhůře se rozpoznávají užitkové vozy, které mají velmi podobný tvar jako osobní automobily. To způsobuje častou chybu klasifikace, kdy je osobní automobil zaměněn s užitkovým vozem. Další chybu zanáší do celého procesu i algoritmus detekce a eliminace stínu, který ne vždy je schopen eliminovat stín na dostatečnou úroveň, která neovlivňuje následný proces klasifikace. Není zde uvedena tabulka výsledků klasifikace pro extrémní případy, jelikož odlesky reflektorů a víření vodní hladiny na vozovce způsobuje značný problém detekce hran vozidla oproti prostředí.

4.3.2 Výsledky klasifikace algoritmu MACE

Dalším z testovaných algoritmů klasifikace vozidel, byl korelační filtr MACE. K posouzení ostrosti vrcholu korelace a tím určení do jaké třídy se zařadí neznámé vozidlo, bylo použito parametru PSR, přičemž definovaná třída vozidla byla přidělena na základě nejvyšší hodnoty tohoto parametru. To znamená, že bylo nutné parametr

PSR vypočítat pro každý filtr zvlášť, což by mohlo mít za následek zpomalení algoritmu. K tomuto jevu při testech nedošlo, jelikož oproti vlastnímu prostoru PCA algoritmu jsou korelační filtry rozměrově v jiné kategorii, protože obsahují pouze $N \times N$ hodnot kde N je velikost strany trénovacích obrázků. V našem případě má korelační filtr pouze 16 384 hodnot. V následující tabulce 4 se nacházejí výsledky rozpoznávání pro algoritmus MACE.

Tabulka 4: Detailní přehled klasifikačních tříd s úspěšností rozpoznávání metodou MACE

Vozidlo	**Videosekvence**						
	V01_30	**V02_30**	**V03_30**	**V04_30**	**V05_30**	**V06_43**	**V07_45**
Osobní	80	92	102	90	86	221	234
Užitkové	11	19	20	16	19	47	31
Nákladní	8	10	12	9	12	7	13
Kamiony	67	85	81	72	67	39	54
∑	**166**	**206**	**215**	**187**	**184**	**314**	**332**
Úspěšnost rozpoznávání jednotlivých tříd metodou MACE							
Osobní	33	19	37	54	49	114	120
Přepravní	2	3	2	2	4	25	12
Nákladní	1	0	2	0	2	4	1
Kamiony	35	42	38	26	39	15	10
∑	**71**	**64**	**79**	**82**	**94**	**158**	**143**
Úspěšnost	**42,77%**	**31,07%**	**36,74%**	**43,85%**	**51,09%**	**50,32%**	**43,07%**

Tabulka 4 vykazuje výrazné zhoršení výsledků oproti algoritmu PCA. Při testech nebyli výjimkou situace, kde jednoznačně definovaný osobní automobil bez známek šumu v pozadí, byl rozpoznán jako kamion. Při následné kontrole PSR parametru bylo zjištěno, že vítězná třída byla určena pouze s malým rozdílem (řádově desetiny). Špatné rozpoznávací skóre je přisuzováno nedostatečné databázi vozidel, která obsahovala 1000 entit, což je dostatečné množství k natrénování algoritmu PCA, ale nikoliv pro algoritmus MACE. Algoritmus při testování vykazoval značnou robustnost vůči změnám osvětlení scény.

4.3.3 Výsledky klasifikace algoritmu SVM

Klasifikátor SVM byl použit s metodou PCA pro extrakci příznaků z dané série trénovacích obrazů. V původním algoritmu PCA se přistupovalo na zařazení do definované třídy na základě metody nejbližšího souseda, jenž byl realizován Euklidovou vzdáleností. V tomto případě se metoda nejbližšího souseda nahrazuje klasifikátorem SVM. Protože klasifikátor SVM je schopen rozeznat pouze dvě třídy (jedná se o binární klasifikátor) byla použita *multiclass* strategie klasifikace. Následující tabulka zaznamenává výsledky tohoto klasifikátoru.

Tabulka 5: Detailní přehled klasifikačních tříd s úspěšností rozpoznávání metodou PCA

Vozidlo	**Videosekvence**						
	V01_30	**V02_30**	**V03_30**	**V04_30**	**V05_30**	**V06_43**	**V07_45**
Osobní	80	92	102	90	86	221	234
Užitkové	11	19	20	16	19	47	31
Nákladní	8	10	12	9	12	7	13
Kamiony	67	85	81	72	67	39	54
∑	**166**	**206**	**215**	**187**	**184**	**314**	**332**
Úspěšnost rozpoznávání jednotlivých tříd metodou SVM							
Osobní	74	86	95	83	84	209	225
Přepravní	7	12	12	12	12	32	18
Nákladní	5	6	8	7	6	4	8
Kamiony	64	81	80	70	63	37	54
∑	**150**	**185**	**195**	**172**	**165**	**282**	**305**
Úspěšnost	**90,36%**	**89,90%**	**90,69%**	**91,97%**	**89,67%**	**89,80%**	**91,86%**

Výsledky klasifikátoru SVM jsou jednoznačně lepší než v případě použití metody nejbližšího souseda. Nedochází zde tak často k zaměňování osobního vozidla s užitkovým i přes svou značnou strukturální i parametrickou podobnost. To je dáno zejména možností natrénovat klasifikátor na obrazových datech čímž vzniká robustnější metoda klasifikace než v případě nejbližšího souseda.

4.4 Detekce barvy automobilu

Výsledky detekce barvy automobilů jsou zaznamenány v tabulce 6. Celý algoritmus se potýká se zásadním problémem volby statického regionu, ve kterém se nachází největší množství barevného přispění odpovídající reálné barvě automobilu. Osobní vozidla jsou většinou jednobarevná, až na různé výjimky, kde si majitelé vozu barevnou složku svého vozu lokálně modifikují. Větší problémem jsou pak kamióny, které nemají shodnou barvu přední konstrukce vozu s návěsem. Pak je určení barvené složky čistě otázkou lokace regionu, ze kterého je extrahována barva.

Tabulka 6: Úspěšnost rozpoznání barevné složky vozidla

Videosekvence	Počet vozidel	Detekováno	Úspěšnost [%]	Chyba [%]
V01_30	166	72	43,37	56,63
V02_30	205	86	41,95	58,05
V03_30	215	78	36,27	63,73
V04_30	187	63	33,68	66,32
V05_30	184	69	37,50	62,50
V06_43	321	95	29,59	70,41
V07_45	332	104	31,32	68,68
			36,24	**63,76**

Jak je vidět z tabulky 6, algoritmus detekce barvy nemá vysoké rozpoznávací skóre ani u jedné videosekvence, právě z důvodu umístění statického regionu extrakce barvy. Dalším problémem při extrakci barvy je úhel natočení záznamového zařízení, jelikož některá vozidla mají navrženu přední část s robustními světlomety, které zasahují do regionu extrakce barvy a tím deformují barevný prostor, jenž je rozpoznáván jak je vidět na obrázku 29.

Obrázek 29: Obrázek osobního automobilu, jehož přední světla deformují prostor extrakce barvy

4.5 Rychlost vozidla

Měření rychlosti vozidel bylo realizováno na speciální sadě videí, které byly pořízeny pouze pro tento účel. Jedná je a velmi krátké sekvence vylučující jakékoliv jiné pohyblivé objekty v obraze (v měřeném směru). Vyskytuje se na něm domluvené vozidlo, které zachovává přesně definovanou rychlost, kontrolovanou dle tachometru zabudovaného uvnitř vozidla. Testovaný subjekt udržuje rychlosti 90, 100, 110, 120 a 130 km/h se záměrem otestovat měření rychlostí navrženým algoritmem díky předem známé konstantě. Jednotlivé rozdíly oproti reálně a naměřené rychlosti jsou zaznamenány v tabulce 7.

Tabulka 7: Rozdíly mezi reálnou a měřenou rychlostí vozidla

Videosekvence	Reálná rychlost [Km/h]	Změřená rychlost [Km/h]	Rozdíl [Km/h]	Chyba [%]
S01_90km	90	87,62	-2,38	2,14
S02_100km	100	94.15	-5.85	5,58
S03_110km	110	115,42	+5,42	5,96
S04_120km	120	124,69	+4,69	5,62
S05_130km	130	126,43	-3,57	4,64
				4,79

Měření rychlosti vozidla bylo poměrně úspěšné, jelikož je při měření zanedbána lineární perspektiva. Testovanému vozidlu nebylo povoleno vyvinout vyšší rychlost než je ustanovená zákonem pro dálniční spoje, ale lze předpokládat, že se zvyšující se rychlostí se zvyšuje i chyba měření. Tento fakt je dát pozorováním, kdy ve výstupní aplikaci byly zaznamenány případy vozidel jedoucí například 260 Km/h. Tato rychlost není nereálná, ale vzhledem k přítomnosti dálničních kamer na měřeném úseku je tato rychlost nepravděpodobná.

Závěr

V rámci této práce byl navržen systém pro automatickou detekci a následnou klasifikaci pohybujících se vozidel v sekvenční scéně pořízené záznamových zařízením s HD rozlišením. Tento systém umožňuje detekovat vozidla v reálném čase. Všechny mnou navržené a implementované algoritmy byly optimalizovány, aby při úbytku počtu snímků přehrávaného videa došlo k dorovnání okolností potřebných pro zpracování a přehrávaný záznam byl stále plynulý. Nicméně v této části se nachází prostor pro vylepšení. Pro zrychlení některých výpočetně náročnějších algoritmů bylo přistoupeno k paralelizaci zaručující zpracování v sublineárním čase.

Detekce pohybu v obraze je koncipováno vytvořením modelu prostředí, který je odečten od aktuálního snímku s cílem detekovat nestacionární objekty. Algoritmy pro aktualizaci modelu prostředí byly testovány na jednotlivých videosekvencích. Ukázalo se, že nejvhodnější je použít algoritmy založené na zpracování více následujících snímků, protože jsou robustní vůči změnám osvětlení snímané scény. Testy detekce pohybu v obraze prokázali vysokou úspěšnost pohybující se kolem **96,65%.** Chybné detekce byly většinou způsobeny spojením několika entit do jednoho celku.

Jako klasifikační algoritmy byly zvoleny *PCA*, *MACE* a *SVM*. *PCA* je algoritmus pro redukci dimenze a extrakci příznaků, který je schopen určit třídu na základě metody nejbližšího souseda. *MACE* je oproti této metodě založen na korelaci neznámého obrazu s filtrem vytvořený z jedné třídy trénovacích vzorů. Třída se určuje výpočtem parametru *PSR*. *SVM* je klasifikátor vyžadující ke své činnosti metodu pro extrakci příznaků, které následně porovnává. Klasifikátor *PCA* dosáhl nejnižší úspěšnosti u prvního testovaného videa **75,90**% a nejvyšší úspěšnosti u videa šestého a to **93.31%**. Výsledky klasifikátoru SVM jsou u každého testovaného videa přibližně o **3 – 5%** lepší než u metody nejbližšího souseda. Algoritmus *MACE* oproti těmto výsledkům selhává a jeho rozpoznávací skóre se pohybuje na hranici **45%**. To je jednak způsobeno malou databází trénovacích vzorků, ale také obecně v přístupu algoritmu k detekci neznámých obrazů. Algoritmy založené na *MACE* jsou určeny především pro rozpoznávání obrazů, které byly součástí trénovací množiny, jak tomu je například při snímání otisků prstů, kde cílem je identifikace osoby jejíž otisk je známý.

Rozpoznání barvy vozu skončilo s úspěšností **36%** a bylo by vhodné navržený algoritmus dále vyvíjet. Měření rychlosti vozidel se zanedbanou lineární perspektivou vykazuje chybovost kolem **4,78%** při běžných rychlostech.

Literatura

[1] ZHANG, Guohui, Ryan AVERY a Yinhai WANG. A Video-based Vehicle Detection and Classification System for Real-time Traffic Data Collection Using Uncalibrated Video Cameras. *Smart transportation application and research* [online]. 2007 [cit. 2014-05-09]. Dostupné z: http://www.uwstarlab.org/___OLD_SITE/STARLab_Papers/2007_TRB_A%20Videobased%20Vehicle%20Detection%20and%20Classification%20System%20for%20Real-time%20TrafficData%20Collection.pdf

[2] JAZAYERI, Amirali, Hongyuan CAI, Jiang ZHENG a Mihran TUCERYAN. Vehicle Detection and Tracking in Car Video Based on Motion Model. *Computer Information Science IUPUI* [online]. June 2011 [cit. 2014-05-09]. Dostupné z: http://cs.iupui.edu/~jzheng/IEEEITS-print.pdf

[3] LUO, Jinman a Juan ZHU. Improved Video-Based Vehicle Detection Methodology. [online]. 2013 [cit. 2014-05-09]. Dostupné z: http://wenku.baidu.com/view/63ee6b63ddccda38376bafd5.html

[4] JOSHI, Kinjal a Darshak THAKORE. A Survey on Moving Object Detection and Tracking in Video Surveillance System. INTERNATIONAL JOURNAL OF SOFT COMPUTING AND ENGINEERING. *International Journal of Soft Computing and Engineering(TM)* [online]. July 2012 [cit. 2014-05-09]. Dostupné z: http://www.ijsce.org/attachments/File/v2i3/C0675052312.pdf

[5] Metody rozpoznání objektů v obrazu. *Úlohy pro praktickou výuku zpracování obrazových dat* [online]. 2011 [cit. 2014-05-09]. Dostupné z: http://www.fbmi.cvut.cz/files/predmety/3528/public/Metody%20rozpozn%C3%A1n%C3%AD%20objekt%C5%AF%20v%20obrazu.pdf

[6] PALEČEK, Karel. *Rozpoznávání osob na základě detekovaného obličeje*. Liberec, 2007. Bakalářská práce. Technická univerzita v Liberci. Vedoucí práce Doc. Ing. Josef Chaloupka, Ph.D

[7] KŘÍŽ, Karel. *Detekce dopravních prostředků a vyhodnocování jejich stavů z vydeozáznamů křižovatek* [online]. Brno, jaro 2013 [cit. 2014-05-09]. Dostupné z: http://is.muni.cz/th/395613/fi_m/xkriz8_DP.pdf. Diplomová práce. Masarykova Univerzita. Vedoucí práce David Svoboda.

[8] What is Deinterlacing? Facts, solutions, examples. *What is Deinterlacing* [online]. 2009 [cit. 2014-05-10]. Dostupné z: http://www.100fps.com/

[9] JUNG, J.H a S.H HONG. *Deinterlacing method based on edge direction refinement using weighted maximum frequent filter*. New York, NY, USA, 2011.

[10] PICCARDI, Massimo. Background subtraction techniques: a review. *University of Technology, Sydney* [online]. April 15, 2004 [cit. 2014-05-09]. Dostupné z: http://www-staff.it.uts.edu.au/~massimo/BackgroundSubtractionReviewPiccardi.pdf

[11] CHEN, Ssu-Wei, Luke WANG a Jen-Hong LAN. Moving Object tracking Based on Background Subtraction Combined Temporal Difference. *International Conference on Emerging Trends in Computer and Image Processing* [online]. Dec., 2011 [cit. 2014-05-09]. Dostupné z: http://psrcentre.org/images/extraimages/1211649.pdf

[12] ALAWI, Mahmoud, Othman KHALIFA a Rafiqul ISLAM. Performance Comparison of Background Estimation Algorithms for Detecting Moving Vehicle. DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, International Islamic University Malaysia. *World Applied Sciences Journal 21 (Mathematical Applications in Engineering)* [online]. 2013 [cit. 2014-05-09]. Dostupné z: http://www.idosi.org/wasj/WASJ21(mae)13/20.pdf

[13] HLAVÁČ, Václav a Miloš SEDLÁČEK. Zpracování signálu a obrazu: Pracovní verze skripta v tisku pro studenty FEL ČVUT. *Neuron* [online]. 7. prosince 1999 [cit. 2014-05-10]. Dostupné z: http://neuron.tuke.sk/pluchta/Pocitacove%20Videnie/Prednasky/NIECO/HLAZSO.PDF

[14] FISHER, R., S. PERKINS, A. WALKER a E. WOLFART. Connected Components Labeling. *Image Analysis: Connected Components Labeling* [online]. 2003 [cit. 2014-05-09]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm

[15] SAMET, Hanan. Connected Component Labeling Using Quadtrees. UNIVERSUY OF MARYLAND, College Park, Maryland. *UMD Department of Computer Science* [online]. 3, July 1981 [cit. 2014-05-09]. Dostupné z: http://www.cs.umd.edu/~hjs/pubs/SametJACM81.pdf

[16] Binary Image Analysis: Connected Components Labeling. *University of Washington Computer Science & Engineering* [online]. Mar 2000 [cit. 2014-05-09]. Dostupné z: http://courses.cs.washington.edu/courses/cse373/00au/chcon.pdf

[17] DHARPURE, Jaydeo, Madhukar POTDAR a Manoj PANDYA. Counting Objects using Homogeneous Connected Components. *International Journal of Computer Applications* [online]. February 2013 [cit. 2014-05-09]. Dostupné z: http://research.ijcaonline.org/volume63/number21/pxc3885585.pdf

[18] AVERY, Ryan, Guohui ZHANG, Yinhai WANG a Nancy NIHAN,. An Investigation into Shadow Removal from Traffic Images. *Smart transportation application and research* [online]. November 15, 2006 [cit. 2014-05-09]. Dostupné z:http://www.uwstarlab.org/STARLab_Papers/2007_TRB_An%20Investigation%20into%20Shadow%20Removal%20from%20Traffic%20Images.pdf

[19] HOLČÍK, Jiří. Analýza a klasifikace dat. *IBA: Institut biostatistiky a analýz Lékařské a Přírodovědecké fakulty Masarykovy univerzity: analýza klinických a biologických dat, vývoj softwaru, klinické registry* [online]. Březen 2012 [cit. 2014-05-09]. Dostupné z: http://www.iba.muni.cz/res/file/ucebnice/holcik-analyza-klasifikace-dat.pdf

[20] SMITH, Lindsay. A tutorial on Principal Components Analysis. *A tutorial on Principal Components Analysis* [online]. February 26, 2002, č. 1, s. 26 [cit. 2014-05-09]. Dostupné z: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[21] HOLAND, Steven. Principal component Analysis. *UGA Stratigraphy Lab* [online]. May 2008 [cit. 2014-05-09]. Dostupné z: http://strata.uga.edu/software/pdf/pcaTutorial.pdf

[22] TIPPING, Michael a Christopher BISHOP. Probabilistic principal component analysis. MICROSOFT RESEARCH. *Microsoft Research* [online]. 2014 [cit. 2014-05-09]. Dostupné z: http://research.microsoft.com/pubs/67218/bishop-ppca-jrss.pdf

[23] TYAMAGUNDLU, Divya, Kaushal PATEL a Keshav SESHADRI. Analysis of Face Recognition Algorithms for Blurred and Partially Occluded Images. *CMU Contributed Webserver* [online]. 2011 [cit. 2014-05-09]. Dostupné z: http://www.contrib.andrew.cmu.edu/~kseshadr/Pat_Rec_Paper.pdf

[24] SAVVIDES, Marios, Vijaya KUMAR a Pradeep KHOSLA. Face Verification using Correlation Filters. *Scourge.fr* [online]. 2011 [cit. 2014-05-09]. Dostupné z: http://scourge.fr/mathdesc/documents/facerecog/mace.pdf

[25] BHAGAVATULA, Vijayakumar. Correlation Filters for Face Recognition. *West Virginia University* [online]. 2014 [cit. 2014-05-09]. Dostupné z: http://www.wvu.edu/~facerecognition/2D%20Matching%20Rep/VJ%20Kumar.pdf

[26] ZHU, Xiangxin, Shengcai LIAO, Zhen LEI, Rong LIU a Stan LI. Feature Correlation Filter for Face Recognition. *Center for Biometrics and Security Research & National Laboratory of Pattern Recognition*, [online]. 1995 [cit. 2014-05-09]. Dostupné z: http://www.ics.uci.edu/~xzhu/paper/FCF-ICB07.pdf

[27] BOSWELL, Dustin. Introduction to Support Vector Machines. *Yaser S. Abu-Mostafa* [online]. August 6, 2002 [cit. 2014-05-09]. Dostupné z: http://www.work.caltech.edu/~boswell/IntroToSVM.pdf

[28] Support Vector Machines. *TheCAT: Web Services Overview* [online]. 2009, Wednesday, 09 September 2009 [cit. 2014-05-09]. Dostupné z: http://web.cecs.pdx.edu/~mm/AIFall2011/SVMs.pdf

[29] BEN-HUR, Asa a Jason WESTON. A User's Guide to Support Vector Machines. DEPARTMENT OF COMPUTER SCIENCE. *PyML: machine learning in Python* [online]. 2010 [cit. 2014-05-09]. Dostupné z: http://pyml.sourceforge.net/doc/howto.pdf

[30] BILLETT, James. *A Testbed for Support Vector Machine software* [online]. Australia, 2005 [cit. 2014-05-10]. Dostupné z: http://www.csse.monash.edu.au/hons/se-projects/2005/James.Billett/thesis.pdf. Bachelor. Clayton School of Information Technology Monash University. Vedoucí práce Dr. David Albrecht.

[31] WEISSTEIN, Eric. Convex Hull. *MathWorld* [online]. 2014 [cit. 2014-05-09]. Dostupné z: http://mathworld.wolfram.com/ConvexHull.html

[32] <gcAllowVeryLargeObjects> Element. MICROSOFT. *Developer Network MSDN* [online]. 2014 [cit. 2014-05-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/hh285054(v=vs.110).aspx

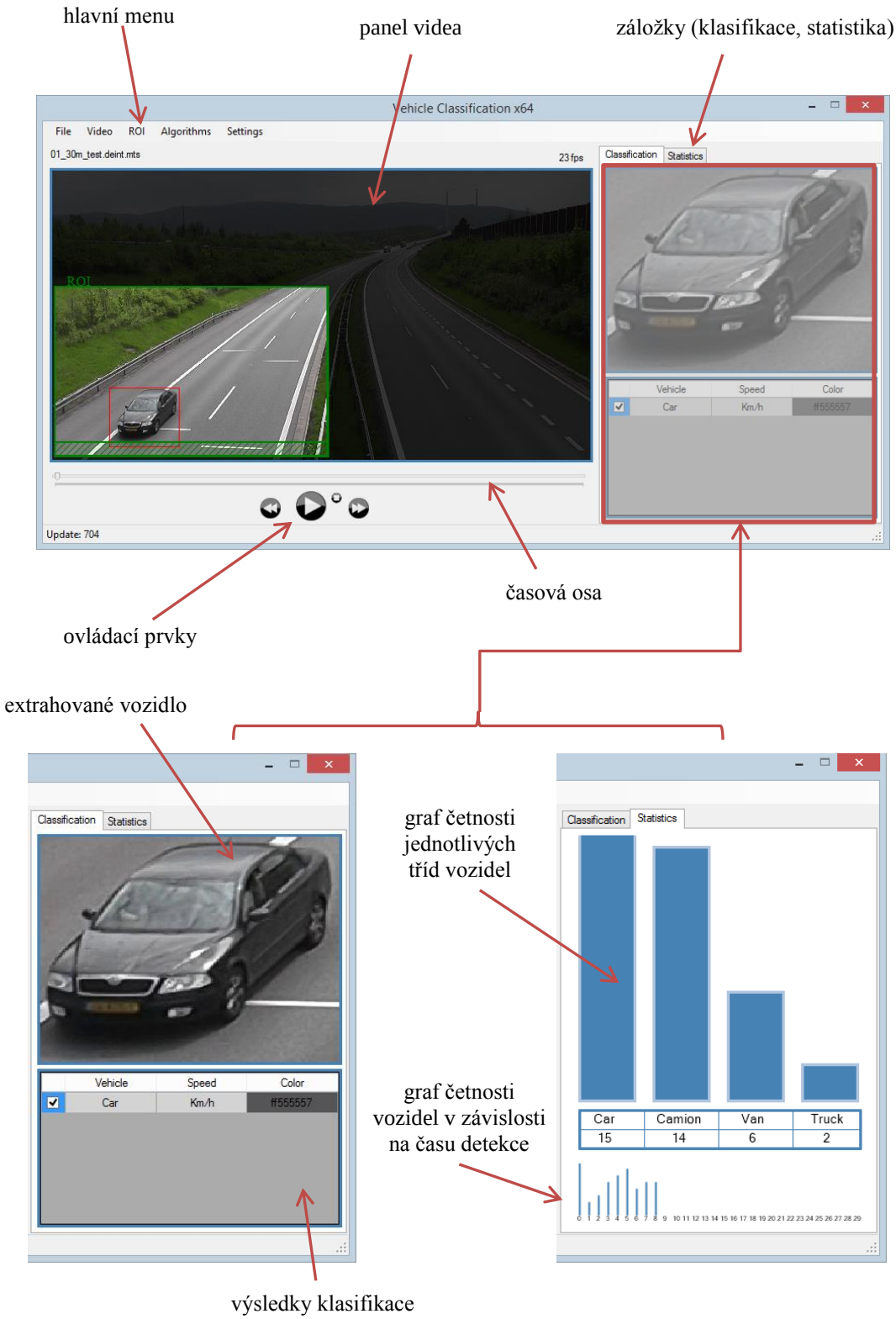
[33] What Every Computer Scientist Should Know About Floating-Point Arithmetic. SUN MICROSYSTEMS. *A P P E N D I X D - What Every Computer Scientist Should Know About Floating-Point Arithmetic* [online]. 2005 [cit. 2014-05-09]. Dostupné z: http://docs.oracle.com/cd/E19422-01/819-3693/ncg_goldberg.html

[34] NOUZA, Jan. Metody rozhodování a klasifikace: Přednáška č.1. *Laboratoř počítačového zpracování řeči* [online]. [cit. 2014-05-09]. Dostupné z:http://itakura.ite.tul.cz/jan/MRK/prednaska1.pdf

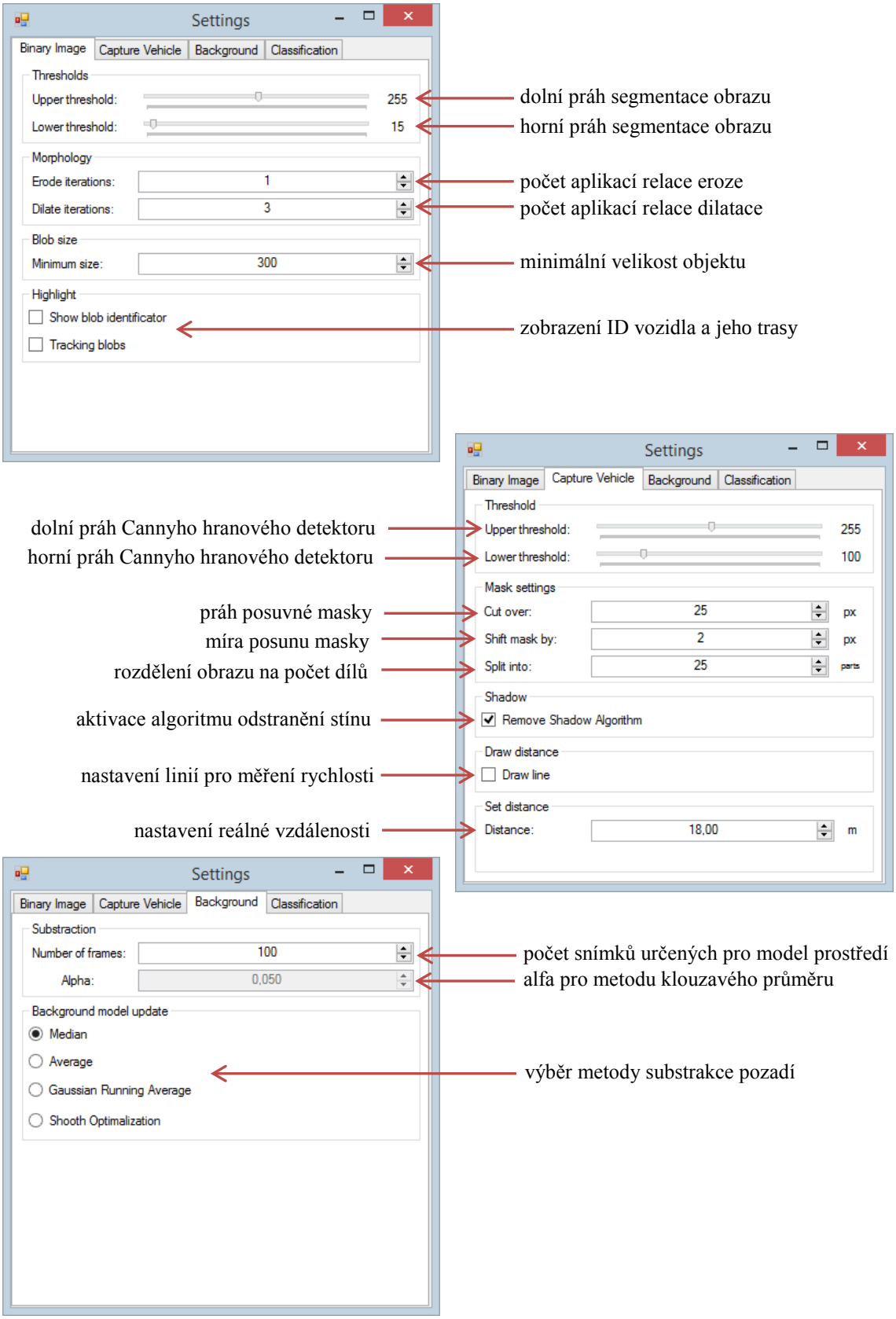
[35] Main Page: EmguCV. *Emgu CV: OpenCV in .NET (C#, VB, C++ and more)* [online]. 31 August 2013 [cit. 2014-05-10]. Dostupné z: http://www.emgu.com/wiki/index.php/Main_Page

[36] HUA, Hong a Kobus BARNARD. UNIVERSITY OF ARIZONA. *A fast connecting component labeling algorithm and its application to real-time pupil detection*. Tucson, USA, 12 September 2008.

GUI aplikace



Nastavení aplikace



Obsah přiloženého DVD

- Text diplomové práce ve formátu PDF
- Testovací videosekvence
- Zdrojové kódy aplikace pro automatickou detekci a rozpoznávání vozidel