VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

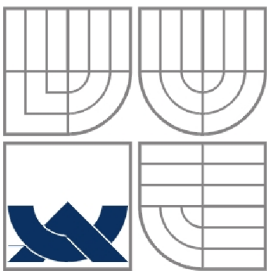
DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEO SEKVENCI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

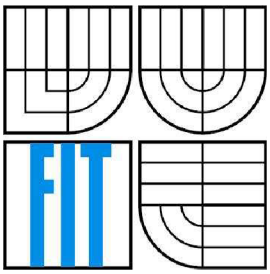
AUTOR PRÁCE
AUTHOR

Bc. ZDENĚK HOCHMAN

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLEDOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEOSEKVENCI

MOVING OBJECTS DETECTION IN VIDEO SEQUENCES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ZDENĚK HOCHMAN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL

BRNO 2010

Abstrakt

Práce se věnuje detekci pohybujících se objektů ve video sekvenci. Zabývá se hlavně problematikou detekce pohybu, určení místa pohybu, nalezení jednotlivých objektů a sledováním průchodu objektu scénou. Následně se zaměřuje na odstranění stínů. V práci je představena metoda detekce pohybu založená na Local Binary Pattern, která spolu s diferenční metodou nad barevným prostorem HSV umožňuje rychlou a přesnou detekci pohybu.

Abstract

This thesis deals with moving objects detection in video sequences. The principal aim of such detection is to detect and locate motion in the image, separate individual objects, and track these objects. Subsequently, to eliminate shadows, the paper introduces method of motion detection based on Local Binary Patterns together with differential method above the HSV color space. The proposed method provides rapid and accurate movement detection in video sequences.

Klíčová slova

Detekce pohybujících se objektů, Local Binary Patterns, HSV barevný prostor, stín, histogram, tracking,C++, video sekvence.

Keywords

Detect moving objects, Local Binary Patterns, HSV color space, shadow, histogram, tracking, C++, video sequence.

Citace

Hochman Zdeněk: Sledování pohybujících se objektů ve video sekvenci, diplomová práce, Brno, FIT VUT v Brně, 2010

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Španěla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zdeněk Hochman
18.5.2010

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Španělovi za cenné rady, doporučení a připomínky. Děkuji také mamince a svým přátelům za poskytnutou pomoc při tvorbě této práce.

Obsah

1 Úvod

Dříve než se člověk naučil mluvit a psát, pozoroval svoje okolí. Pomocí svého zraku zkoumal okolí a byl schopen identifikovat pohybující se objekty. Kromě zachycení pohybu se u lidí vyvinula schopnost odhadnout rychlost a směr pohybu. Tyto schopnosti v pradávných dobách pomáhaly lidem při lovu. Ale i v dnešních dobách jsou pro nás tyto schopnosti důležité.
Mnoho činností, které dříve vykonával člověk, jsme dnes schopni nahradit nebo velmi usnadnit pomocí počítačů. Některé činnosti se nám ale zatím nepodařilo automatizovat na takové úrovni, aby nebyla nutná častá kontrola výsledků ze strany člověka. Takovýmto problémem je i sledování pohybujících se objektů.

Sledování pohybujících se objektů nachází široké uplatnění. Může být využito při ochraně objektů, sledování hustoty provozu na silnicích, jako vstup dalších algoritmů zpracovávajících obraz (například identifikace osob, detekce rychle jedoucích automobilů) a v mnoha dalších činnostech.

Sledování pohybujících se objektů je stále otevřené téma. Doposud nebyl nalezen univerzální algoritmus, který by fungoval v různých prostředích, za různých podmínek a detekoval všechny možné pohybující se objekty. Protože nelze vytvořit obecný detektor, rozhodl jsem se zaměřit na sledování pohybujících se automobilů.

Ve své diplomové práci se pokusím určit pixely, ve kterých dochází k pohybu. Tato detekce je velmi obecná a dá se využít pro jakoukoliv třídu objektů. Tyto pixely poté seskupím do jednotlivých objektů. Nakonec se pokusím sledovat průchod jednotlivých objektů scénou (angl. tracking). Při sledování průchodu objektů scénou, již využívám vlastností specifických pro automobily.

V této diplomové práci navazuji na svůj semestrální projekt a bakalářskou práci. Zkušenosti, které jsem získal při tvorbě semestrálního projektu a bakalářské práce, mně poskytly teoretický základ pro vypracování této práce.

Ve druhé kapitole uvádím metody, které využívám při předzpracování obrazu před samotnou detekcí pohybu. Kapitola třetí se zabývá metodami pro detekci pohybu. V první části této kapitoly uvádím přehled metod, které se dají použít k detekci pohybu. V další části podrobně rozebírám princip detekční metody využívající Local Binary Patterns. Ve čtvrté kapitole popisuji princip čtyř přístupů pro odstranění stínů. V kapitole páté popisuji, jak jsem postupoval při návrhu aplikace. Kapitola šestá se věnuje implementaci aplikace. V závěrečné kapitole představuji a hodnotím výsledky své práce.

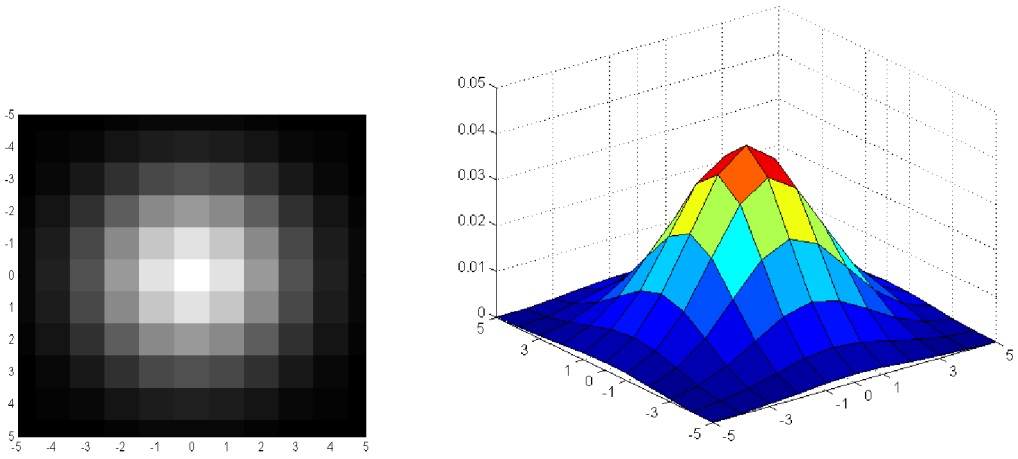
2 Předzpracování obrazu

Před samotnou detekcí pohybu je vhodné předzpracovat obraz. V této kapitole popíši techniky a jejich vliv na obraz, které využívám pro předzpracování obrazu ve své aplikaci.

2.1 Odstranění šumu Gaussovým filtrem

Gaussův filtr je lineární konvoluční filtr, který má jádro definované vztahem:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\ \mathrm{e}^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \qquad (1)$$



Obrázek 1: Gaussova funkce pro σ=2. Převzato z [8]

kde parametr σ určuje špičatost filtru. Čím je filtr špičatější, tím méně se ve výsledku uplatňují pixely vzdálenější od středu.

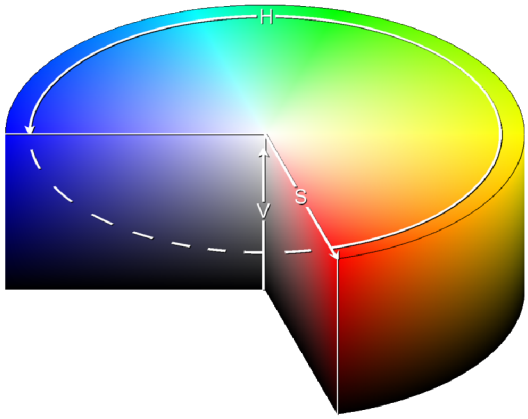
Gaussův filtr distribuuje informaci z okolí do pixelu. Filtr způsobí rozmazání obrazu, ale zároveň pomůže redukovat šum. Reálný šum se dá v mnoha případech aproximovat Gaussovým šumem. Takže pokud použijeme Gaussův filtr na obraz, ve kterém je obsažen šum, dojde k redukci šumu v obraze.

2.2 Podvzorkování obrazu

Podvzorkování je operace, při které z oblasti původního obrazu o rozměru n krát n převedu na jeden pixel nového obrazu. Existuje více možností, jak určit hodnotu pixelu ve výsledném obraze. Mohu počítat průměrnou hodnotu pixelu v oblasti, vybrat medián nebo náhodně vybrat některý z původních pixelů. Výsledkem je obraz s nižším rozlišením. Operaci můžeme rekurzivně opakovat a vytvořit tak obrazovou pyramidu.

2.3 Převod do HSV barevného modelu

HSV barevný model nejvíce odpovídá lidskému vnímání barev. Model se skládá ze tří složek: hue, saturation a value. Složka hue představuje odstín barvy. Měří se jako poloha na barevném kole. Její rozsah je 0° až 360°. Složka saturation udává sytost barvy neboli čistotu barvy. Představuje množství šedi v poměru k odstínu. Měří se v procentech, kde 0% představuje šedou a 100% představuje plně sytou barvu. Sytost na barevném kole vzrůstá směrem k okrajům kola. Poslední složka value udává hodnotu jasu. Určuje relativní světlost nebo tmavost barvy.



Obrázek 2: HSV barevný model. Převzato z [7].

Pro převod z barevného prostoru RGB do HSV realizujeme pomocí následujících rovnic

$$H = \begin{cases} nedefinován, & jestliže\ max = min \\ 60° * \dfrac{G-B}{max-min} + 0° \quad , & jestliže\ max = R \wedge G \geq B \\ 60° * \dfrac{G-B}{max-min} + 360°, & jestliže\ max = R \wedge G < B \\ 60° * \dfrac{B-R}{max-min} + 120°, & jestliže\ max = G \\ 60° * \dfrac{R-G}{max-min} + 240°, & jestliže\ max = B \end{cases} \tag{2}$$

$$S = \begin{cases} 0, & jestliže\ max = 0 \\ \dfrac{max-min}{max} = 1 - \dfrac{min}{max}, & jinak \end{cases}$$

$$V = max$$

Kde R, G a B jsou barevné složky v intervalu [0,1], max je rovna největší ze složek R,G a B, min je nejmenší ze složek R,G a B.

Pro snazší reprezentaci se hodnoty HSV převádí na celá čísla v rozsahu 0 až 255.

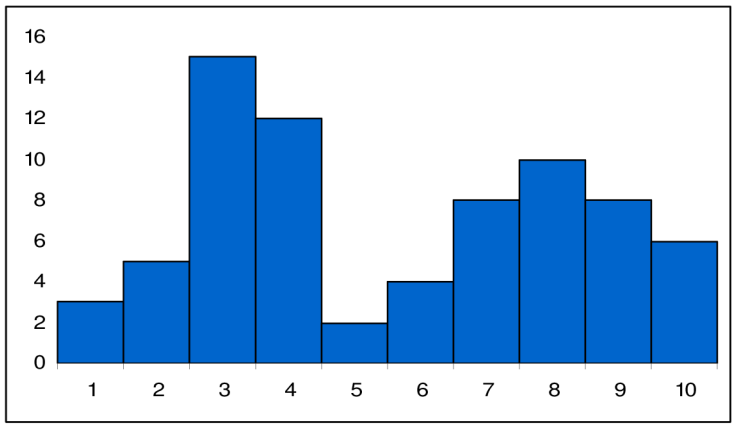
3 Metody detekce pohybu

Detekce pohybu hraje v mnoha aplikacích klíčovou roli. Uplatňuje se často v průmyslových aplikacích, například ve sledování dopravy, ostrahy objektů nebo při detekci nebezpečných předmětů v kolejišti. Detekce pohybu nachází uplatnění i ve vojenských aplikacích. Například při automatickém zaměřování. Protože je detekce pohybu velmi využívaná, snaží se vědci o nalezení nových, rychlejších a přesnějších algoritmů pro detekci nebo o vylepšení stávajících.

Cílem detekce pohybu je určit ve video sekvenci pohybující se objekt. I když tento úkol zní velmi triviálně, dosud neexistuje žádná univerzální metoda pro detekci pohybu, která by výborně pracovala v různých prostředích a za různých podmínek. Při detekci pohybujících se objektů se musíme vypořádat se změnou osvětlení, stíny, odrazy, šumem v obrazu a mnoha dalšími vlivy, které jsou specifické pro místo nasazení aplikace (např. otřesy kamery).

3.1 Histogram

Histogram je využíván ve statistice, k reprezentaci četnosti výskytu hodnot v tabulce. Na ose x jsou jednotlivé třídy histogramu. Každá třída je specifikována na určitém intervalu, který se nepřekrývá s intervaly ostatních tříd. Velikost intervalu bývá obvykle pro všechny třídy stejná. Osa y udává počet výsledků, které patří do příslušné třídy. Na obrázku 3, je znázorněn histogram, který má 10 tříd, každá třída má velikost intervalu 1 a například do třídy 4, patří 12 výsledků.



Obrázek 3: Ukázka histogramu.

Dalším problémem, kterým jsem se musel zabývat, bylo zjistit, zda jsou si dva histogramy podobné nebo zda jsou zcela odlišné. Ke zjištění podobnosti existuje celá řada metod. Některé z nich zde uvedu.

Průnik histogramů (histogram intersection):

$$H(x_1, x_2) = \sum_i \min(x_{1,i}, x_{2,i}) \qquad (3)$$

Porovnání se provádí na normalizovaných histogramech x_1 a x_2. Metoda prochází všechny třídy histogramů a zjišťuje, ve kterém ze dvou histogramů má aktuální třída menší hodnotu. Tuto hodnotu pak započte do výsledku.

Nejmenší čtverce (Chi-square):

$$H(x_1, x_2) = \sum_i \frac{(x_{1,i} - x_{2,i})^2}{x_{1,i} + x_{2,i}} \tag{4}$$

Metoda nejmenších čtverců je daleko přesnější při porovnávání histogramů než metoda průnik histogramů, avšak je časově náročnější. Tuto metodu používám ve své aplikaci k porovnání LBP histogramů

Logaritmická pravděpodobnost (Log-likelihood):

$$H(x_1, x_2) = 2\sum_i \left[x_1 \ln x_1 - x_1 \ln x_2\right] \tag{5}$$

Je časově náročnější než předešlé, protože je nutné vyčíslit logaritmus. Tato metoda pracuje s normalizovanými histogramy.

Kromě těchto vyjmenovaných funkcí se dá nalézt ještě další velké množství funkcí určených pro statistické testy. Jednou z nich je například korelace (Correlation).

3.2 Přehled metod detekce pohybu

3.2.1 Diference pixelů

Diferenční metody porovnávají rozdíl mezi každým pixelem popředí s korespondujícím pixelem pozadí. Pokud je rozdíl pixelů větší než nastavený práh, je pixel klasifikován jako pixel s pohybem. Pokud má zpracovávaný obraz více kanálů, mohu nastavit práh pro každý kanál nebo mohu vypočíst celkový rozdíl ve všech kanálech jako Eulerovu vzdálenost. Pokud použiji Eulerovu vzdálenost, stačí jeden práh pro libovolný počet kanálů.

3.2.2 Porovnání jasových histogramů

V jasovém histogramu představují třídy histogramu intenzitu pixelu. Pokud pracuji se snímkem ve stupních šedi, má jasový histogram 256 tříd. Jasový histogram vytvořím tak, že zjistím intenzitu pixelu a zvýším o 1 počet výskytů ve třídě, která odpovídá intenzitě příslušného pixelu.

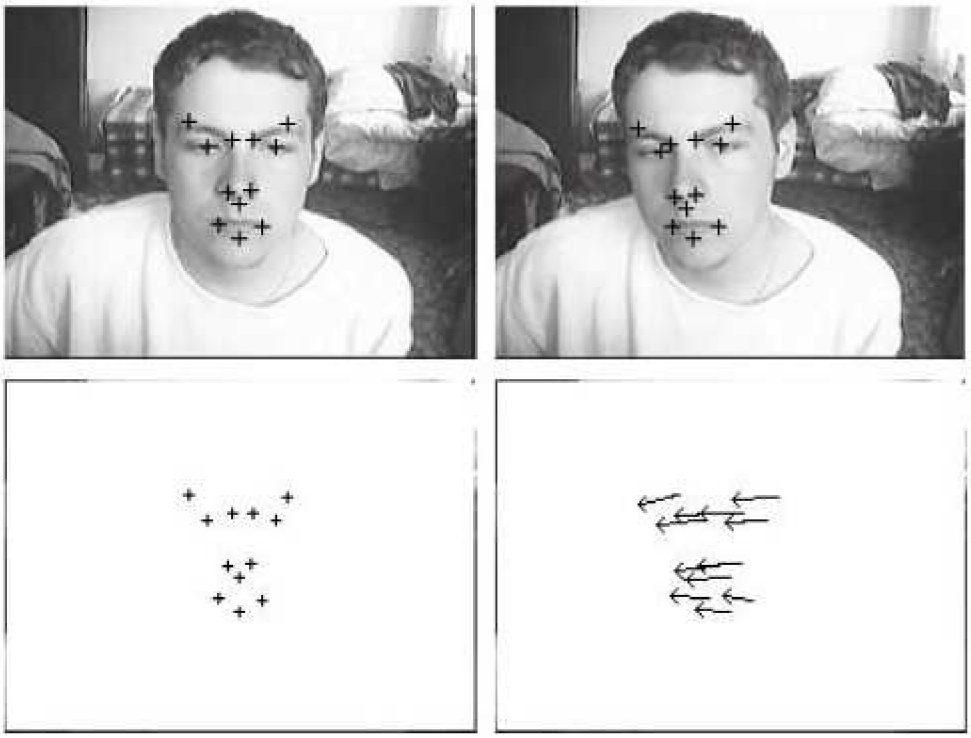
Metoda je založena na porovnání histogramu pozadí s histogramem aktuálního snímku. Pokud jsou histogramy odlišné (odlišné o větší hodnotu než je nastaven práh), mohu konstatovat, že ve scéně došlo k pohybu (Obrázek 4). Bohužel nevím, kde přesně ve scéně došlo k pohybu, protože porovnáním histogramů pouze zjistím, že došlo ke změně jasové charakteristiky snímku.

Obrázek 4: Porovnání jasových histogramů. Červeně vyznačeny významné odchylky.

Jedná se o jednoduchou a rychlou metodu. Je náchylná ke změně světelných podmínek (změní se jas pixelů a tím i celý histogram). Nedokáže určit, kde ve snímku ke změně došlo.

3.2.3 Optický tok

Zachycuji změny obrazu v čase. Každý bod obrazu má přiřazen vektor, který obsahuje směr a rychlost pohybu v daném bodě. Jsem schopen popsat změny obrazu uvnitř regionu, který je dán parametrickými funkcemi souřadnic (Obrázek 5).



Obrázek 5: Detekce pohybu pomocí optického toku. Převzato z [2].

Metoda dokáže určit místo, kde k pohybu došlo. Ale výpočetně je velmi náročná. Pokud by v obraze nedocházelo k pohybu, zbytečně bych plýtval výkonem. Tento algoritmus je vhodnější použít jako dodatečnou detekci, například k predikci pohybu. Další nevýhodou je, že potřebuje konstantní světelné podmínky.

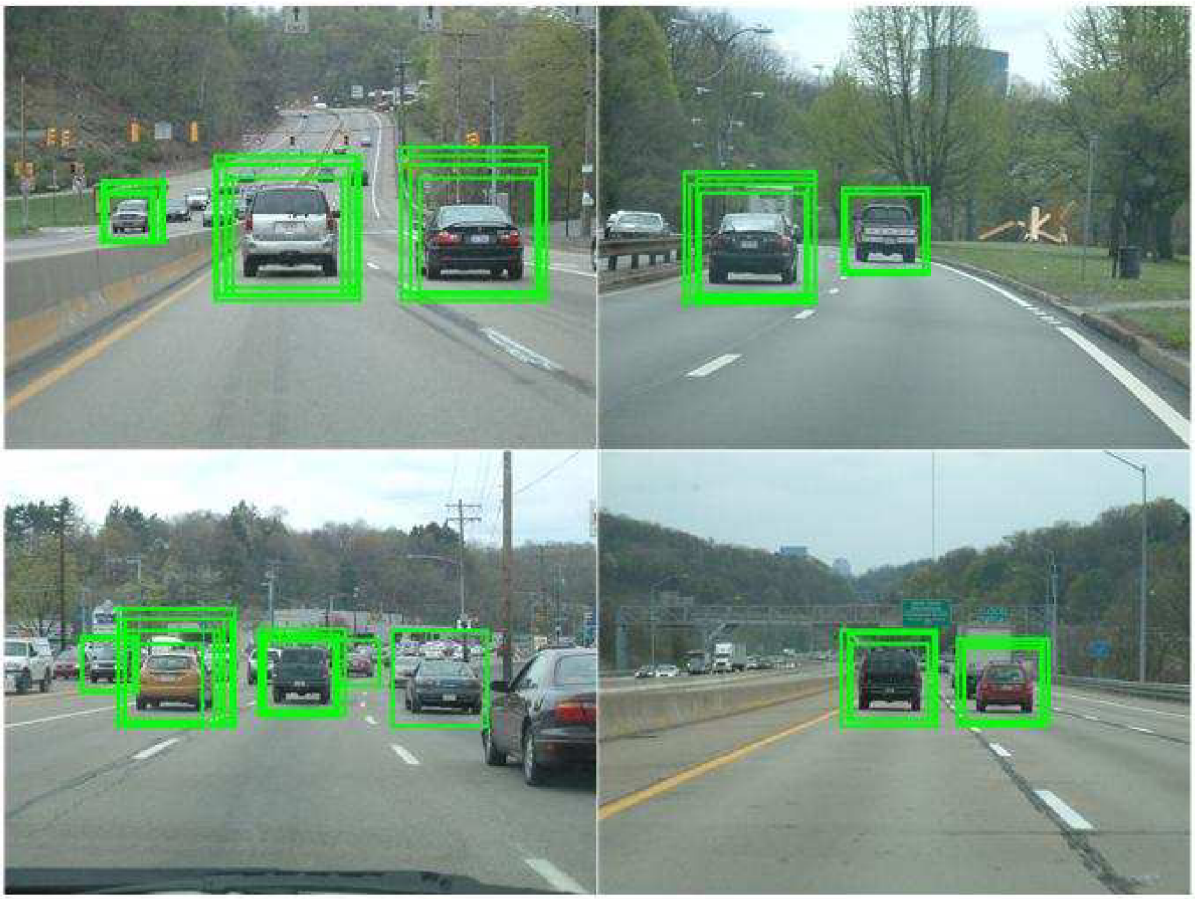
Detailní popis a použití této metody lze nalézt v [2].

3.2.4 Využití klasifikátoru k detekci pohybujících se objektů

Klasifikátorů se využívá ke zjištění přítomnosti objektu v obraze. Pokud budeme zjišťovat přítomnost objektu v jednotlivých snímcích video sekvence, jsme následně schopni konstatovat, zda se jedná o pohybující se objekt.

Klasifikátor je natrénován na snímcích hledaného objektu a pozadí. Vstupem klasifikátoru mohou být přímo hodnoty pixelů, ale častěji se používají odezvy jednoduchých filtrů. Klasifikátorů existuje velké množství. Zde uvádím několik možných variant klasifikátorů:

- Modelování rozložení pravděpodobnosti tříd (Bayes)
- Neuronové sítě [17]
- Konvoluční neuronové sítě
- AdaBoost [18]
- SVM [19]
- RVM [20]



Obrázek 6: Využití klasifikátoru pro detekci automobilů. Převzato z [6].

Klasifikátory jsou schopny zpracovávat vstupní obraz velmi rychle a jsou schopny vyhledat i velmi malé objekty. Dosahují vysoké úspěšnosti detekce. Klasifikátory umí vyhledávat omezený počet tříd objektů. Tudíž se dají využít k detekci pohybujících se objektů specifické třídy (budu detekovat pohybující se auta, ale nebudu schopen detekovat pohybující se osoby). Klasifikátory potřebují velké množství dat a času k natrénování.

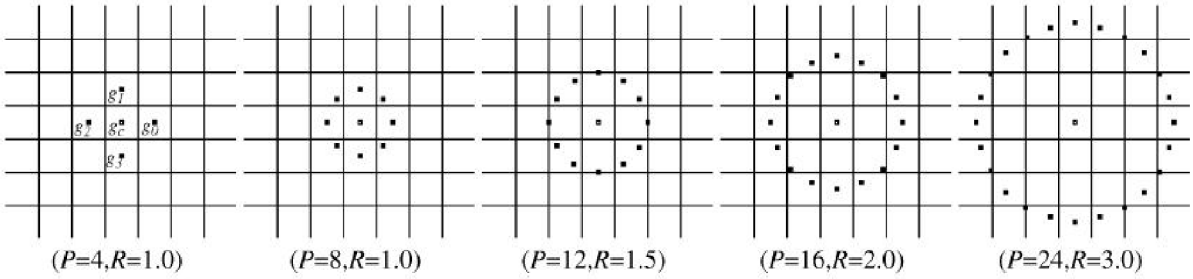
3.2.5 Local Binary Pattern

Local Binary Patterns (dále jen LBP) je metoda původně vyvinutá pro klasifikaci textur. Byla vyvinuta na universitě Oulu, která se nachází ve Finsku. Kromě klasifikace textur našla uplatnění i v segmentaci obrazu, detekci obličeje a detekci pohybu.

Metoda LBP nepoužívá pouze hodnotu pixelu. Bere v úvahu i hodnoty okolních pixelů. Pomocí LBP operátoru vypočítá pro daný pixel LBP příznak. Pří detekci pohybu se pomocí LBP příznaků konstruuje LBP histogram. Porovnáním LBP histogramů popředí a pozadí lze určit, zda dochází ve snímku k pohybu. Bližší popis metody naleznete v kapitole 3.3.

3.3 Local Binary Pattern

Metoda LBP se nezabývá pouze hodnotou pixelu, ale při svém výpočtu uvažuje i hodnoty okolních pixelů. Protože se LBP operátor zabývá nejen zkoumaným pixelem, ale i jeho okolím, je nutno určit, v jakém okolí bude pracovat. LBP operátor $LBP_{P,R}^{riu2}$ je určen počtem zkoumaných sousedních pixelů P a vzdáleností těchto pixelů od zkoumaného pixelu R (tzv. rádius). Na obrázku (Obrázek 7) je znázorněno, jakými sousedními pixely se bude zabývat LBP operátor pro různá nastavení parametrů P a R.



Obrázek 7: Převzato z [4]. Ukazuje různá nastavení parametrů P a R LBP operátoru.

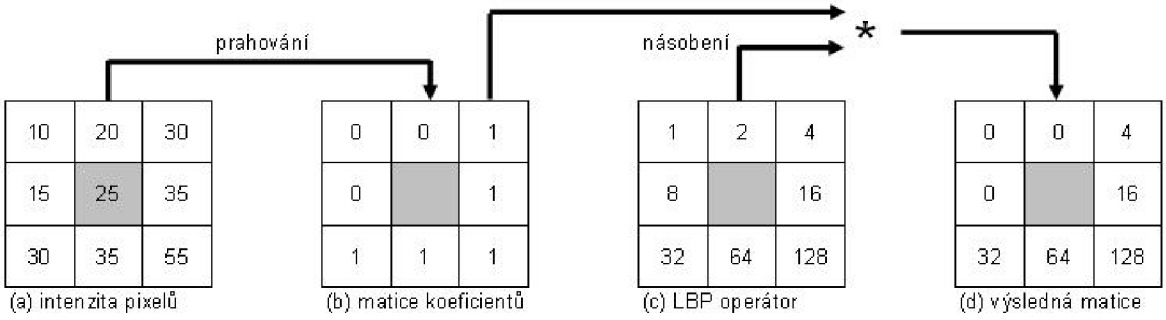
Po určení sousedních pixelů, se kterými bude LBP operátor pracovat, je dále nutné určit váhu sousedních pixelů, s jakou se budou započítávat do výsledné hodnoty LBP příznaku. Ve své aplikaci používám LBP operátor P=8, R=1.0. Operátor s příslušnými váhami je uveden na obrázku 8c.

LBP příznaky se nejčastěji vyhodnocují nad obrazem převedeným do stupňů šedi.

Výpočet LBP příznaku pixelu probíhá následovně. Pomocí prahování hodnoty středového pixelu získám matici koeficientů (Obrázek 8b). Prahování provádím podle vzorce:

$$f(x) = \begin{cases} 0 & x > y \\ 1 & x \leq y \end{cases} \qquad (6)$$

Kde x je hodnota středového pixelu a y je hodnota sousedního pixelu. Poté vynásobím matici koeficientů (Obrázek 8b) s LBP operátorem (obrázek 8c) (pozn. Nejedná se o násobení matic. Pouze mezi sebou násobím prvky matic se stejnými indexy). Vynásobením dostanu výslednou matici (Obrázek 8d). LBP příznak pro středový pixel získám součtem prvků výsledné matice. LBP příznak středového pixelu v příkladu by byl 4+16+32+64+128=244.



Obrázek 8: Výpočet LBP

Pokud je středovým pixelem krajní bod obrazu (nemá úplné okolí), buď se nebude LBP příznak pro krajní bod počítat, nebo jsou neexistující body v matici koeficientů nahrazeny nulou.

Četnost jednotlivých LBP příznaků v obraze se vyhodnocuje pomocí histogramu. Třídy LBP histogramu odpovídají hodnotám, kterých může nabývat LBP příznak pixelu. LBP histogram má typicky 256 tříd. Detekci pohybu pak provádím na základě porovnání LBP histogramu pozadí s LBP histogramem aktuálního snímku. Pokud je rozdíl histogramů dostatečný (větší než práh), mohu konstatovat, že došlo k pohybu (histogramy a jejich porovnání popisuji v kapitole 3.1).

Metoda se dá zařadit mezi rychlé metody. Výsledek není ovlivňován změnou světelných podmínek (globální změnou osvětlení). Další výhodou je, že drobné pohyby (jako pohyb listů nebo trávy) neovlivňují výsledek. Jedinou podstatnou nevýhodou při realizaci detektoru pohybu je, že LBP příznaky nejsou schopny určit místo pohybu v obraze. Odstraněním této vady se zabývám v kapitole 4.1.

3.4 Aktualizace pozadí

Aktualizace pozadí je v podstatě započtení aktuálního snímku do pozadí. Tedy pokud do záběru vstoupí objekt a zůstane na stejném místě určitou dobu (určitý počet snímků), nebude považován již za pohybující se objekt, ale za pozadí. Volba správné aktualizace se také odvíjí od účelu aplikace. Pokud budu sledovat parkoviště, tak auto, které zaparkuje na parkovacím místě, mohu do 2 minut přestat považovat za pohyblivý objekt, ale již ho považovat za pozadí. Avšak pokud se budu snažit detekovat překážky v kolejišti, byla by chyba, aby předmět, který představuje ohrožení, se stal pozadím.

Aktualizace pozadí se dá dále využít při změně osvětlení. Detektor může detekovat chvíli pohyb, ale po určité době se sám s nastalými podmínkami vyrovná a již pracuje správně. Podobný problém jako změna osvětlení může způsobovat i posunutí kamery.

Jak jsem naznačil, je aktualizace pozadí velmi rozsáhlým oborem, proto zde uvedu jenom několik metod aktualizace pozadí.

Průměr předchozích snímků je velmi jednoduchá metoda. Vezmu několik předchozích snímků a pro každý pixel vypočítám průměrnou hodnotu. Průměrnou hodnotu poté považuji za pozadí.

$$x = \frac{\sum_i y_i}{i} \tag{7}$$

X je průměrná hodnota pixelu, i počet předchozích snímků a y hodnota pixelu předchozího snímku.

Váha předchozích snímků je podobná předchozí metodě. Jediný rozdíl je, že předchozí snímky se do výsledku nezapočítávají se stejnou vahou.

Započtení aktuálního snímku vahou do pozadí. Tato metoda započte hodnoty, které obsahuje aktuální snímek x_a s určitou vahou α do pozadí x_b.

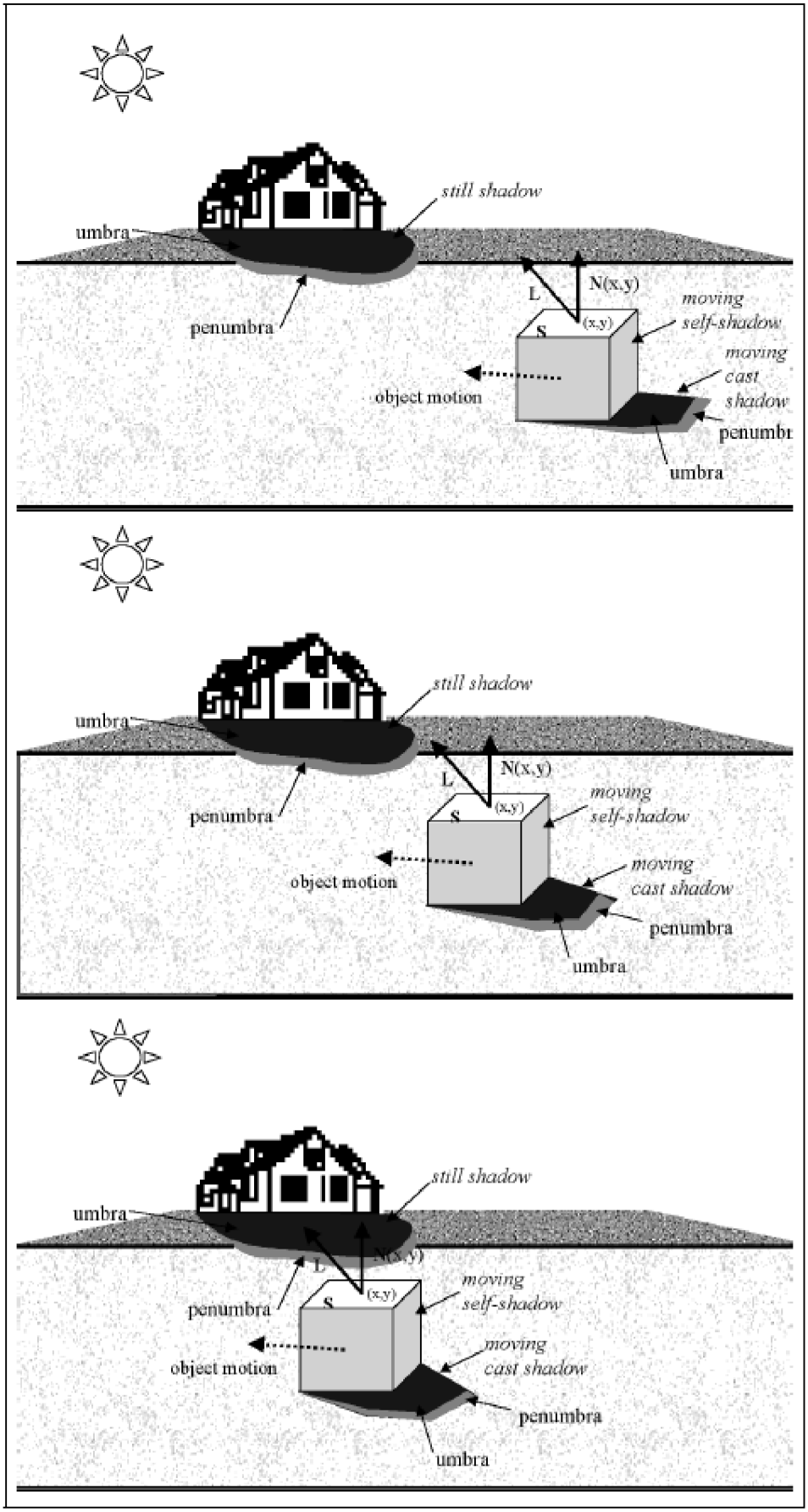
$$x_{b+1} = \alpha x_a + (1-\alpha)x_b \tag{8}$$

4 Metody detekce stínů

Detekce stínů pohybujících se objektů kriticky ovlivňuje přesnost detekce objektů. Stín je často chybně označen jako součást pohybujícího se objektu, což způsobuje chyby při trackingu a segmentaci.

4.1 Stín a model stínu

V [9] je stín popsán a modelován jako místo, tmavá oblast, kam nedopadá světlo. Část objektu, která není osvětlena, se nazývá self-shadow. Cast-shadow je plocha ve scéně, na kterou nedopadá světlo, kvůli objektu, který se nachází mezi světelným zdrojem a touto plochou. Cast-shadow dále můžeme rozdělit na umbru a preumbru (Obrázek 9). Umbra je oblast, ve které objekt zcela blokuje světlo ze zdroje. Preumbra je oblast, ve které je světlo ze zdroje pouze částečně blokováno. Pokud se objekt pohybuje cast-shadow se nazývá moving cast shadow (stín vrhaný pohybujícím se objektem).



Obrázek 9: Rozdělení stínů. Převzato z [9].

Člověk využívá pro detekci stínů několika různých vlastností stínů

- stín ztmavuje texturu pozadí, na které dopadá, ale nijak významně nemění barvu a texturu pozadí.
- stín je vždy spojen s objektem, který ho vrhá.
- stín je projekcí tvaru objektu na pozadí. Pro plošné světelné zdroje je nepravděpodobné, že projekce bude perspektivní.
- je známá pozice a síla světelného zdroje
- hranice stínu mají tendenci měnit tvar v závislosti na geometrii povrchu, na který je stín vrhán.

Algoritmy pro klasifikaci stínů vycházejí z fyzikálních předpokladů. Pokud zdroj vyzařuje bílé světlo a světlo dopadá na matný povrch (odrazivost povrchu je skoro nulová) můžeme pixel klasifikovat následující rovnicí:

$$s_k(x,y) = E_k(x,y)\rho_k(x,y) \tag{9}$$

kde s_k je jas pixelu na souřadnicích (x,y) v čase k. $\rho_k(x,y)$je odrazivost povrchu objektu a $E_k(x,y)$ je ozáření (irradiance). Ozáření je množství energie obdržené povrchem S. Hodnotu ozáření můžeme odvodit pomocí Phongova osvětlovacího modelu. Pokud je objekt daleko od světelného zdroje, vzdálenost mezi světelným zdrojem a povrchem S je konstantní, světelný zdroj vyzařuje rovnoběžné světelné paprsky a poloha pozorovatele se nemění, můžeme ozáření $E_k(x,y)$ aproximovat jako:

$$E_k(x,y)\begin{cases} c_A + c_P \cos\angle(N(x,y),L) & osvětlený \\ c_A + k(x,y)c_P \cos\angle(N(x,y),L) & preumbra \\ c_A & umbra \end{cases} \tag{10}$$

kde c_A je intenzita ambientního světla, c_P intenzita světelného zdroje, L směr světelného zdroje, $N(x,y)$ normála k povrchu objektu a $k(x,y)$ popisuje tmavost preumbry ($0 \leq k(x,y) \leq 1$).

Předchozí popis není vhodný pro detekci stínů ve video sekvenci. Máme pouze informace z video sekvence, parametry $L, N, k, c_A\, a\, c_P$ neznáme. Proto algoritmy pro detekci stínů přidávají některé zjednodušující podmínky:

- zdroj světla je silný
- pozadí je statické a má na sobě texturu
- pozadí se bere jako rovina
- plocha světelného zdroje umožňuje vznik peumbry

Ozáření a jas vypočítáme pomocí rozdílu aktuálního snímku s_k s referenčním snímkem (modelem pozadí) s_0. Pokud je model pozadí statický, odrazivost $\rho_k(x,y)$ se nemění. Tudíž $\rho_k(x,y) = \rho_0(x,y) = \rho(x,y)$. Rozdíl můžeme zapsat jako:

$$D_k(x,y) = \rho(x,y)\big(E_k(x,y) - E_0(x,y)\big) \tag{11}$$

Pokud na pixel dopadá stín (umbra) rozdíl by měl být velký. Pro umbru můžeme předchozí rovnici (11) vyjádřit jako:

$$D_k(x,y) = \rho(x,y)c_P \cos\angle(N(x,y),L) \tag{12}$$

kde c_P vyjadřuje sílu světelného zdroje.

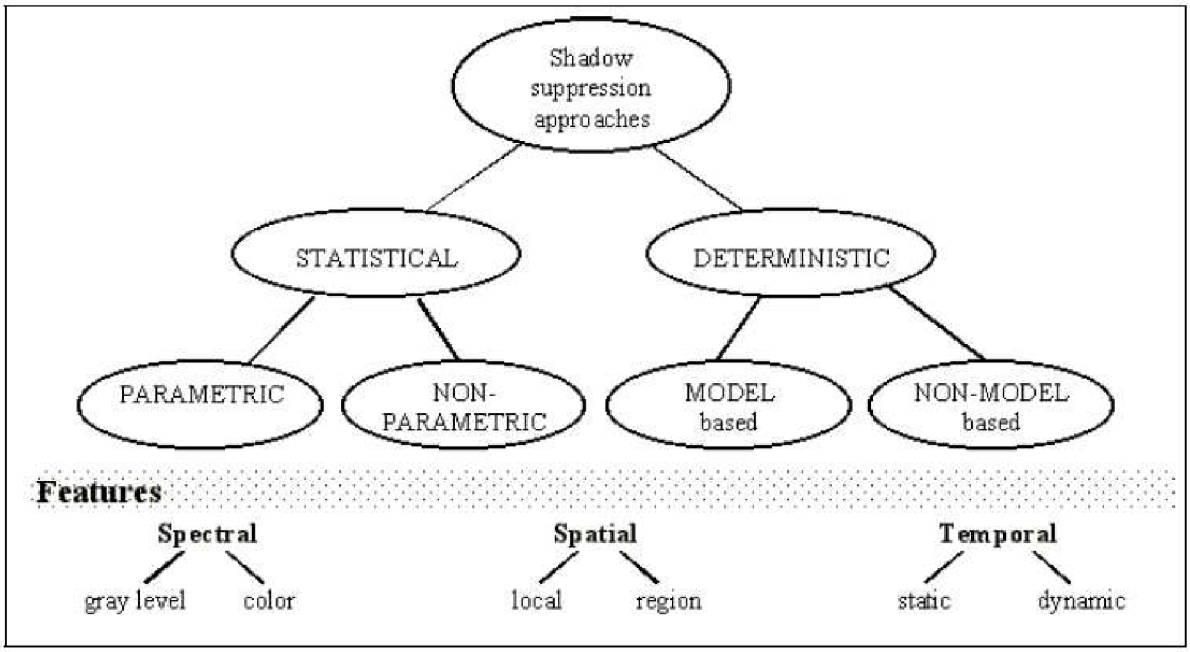
Rozdíl však není schopen rozpoznat preumbru. Pro pixely preumbry lze rozdíl vyjádřit jako:

$$D_k(x,y) = \rho(x,y)\big(1 - k(x,y)\big)c_P \cos\angle(N(x,y),L) \tag{13}$$

Pokud je parametr $k(x,y)$ malý, je preumbra tmavá. $1 - k(x,y)$ poskytuje dobrou diskriminaci mezi pixely stínu a pixely pozadí.

4.2 Rozdělení metod pro detekci stínů

Metody detekce stínů se od sebe liší podle toho, jakým způsobem rozlišují mezi pixely popředí a pozadí. Rozdělení metod může být na základě různých parametrů. V této práci využiji rozdělení uvedené v [9].



Obrázek 10: Rozdělení metod pro detekci stínů. Převzato z [9].

Deterministický přístup využívá tvrdého rozhodnutí. Striktně rozhodne, do jaké třídy pixel patří. Deterministický přístup pak můžeme dále rozdělit podle toho, zda používá databázi znalostí (model). Pokud zvolíme metody používající modely, dosáhneme nejlepších výsledku. Ale metody založené na modelech ve srovnání s metodami nepoužívajícími modely, jsou mnohem více komplexní a časově náročnější.

Statistický přístup využívá pravděpodobnostních funkcí k určení příslušnosti do třídy. Pixel přísluší k třídě s jistou pravděpodobností. Kritickým je vhodně zvolit parametr selekce. Proto můžeme statický přístup dále rozdělit na parametrické a bezparametrické metody.

Dále můžeme metody rozdělit podle tří domén:

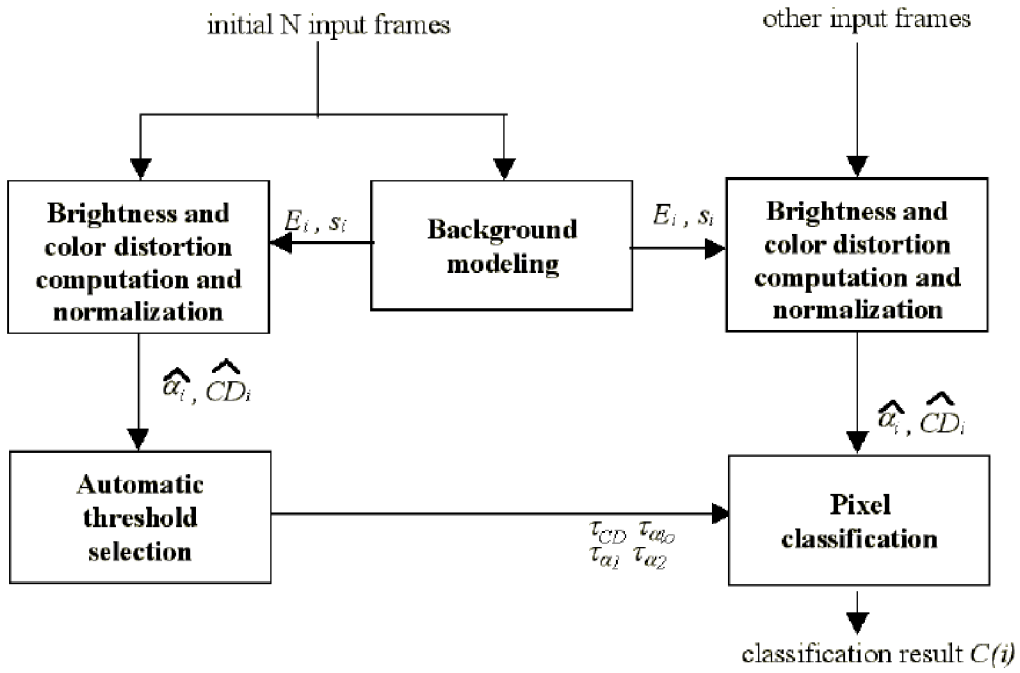
- spektra - používají šedotónový nebo barevný obraz
- prostoru - pracují s pixely nebo oblastí
- času - některé metody využívají časového zpoždění ke zkvalitnění výsledků

4.3 Metody detekce stínů

V této části popisuji čtyři metody detekce stínů. Popis metod je převzat z [9]. Metody nemají přesné názvy, většinu metod nazývám podle třídy, do které patří.

Statistická bezparametrická metoda (Statistical non-parametric - SNP)

Metoda je založena na modelování pixelu a modelu pozadí [21]. Princip metody je znázorněn na obrázku 11.



Obrázek 11: Schéma SNP metody. Převzato z [9].

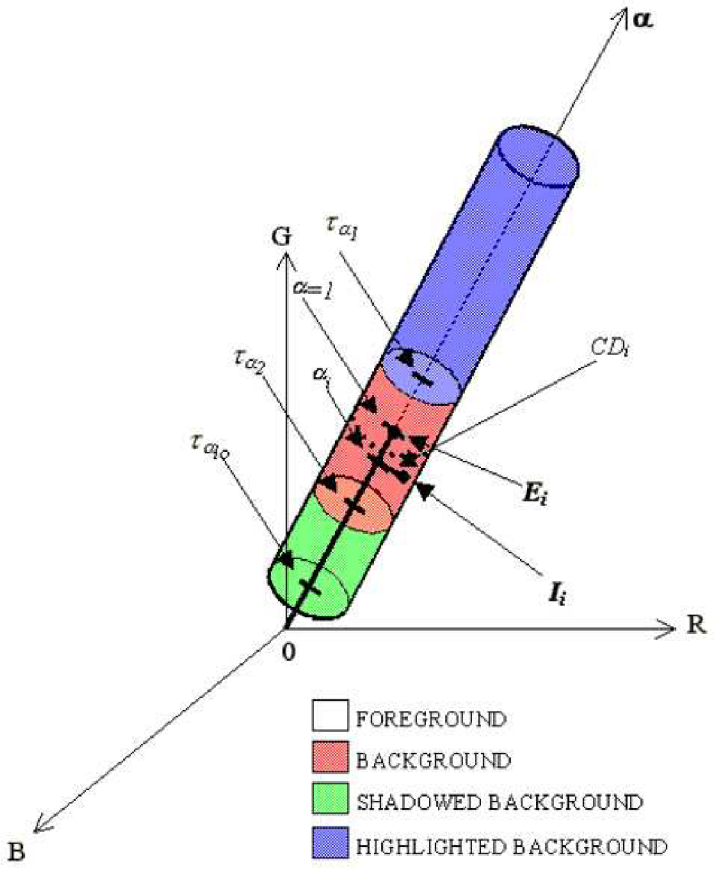
Prvních N snímků se použije k vytvoření modelu pro každý pixel. Pro každý pixel je vypočtena střední hodnota $E_i = [\mu_R(i), \mu_G(i), \mu_B(i)]$ a rozptyl $s_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)]$ v každém barevném kanálu. Deformace jasu α_i a deformace barvy CD_i jsou určeny jako rozdíl mezi očekávanou barvou pixelu a jeho skutečnou hodnotou ve snímku $I_i = [I_R(i), I_G(i), I_B(i)]$.

$$\alpha_i = \frac{\left(\frac{I_R(i)\mu_R(i)}{\sigma_R^2(i)} + \frac{I_G(i)\mu_G(i)}{\sigma_G^2(i)} + \frac{I_B(i)\mu_B(i)}{\sigma_B^2(i)}\right)}{\left(\left[\frac{\mu_R(i)}{\sigma_R(i)}\right]^2 + \left[\frac{\mu_G(i)}{\sigma_G(i)}\right]^2 + \left[\frac{\mu_B(i)}{\sigma_B(i)}\right]^2\right)} \tag{14}$$

$$CD_i = \sqrt{\left(\frac{I_R(i) - \alpha_i\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\frac{I_G(i) - \alpha_i\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\frac{I_B(i) - \alpha_i\mu_B(i)}{\sigma_B(i)}\right)^2} \tag{15}$$

Na základě normalizovaných hodnot $\widehat{\alpha_\iota}$ a $\widehat{CD_\iota}$ je pixel i klasifikován do jedné ze čtyř tříd:

$$C(i)\begin{cases} Popředí: \widehat{CD_\iota} > \tau_{CD} \vee \widehat{\alpha_\iota} < \tau_{\alpha lo}\ jinak \\ Pozadí: \widehat{\alpha_\iota} < \tau_{\alpha 1} \vee \widehat{\alpha_\iota} > \tau_{\alpha 2}\ jinak \\ Stín\ na\ pozadí: \widehat{\alpha_\iota} < 0\ jinak \\ Osvětlené\ pozadí: zbylé\ případy \end{cases} \tag{16}$$

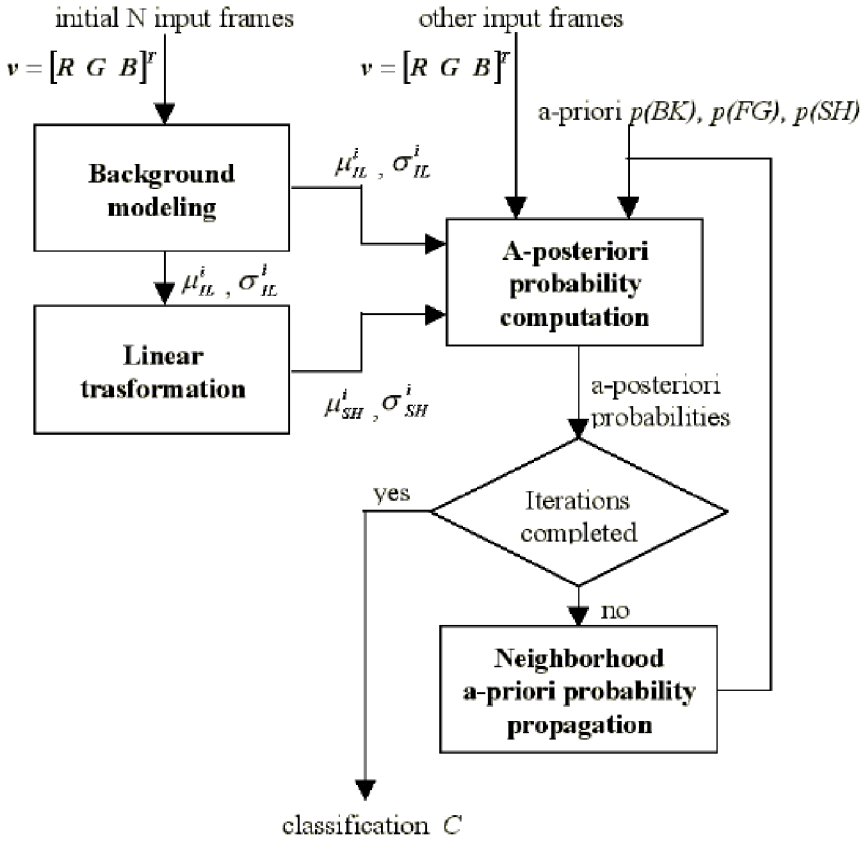


Obrázek 12: Klasifikace pixelu. Převzato z [9].

Stíny mají podobnou barvu, ale nižší jas než pozadí. Dolní mez u popředí je k zamezení chybné detekce objektů s velmi nízkou hodnotou RGB jako stínů. Vhodné hodnoty prahů se určí automaticky z histogramů $\widehat{\alpha_\iota}$ a $\widehat{CD_\iota}$. Metoda je robustnější než metody založené na Gaussově rozložení.

Statistická parametrická metoda (Statical parametric - SP)

Tato metoda používá dva zdroje informací: lokální, založenou na vzhledu pixelu a prostorovou, založenou na předpokladu, že objekt a stín jsou kompaktní regiony [22].



Obrázek 13: Schéma SP. Převzato z [9].

Nejdříve se vypočte střední hodnota a odchylka pozadí pro všechny pixely oblastí. Pravděpodobnostní rozložení pixelů se stínem $\aleph(\boldsymbol{\mu_{SH}}, \boldsymbol{\sigma_{SH}})$ je odhadnuto z pixelů bez stínu $\aleph(\boldsymbol{\mu_{IL}}, \boldsymbol{\sigma_{IL}})$. $\boldsymbol{v} = [\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}]^T$je hodnota pixelu bez stínu. Pomocí aproximace lineární transformace $\overline{\boldsymbol{v}} = \boldsymbol{Dv}$ odhadneme barvu pixelu se stínem. $\boldsymbol{D} = \boldsymbol{diag}(\boldsymbol{d_R}, \boldsymbol{d_G}, \boldsymbol{d_B})$ je diagonální matice získaná na základě experimentálního ohodnocení. Matice D předpokládá konstantní plochý povrch. Pokud není pozadí ploché přes celý snímek, musí se vypočíst matice D pro každou podoblast. Pro referenční pixely mám dánu střední hodnotu a odchylku. Z těchto hodnot jsem pak schopen odhadnout hodnotu pod stínem jako $\boldsymbol{\mu_{SH}^i} = \boldsymbol{\mu_{IL}^i d_i}$ a $\boldsymbol{\sigma_{SH}^i} = \boldsymbol{\sigma_{IL}^i d_i}$, kde $\boldsymbol{i} \in \boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}$. Nakonec jsou vypočteny a maximalizovány posteriorní pravděpodobnosti náležející třídám pozadí, popředí a stínů.

Ke zlepšení výkonu detekce se využívá iterativní šíření informace z okolních pixelů. K přiřazení členství ve třídě, je v dalším kroku využit výsledek z předchozích kroků provedených na sousedních pixelech.

Hlavní nevýhodou je obtížná volba parametrů. Je nutná manuální segmentace určitého počtu snímků, aby bylo možné vytvořit statistiky a vypočíst matici D. Expectation maximization (EM) metoda by mohla být využita k automatizaci tohoto procesu.

Diference stínů v HSV barevném prostoru

Tuto metodu lze zařadit mezi deterministické metody nepoužívající model (deterministic non-model based - DNM). Metoda pracuje v HVS barevném prostoru na úrovni pixelů. HSV barevný prostor poměrně dobře reprezentuje lidské vnímání barev. Stín výrazně nemění odstín (složku hue) pozadí, ale výrazněji se projevuje ve složce saturation. Aby byl pixel označen za pixel obsahující pohybující se stín, musí splňovat následující podmínku:

$$\alpha \leq \frac{I_k^V(x,y)}{B_k^V(x,y)} \leq \beta \wedge \left(I_k^S(x,y) - B_k^S(x,y)\right) \leq \tau_S \wedge \left|I_k^H(x,y) - B_k^H(x,y)\right| \leq \tau_H \qquad (17)$$

kde $I_k(x,y)$ je pixel aktuálního snímku na souřadnicích x a y. $B_k(x,y)$ je pixel modelu pozadí na stejných souřadnicích jako pixel popředí. β zabraňuje, aby body pozadí, které se změnily kvůli šumu, byly identifikovány jako stín. Parametr α určuje, jak silný je světelný zdroj. Čím Slunce svítí silněji a je výše na obloze, tím musí být hodnota α menší.

Deterministická metoda bez modelu (Deterministic non-model based - DNM)

Tato metoda je velmi komplexní [23]. Dokáže detekovat i preumbru pohybujícího se stínu. Stín je detekován na základě tří kritérií: hran, tmavých oblastí a výrazných jasových změn.

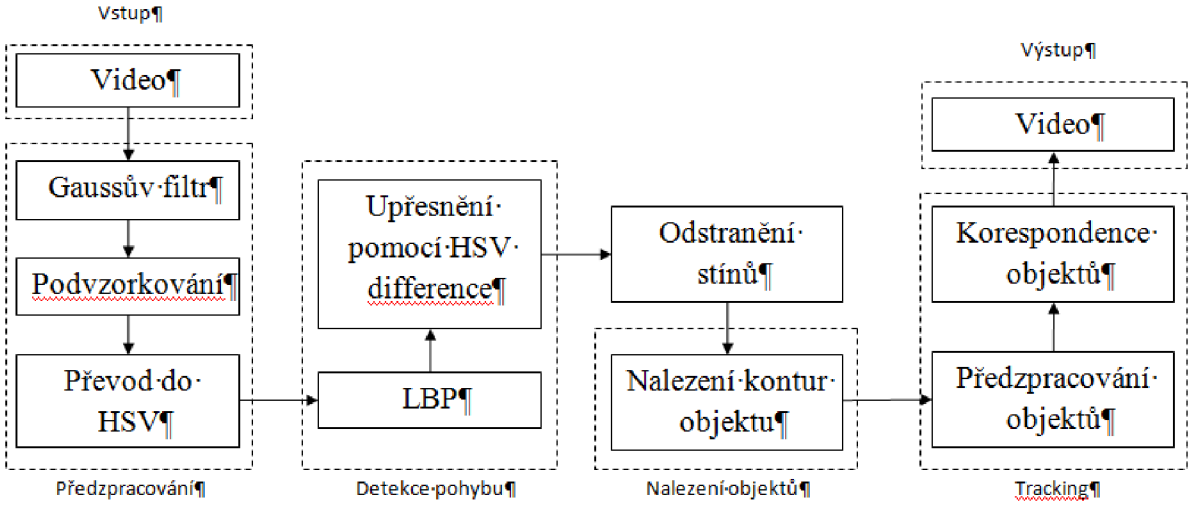
Hrany jsou rozděleny na hrany pohybující se a hrany statické. Hrana je klasifikována jako statická, pokud energie ve vysokých frekvencích v rozdílovém snímku (rozdílu současného a předchozího snímku) je nízká v blízkém okolí pixelu.

Pokud je oblast ovlivněna stínem, je poměr ozáření současného snímku a předchozího snímku pro všechny pixely v dané oblasti konstantní.

Pomocí těchto dvou podmínek a heuristických pravidel je každý region klasifikován do jedné ze sedmnácti tříd. Pro výpočet preumbry autoři této metody používají šířku hrany v rozdílovém obraze. Na hranici stínů, kde se vyskytuje preumbra, je rozdíl jasů nižší než na hraně objektu. Autoři tvrdí, že šířka hrany je velmi spolehlivý způsob, jak rozlišit objekt a stín.

5 Návrh aplikace

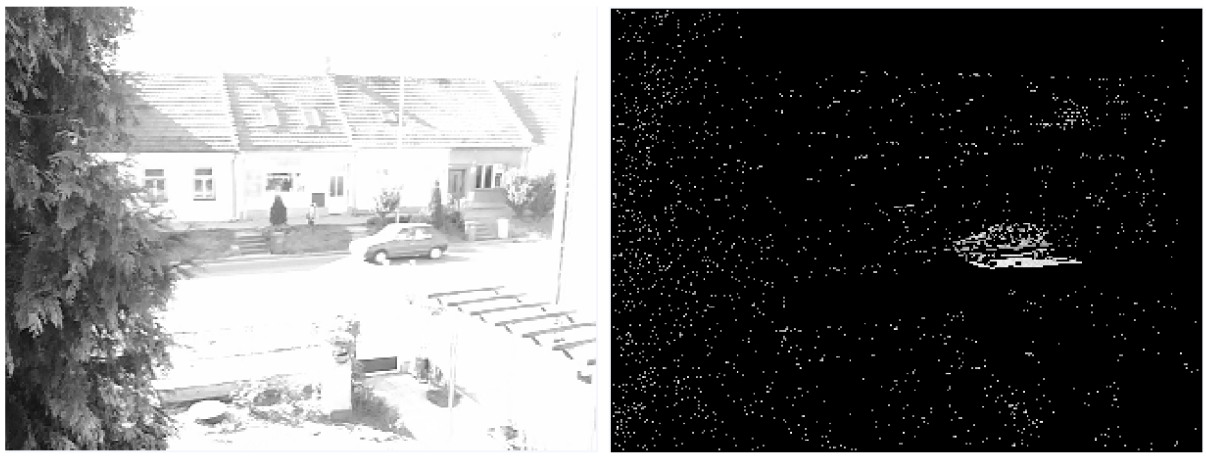
Mým cílem je vytvořit aplikaci, která bude schopna detekovat pohyb a sledovat průchod pohybujícího se objektu scénou. Při detekci kladu důraz nejen na přesnost, ale i rychlost detekce. Protože nelze sledovat všechny možné druhy objektů, zaměřím se ve své aplikaci na sledování automobilů. Budu se snažit, aby aplikace byla schopna detekce v reálném čase. Na obrázku 14 je návrh mé aplikace.



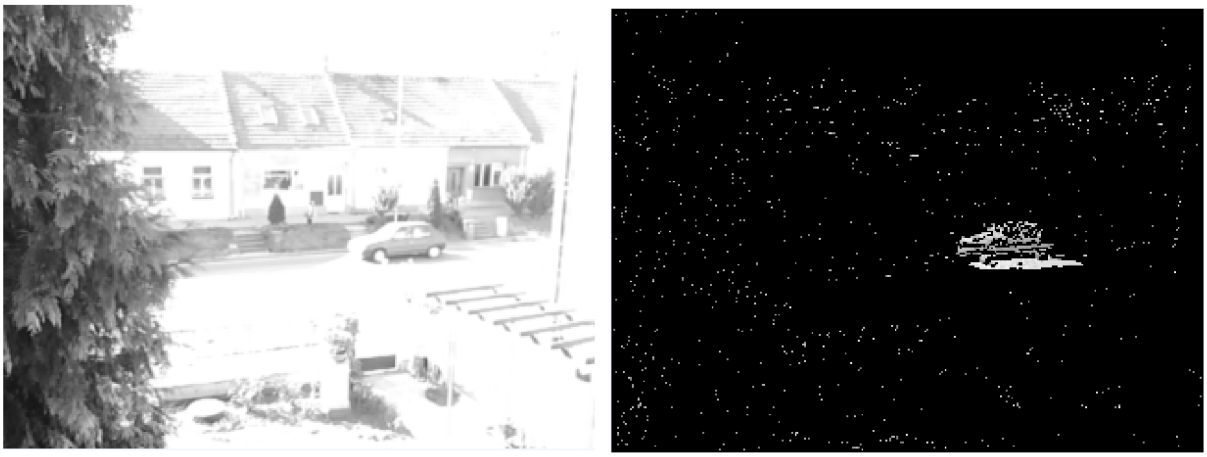
Obrázek 14: Návrh aplikace

5.1 Předzpracování obrazu

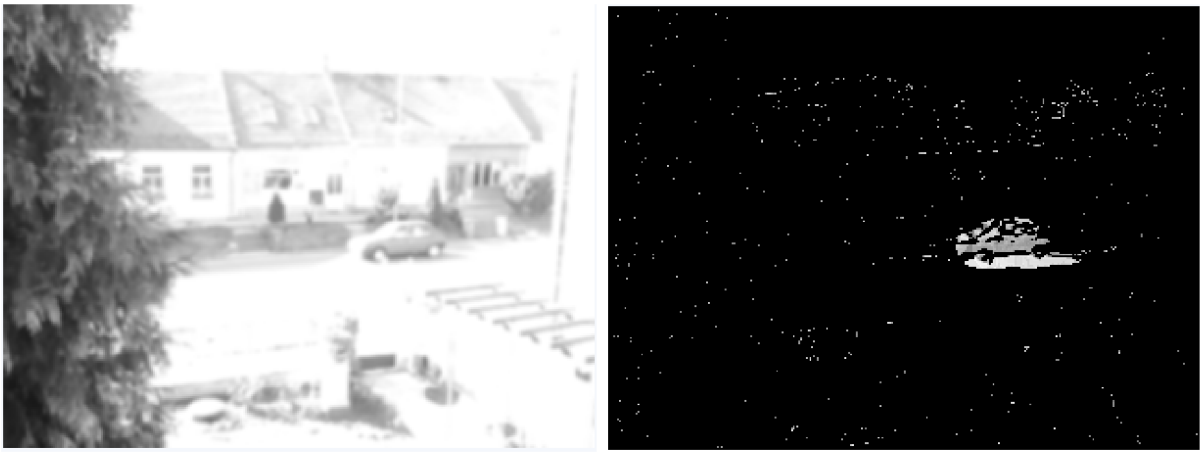
V aplikaci šum způsoboval velké problémy, proto jsem se rozhodl aplikovat na vstupní obraz Gaussův filtr. Musel jsem však vyřešit, jaké rozměry má mít použitý filtr, aby odstranil co nejvíce šumu, ale zároveň aby nebyl obraz příliš deformován a výpočet netrval neúměrně dlouho.



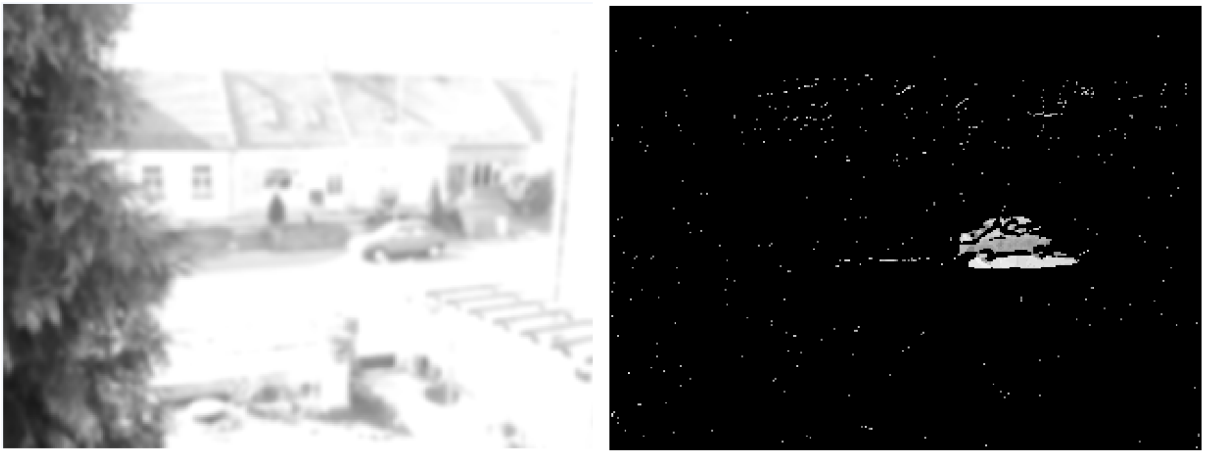
Obrázek 15: Nepoužit Gaussův filtr



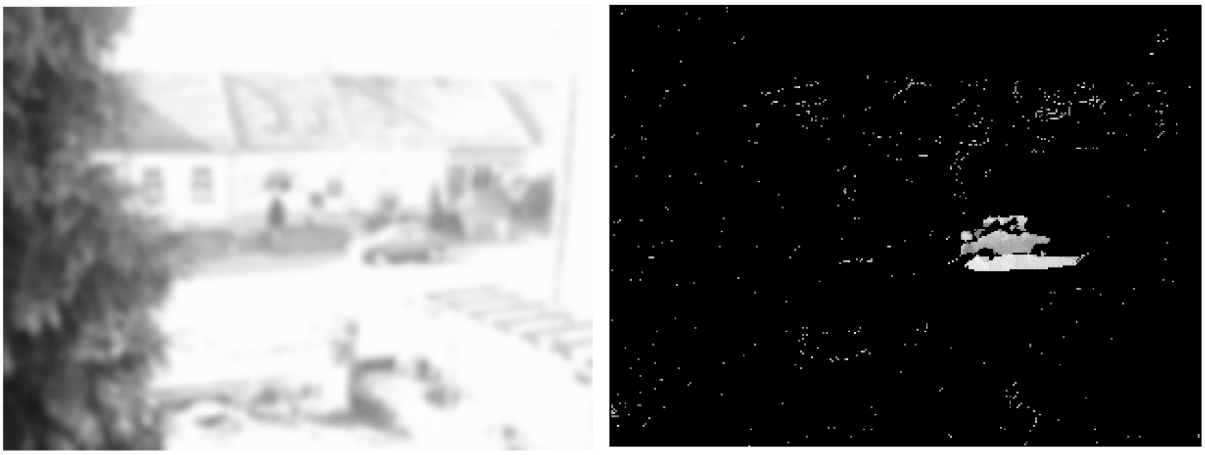
Obrázek 16: Použit Gaussův filtr o rozměrech 5x5 s σ=1



Obrázek 17: Použit Gaussův filtr o rozměrech 11x11 s σ=2,2



Obrázek 18: Použit Gaussův filtr o rozměrech 15x15 s σ=3



Obrázek 19: Použit Gaussův filtr o rozměrech 21x21 s σ=4,2

Jak můžete na obrázcích 15 až 19 vidět, nejlepších výsledků dosáhl filtr o rozměru 15x15 pixelů s σ=3 (Obrázek 18). Tento filtr používám ve své aplikaci k redukci šumu.

Po aplikaci Gaussova filtru oblast podvzorkuji. Podvzorkováním výrazně zmenším počet pixelů, který musím zpracovávat algoritmy pro detekci pohybu, čímž výrazně zkrátím čas potřebný pro detekci pohybu. Cenou za snížení počtu pixelů je ztráta informace.

V mé aplikaci má oblast, z níž vybírám pixel, velikost 2x2. Výsledný obraz má pak pouze $\frac{1}{4}$ pixelů původního obrazu (tzn. Výška i šířka obrazu se zmenší na polovinu). Dále obraz již nepodvzorkovávám. Do výsledného obrazu ukládám levý horní pixel. Tím ušetřím čas potřebný k podvzorkování a zároveň ztratím minimum informace obsažené v ostatních pixelech. Takovýto způsob podvzorkování si mohu dovolit, protože Gaussův filtr distribuoval informace z pixelů do okolí.

5.2 Výpočet LBP příznaku a rozdělení snímku na bloky

Do LBP detektoru vstupuje snímek. Tento vstup označuji jako popředí. V [5] autoři vyhodnocují LBP příznaky nad obrazem převedeným do stupňů šedi. Pokusy jsem však zjistil, že je lepší převést vstupní obraz, který má pixely v RGB barevném prostoru, do HSV barevného prostoru. Následně, vyhodnocuji LBP příznaky nad složkou hue. Touto úpravou jsem dosáhl lepší odolnosti vůči šumu.

Pro každý pixel popředí je vypočten LBP příznak, který je uložen do matice. Stejně jako pro popředí vypočtu LBP příznaky i pro pozadí a uložím je také do matice. LBP hodnota krajních pixelů snímku je nula. Pokud nyní vytvořím LBP histogramy pro popředí a pozadí a provedu jejich srovnání, mohu pouze konstatovat, že ve snímku je nebo není pohyb. Já však potřebuji určit přesné místo, kde se pohyb nachází.

K určení místa pohybu používám velmi jednoduchou metodu. Matici rozdělím do menších oblastí, které označuji jako bloky. Pro každý blok popředí a pozadí vytvořím LBP histogram a histogramy porovnám. Pokud rozdíl histogramů je větší než nastavený práh, mohu konstatovat, že v bloku dochází k pohybu.

Abych byl schopen co nejpřesněji detekovat pohyb, je nutné určit vhodnou velikost bloku. Je nutné, aby blok nebyl příliš velký ani příliš malý. Pokud bude blok velký, nebude pohyb detekován přesně. Pokud bude blok malý, nebude obsahovat dostatek informací pro histogram, což bude v důsledku vést k chybné detekci. Pokusy jsem zjistil, že vhodná velikost bloku je $\frac{1}{10}$ velikosti podvzorkovaného snímku. Pokud má podvzorkovaný snímek velikost 320x240 (původní snímek měl velikost 640x480) bude velikost bloku 32x24. Blok bude obsahovat 768 LBP příznaků, což je dostačená hodnota pro porovnání. I pro podvzorkovaný snímek o velikosti 160x120, blok 16x12 obsahuje dostatek informací pro porovnání histogramů.

Na obrázku 20 vidíte výstup detektoru, který rozdělil snímek do bloků. Samotné rozdělení do bloků nestačí k přesné detekci pohybu. Abych detekci zpřesnil, využívám částečného překrytí bloků.

Obrázek 20: Rozdělení obrazu do bloků

Čím více se bloky překrývají, tím přesnější výstup dostávám, ale současně dochází k nárůstu výpočetní náročnosti. Na obrázcích 21 až 25 je vidět výsledek výstupu detektoru s různou mírou vzájemného překrytí bloků. Jako nejvhodnější mohu doporučit posuv o 4 pixely jak v řádcích, tak ve sloupcích (pokud má blok šířku 32 pixelů, bude se při posunu o 4 pixely s následujícím blokem překrývat na 28 pixelech). S nárůstem míry překrytu, lépe detekuji pohyb, ale narůstá i množství detekce neexistujícího pohybu při zachování stejného prahu detektoru. Odstranění tohoto problému popisuje kapitola 4.2.

Obrázek 21: Posuv o 16 řádků a 12 sloupců

Obrázek 22: Posuv o 8 řádků a 6 sloupců

Obrázek 23: Posuv o 4 řádky a 4 sloupce

Obrázek 24: Posuv o 2 řádky a 2 sloupce

Obrázek 25: Posuv o 1 řádek a 1 sloupec

Protože kromě přesné detekce požaduji i rychlé zpracování, je potřeba efektivně implementovat vytváření LBP histogramů bloků z matice s LBP příznaky. Protože se bloky překrývají z podstatné části (Obrázek 26), je neefektivní po porovnání zahodit histogram pro blok a znovu vytvořit nový histogram pro sousední blok. Při velkém překrytu je efektivnější, pokud jsou společné příznaky LBP

v histogramu zachovány, LBP příznaky unikátní pro původní blok odstraněny z histogramu a LBP příznaky unikátní pro aktuální blok do histogramu přidány.

Obrázek 26: Vzájemný překryt bloků.

Pokud bude velikost bloku 32x24, s posunem o 4 pixely, budou se histogramy překrývat v 28 sloupcích. Tudíž z histogramu odstraním 4 krát 24 LBP příznaků a přidám do histogramu 4 krát 24 LBP příznaků, což vyžaduje 192 přístupů do matice s LBP příznaky. Pokud bych vytvářel zcela nový histogram, potřeboval bych 768 přístupů do paměti, což je 4 krát více!

Vyhodnocování pohybu ve snímku provádím tak, že vypočtu LBP histogram pro první blok na řádku. Posouvám se po řádku a vytvářím histogram pro aktuální blok výše uvedeným způsobem. Při přechodu na nový řádek původní LBP histogram zahodím a vytvořím zcela nový histogram.

5.3 Využití diference pixelů v HSV pro zpřesnění detekce pohybu

Jak je patrné na obrázku 27, detekce pomocí LBP příznaků s využitím bloků nenajde přesné hranice objektu. Pokud dostatečně velká část objektu zasahuje do bloku, je za možnou oblast pohybu označen celý blok. Tudíž i ta část, která není pohybujícím se objektem.

Abych našel přesné hranice objektu, rozhodl jsem se využít diference pixelů. Diferenci provádím v HSV barevném prostoru (kapitola 3.4). Protože obraz mám uložen v HSV barevném prostoru a k odstranění stínů (kapitola 4.4) také využívám HSV barevnou reprezentaci obrazu, převod do jiného barevného modelu by zpomaloval vyhodnocení.

Rozdíl mezi pixely popředí a pozadí určuji pouze ve složkách hue a saturation. Pokusy jsem zjistil, že pokud je složka value zahrnuta do výpočtu, způsobuje zhoršení detekce. Celkový rozdíl mezi pixely popředí a pozadí počítám jako Eulerovu vzdálenost:

$$DIS = \sqrt{(F_H - B_H)^2 + (F_S - B_S)^2} \qquad (18)$$

kde F je hodnota pixelu popředí, B hodnota pixelu pozadí, H je hue , S saturation a DIS celkový rozdíl.

Diferenci provádím pouze v oblasti, ve které LBP detektor určil pohyb. Tím výrazně zkrátím čas vyhodnocení. Pokud mám u LBP detektoru nastaven nízký práh, mohu často i detekovat neexistující pohyb. Tento problém zapříčiňuje ve velké míře šum. Pokud bych nastavil práh LBP

detektoru na vysokou hodnotu, zbavím se sice šumu, ale zároveň nemusí být detekován skutečný pohyb. Vhodně nastaveným prahem u diference pixelů se zbavím neexistujícího pohybu, ale pohybující se objekty zůstanou zachovány. Na obrázku 28 je výstup z detektoru, který kombinuje LBP detekci a diferenci pixelů v HSV.

Obrázek 27: Pouze LBP detektor

Obrázek 28: LBP v kombinaci s diferencí

Ve své aplikaci používám aktualizaci pozadí založenou na *započtení snímku do pozadí s určitou vahou*. Aktualizaci pozadí provádím na úrovni podvzorkovaného snímku převedeného do HSV barevného modelu. I když mám vypočteny matice s LBP příznaky pro popředí a pozadí, nemohu aktualizovat přímo matici LBP příznaků pozadí. Důvodem proč nemohu aktualizovat přímo LBP matici, jsou vlastnosti LBP příznaků. LBP příznak je určitým popisem okolí pixelu. Pokud se změní nějaký pixel v okolí, změní se i LBP hodnota. Protože změna LBP hodnoty je nelineární (pokud se změní pixel v pravém dolním rohu matice koeficientů, bude LBP hodnota úplně jiná, než když se změní pixel v levém horním rohu), nemohu aktualizovat přímo LBP matice.

5.4 Odstranění stínů

Pro odstranění stínů jsem zvolil diferenci v HSV barevném prostoru popsanou v kapitole 4.3. Důvodem této volby byla především rychlost metody, její dobré výsledky v exteriérních scénách (uvedeny v [9]) a to, že obraz je již v HSV barevném prostoru.

Aby byl pixel označen za pixel obsahující stín, musí splňovat podmínku danou rovnicí 17. Tuto podmínku jsem ve své aplikaci mírně modifikoval, kvůli snazšímu nastavení prahů:

$$\left(\frac{I_k^V(x,y)}{B_k^V(x,y)} \leq \alpha \vee \frac{B_k^V(x,y)}{I_k^V(x,y)} \leq \beta\right) \wedge \left(I_k^S(x,y) - B_k^S(x,y)\right) \leq \tau_S \tag{19}$$

$$\wedge \left|I_k^H(x,y) - B_k^H(x,y)\right| \leq \tau_H$$

kde práh α uplatňuje u pixelů umbry, zatímco práh β odstraňuje pixely preumby.

Největším problémem bylo najít vhodné hodnoty prahů $\alpha, \beta, \tau_S, \tau_H$. Počet všech možných nastavení prahů je velmi velký. Nastavení prahů samotným uživatelem by bylo velmi obtížné. Rozhodl jsem se, že zjistím optimální hodnoty prahů a uživateli umožním měnit tyto optimální hodnoty.

Optimální hodnoty prahů jsem zjistil automaticky nad anotovanou sadou. Z několika video sekvencí jsem vybral snímky, v nichž jsem ručně vyznačil pohyb. První možností, kterou jsem pro automatické zjištění prahů vyzkoušel, bylo postupně automaticky měnit hodnoty prahů s určitým krokem. Pro každou kombinaci prahů jsem uložil počet pixelů správně označených jako pixely s pohybem (hits), počet pixelů, ve kterých nebyl pohyb detekován (miss) a počet pixelů, kde byl pohyb detekován klamně (fails). Za nejlepší řešení jsem považoval řešení, kde:

$$max\left(e^{\frac{hits}{miss+fails}}\right) \tag{20}$$

Problémem tohoto postupu je pevný krok řešení. Pokud je krok příliš velký nemusím najít optimální řešení. Pokud je však krok příliš malý, je počet zkoumaných kombinací velmi velký a tímto extrémně narůstá časová náročnost hledání prahu.

Pro nalezení optimální hodnoty prahů jsem se rozhodl použít evoluční strategii. Evoluční strategie je optimalizační metoda vhodná k nalezení řešení komplikovaných úloh s mnoha parametry. Vzorem této metody je evoluce v přírodě. Jedinec zde představuje hodnoty prahů. Jedinci se mezi sebou vzájemně kříží (kombinují vzájemně parametry) a mutují (přičítají k parametrům náhodná čísla daná střední odchylkou normálním rozložením). Z jedné generace jsou vybráni nejlepší jedinci, kteří představují další populaci rodičů, z nichž prostřednictvím mutace a křížení vnikají opět noví jedinci. V každé následující generaci se objevuje více kvalitnějších jedinců. Kvalita jedince se určuje pomocí fitness funkce. V mém případě představuje fitness funkci $e^{\frac{hits}{miss+fails}}$.

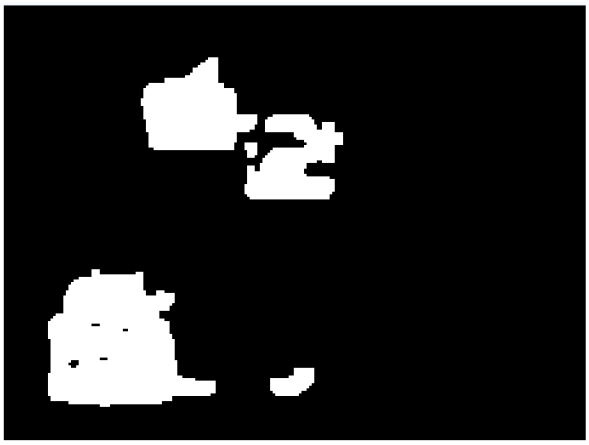
Evoluci je možno řídit pomocí střední odchylky normálního rozložení. Střední odchylka normálního rozložení určuje, jak velká bude změna parametru jedince. Pokud je střední odchylka velká, dochází k velkým změnám parametru, čímž se prohledává větší prostor řešení. Pokud je střední odchylka malá, je změna parametru menší, díky čemuž se hledá přesnější řešení v menším prostoru. V prvních generacích je střední odchylka velká a dochází k hledání míst, kde by mohla nacházet kvalitní řešení. S rostoucím počtem generací se parametry mění pozvolna a dochází ke zpřesňování řešení.

Díky evoluční strategii jsem nalezl optimální práh mnohem rychleji a kvalitněji, než když jsem použil enumeraci.

5.5 Označení jednotlivých objektů

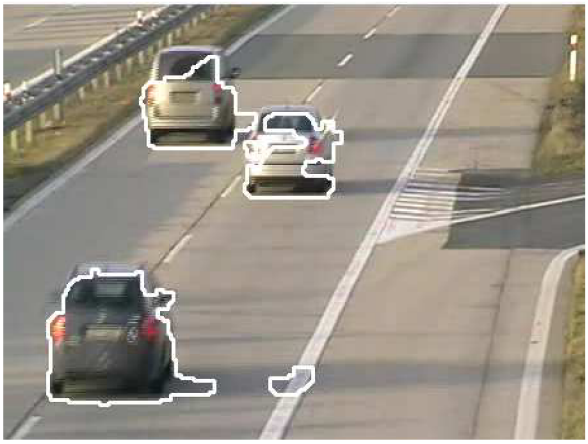
Výstupem detekční části je obraz, ve kterém jsou vyznačeny pixely s pohybem. Není zde žádná informace o objektech. Informaci o objektech musí získat uživatel pozorováním. Pro další zpracování je nutné spojit pixely s pohybem do objektů. Ke spojování využívám algoritmus pro detekci kontur. Algoritmus detekce kontur pracuje s binárním obrazem (Obrázek 30). Detekuji pouze okraje objektů. Pokud je v objektu díra, tuto díru nepovažuji za objekt, ale zahrnu ji do objektu, uvnitř kterého se nachází.

Obrázek 29: Obraz s detekovaným pohybem

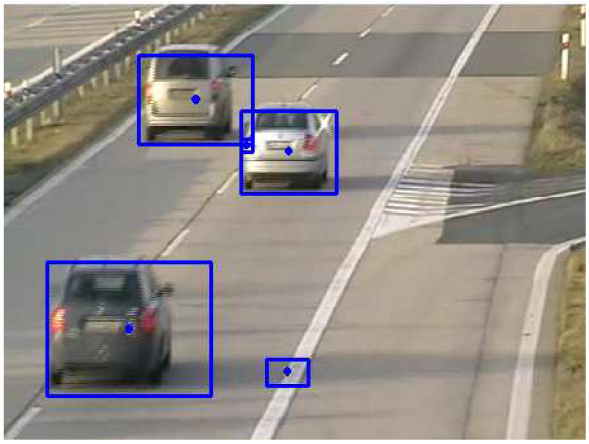


Obrázek 30: Binární obraz s pohybem

Výstupem algoritmu jsou jednotlivé objekty. Objekt je reprezentován seznamem přímek, které tvoří jeho hranice (Obrázek 31). Pro jednodušší práci s objekty nepracuji přímo s hranicemi objektu, ale vytvořím okolo objektu obdélníkový bounding box (Obrázek 32).



Obrázek 31: Kontury objektů



Obrázek 32: Bounding boxy objektů

5.6 Tracking

V této části opouštím od obecného řešení a zaměřuji se na detekci automobilů.

Nyní, když mám nalezeny jednotlivé objekty, které jsou reprezentovány bounding boxy, se mohu pokusit najít, které objekty si v různých snímcích video sekvence odpovídají. Shodnost dvou

objektů označuji jako korespondenci. Objekty, které byly nalezeny v aktuálním snímku, označuji jako aktuální objekty. Objekty, které byly nalezeny v předchozích snímcích, označuji jako předchozí objekty.

V prvním kroku provedu předzpracování. Předzpracování probíhá pouze na objektech nalezených v aktuálním snímku.

Objekty, které jsou do sebe vnořeny, se z velké části překrývají nebo jsou velmi blízko sebe, jsou spojeny do jednoho objektu. Spojování objektů probíhá na základě dvou podmínek.

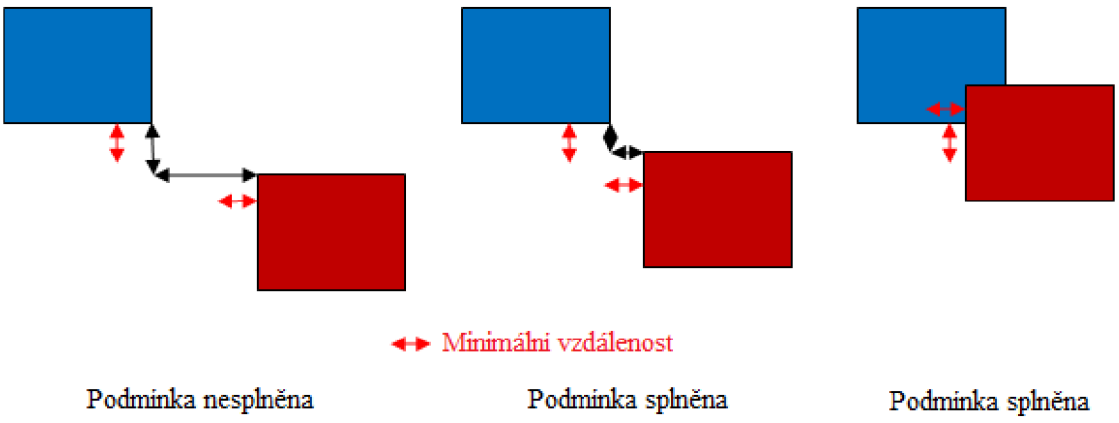
První podmínkou je, že jejich vzdálenost v ose x i ose y je menší než zadaná minimální vzdálenost (Obrázek 33). Druhou podmínkou, kterou musí splnit je poměr překrytí. Pokud se objekty v ose x nebo ose y překrývají více než je minimální hodnota, jsou spojeny do jednoho objektu (Obrázek 34). Poměr překrytí se určuje následujícím vzorcem:

$$mp = \frac{1}{pixelSmaller} pixelOverlap \tag{21}$$

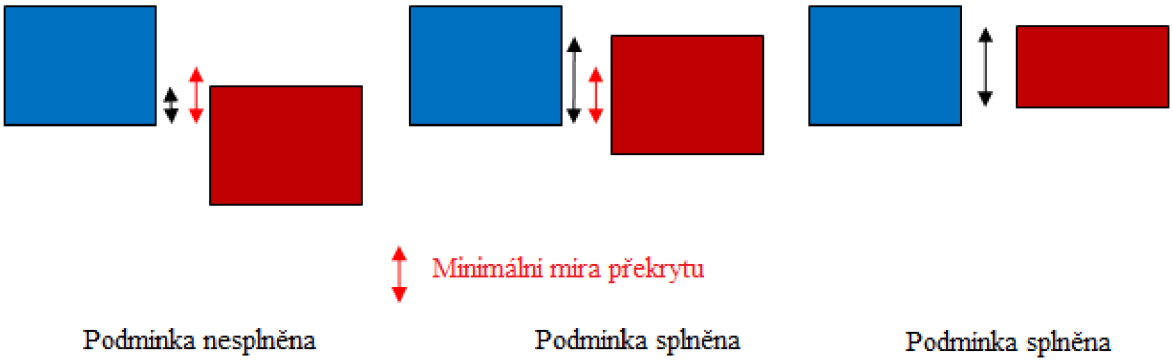
kde pixelOverlap je počet pixelů, kterými se objekty překrývají a pixelSmaller je velikost menšího objektu v pixelech. Míra překrytu se pohybuje v intervalu $< 0{,}1 >$. Při nulové míře překrytu se objekty vůbec nepřekrývají, při míře překrytu jedna je menší objekt zcela překryt.

U vnořených objektů je první i druhá podmínka splněna automaticky, tudíž vnořené objekty jsou automaticky spojeny s objektem, do něhož jsou vnořeny. U objektů, které se částečně překrývají, rozhoduje o spojení pouze druhá podmínka. Blízké objekty jsou spojeny do jednoho objektu, pokud jsou splněny obě podmínky.

Výsledkem spojení objektů je nový bounding box. Výsledky vidíte na obrázku 35.



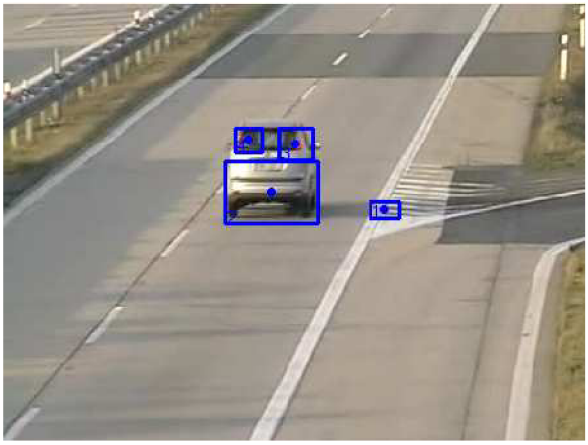
Obrázek 33: První podmínka pro spojení objektů



Obrázek 34: Druhá podmínka pro spojení objektů

Obrázek 35: Nové bounding boxy

Dalším krokem předzpracování je odstranění malých objektů. Objekty, které pokrývají méně pixelů, než je nastavený práh, jsou odstraněny z dalšího zpracování. V posledním stupni předzpracování jsou odstraněny objekty, které mají velmi rozdílnou výšku a šířku. Jak tyto parametry ovlivňují trakting objektů se dovíte v kapitole 6.2.



Obrázek 36: Vstup trackeru

Obrázek 37: Výstup po předzpracování

Po předzpracování následuje samotné hledání korespondujících objektů. Porovnání aktuálního objektu s předchozím objektem probíhá na základě Eulerovy vzdálenosti středů objektů a na základě velikosti objektu.

Pokud jsou středy objektů od sebe vzdáleny více než nastavená maximální vzdálenost je hodnota korespondence rovna nule a vyhodnocení korespondence na základě velikosti se neprovádí. Pokud je výsledek korespondence vzdálenosti nenulový, je provedeno vyhodnocení korespondence na

základě velikosti. Korespondence na základě velikosti je dána poměrem počtu pixelů, které objekty zabírají. Výsledná korespondence se dá vyjádřit vztahem:

$$kor = \begin{cases} 0 \; pokud \; \sqrt{(x_{new} - x_{old})^2 + (y_{new} - y_{old})^2} > maxDistance \\ \dfrac{1}{\sqrt{(x_{new} - x_{old})^2 + (y_{new} - y_{old})^2} + 1 + \dfrac{pixelLarger}{pixelSmaller}} \; jinak \end{cases} \tag{22}$$

kde x_{new}, y_{new} je střed aktuálního objektu, x_{old}, y_{old} je střed předchozího objetu, $pixelLarger$ je počet pixelů většího objektu, $pixelSmaller$ je počet pixelů menšího objektu.

Jak je z výrazu 22 patrné, hodnota korespondence dvou objektů se pohybuje v intervalu $< 0,1 >$. Pokud je hodnota rovna nule, nejsou si objekty vůbec podobné. Čím více se hodnota korespondence blíží jedné, tím více jsou si blíže středy objektů a objekty mají i podobnou velikost.

Aktuální objekt je porovnán se všemi předchozími objekty. Pokud je hodnota korespondence nenulová, aktuální objekt si uloží hodnotu korespondence s předchozím objektem a identifikátor předchozího objektu. Po vyhodnocení korespondence pro všechny aktuální objekty se pokusím určit, které aktuální a předchozí objekty si ve snímku vzájemně odpovídají. K nalezení odpovídajících si objektů používám algoritmus stabilního spárování [11]. Každý aktuální objekt pošle nabídku předchozímu objektu, se kterým má nejvyšší hodnotu korespondence. Každý předchozí objekt, který obdržel více nabídek, si ponechá pouze nabídku maximální, ostatní odmítne. Aktuální objekty, které byly odmítnuty, zašlou nabídku nejpreferovanějšímu předchozímu objektu, který je ještě neodmítl. Předchozí objekt, který obdrží více nabídek (mezi nabídky zahrne i ponechanou nabídku z minulého kola), si opět ponechá maximální nabídku aktuálního objektu. Postup se opakuje, dokud všechny aktuální objekty nevytvoří dvojici s předchozím objektem nebo nejsou vyčerpány všechny nabídky.

Pokud byl pro aktuální objekt nalezen předchozí objekt, je předchozímu objektu aktualizován jeho výskyt. Pokud se pro aktuální objekt nepodařilo najít vhodný předchozí objekt, je přidán objektu identifikátor a je uložen do předchozích objektů. Předchozí objekty, které nenašly odpovídající aktuální objekt, zůstávají v seznamu předchozích objektů, po určitý počet snímků. Může nastat situace, kdy kvůli chybě při detekci není objekt pro velmi malý počet snímků detekován. Pokud bych odstranil ze seznamu předchozích objektů objekt okamžitě, došlo by při opětovné detekci k vytvoření objektu s jiným identifikátorem, což je nežádoucí. Pokud by však objekty zůstávaly v seznamu předchozích objektů příliš dlouho, aniž by se spárovaly s nějakým aktuálním objektem, byly by nově se vyskytující objekty považovány za objekty, které se vyskytly již někdy v předchozích snímcích.

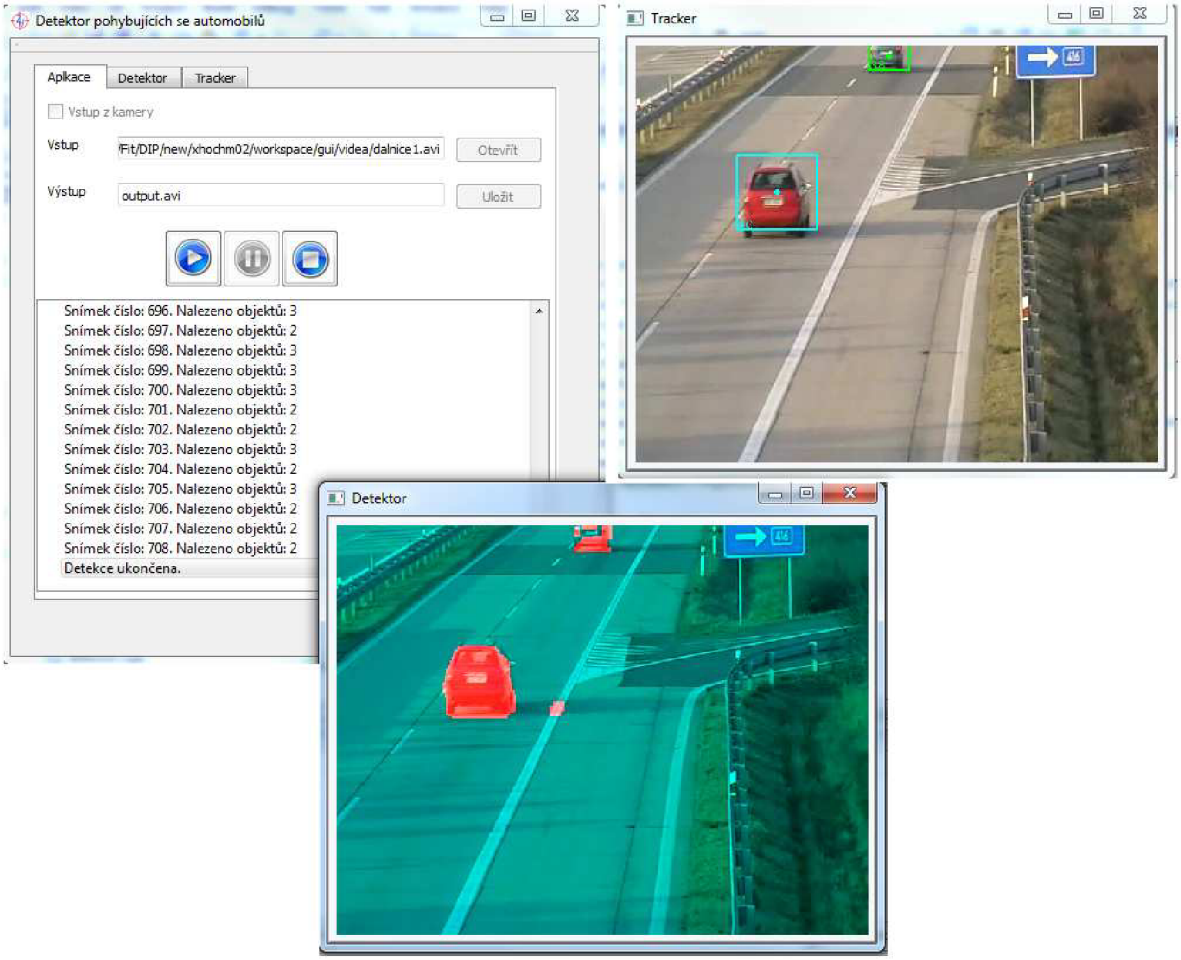
Aktualizace výskytu objektu využívá současnou polohu a velikost, ale i predikované hodnoty polohy a velikosti. Predikci provádím pomocí Kalmanova filtru [12]. Kalmanův filtr pomocí diferenciálních rovnic odhaduje na základě předchozích stavů stav následující. Každý objekt má svůj Kalmanův filtr. Filtr se inicializuje polohou a velikostí, kterou má objekt při svém vzniku. Pokaždé když je objekt nalezen v aktuálním snímku (je spárován s některým aktuálním objektem), je uložena nová pozice výskytu, která je dána skutečnou pozicí a velikostí a predikovanou pozicí a velikostí.

Míru, s jakou se ve výsledku uplatní predikované hodnoty, určuje míra korespondence. Pokud je míra korespondence vysoká, uplatňují se predikované hodnoty méně. Skutečné hodnoty nakonec předám Kalmanovu filtru, který pomocí nich provede korekci svého stavu a snaží se tak zkvalitnit další odhad.

6 Implementace

Aplikaci pro detekci pohybujících se automobilů jsem implementoval v jazyce C++. Jako vývojové prostředí jsem použil Microsoft Visual Studio 2008. V aplikaci velmi využívám knihovny OpenCV 2.0 [13]. Knihovna OpenCV je určena pro práci s obrazem a obsahuje i některé algoritmy počítačového vidění. Knihovna je implementována velmi efektivně, a při zpracování obrazu dosahuje vysokého výkonu. Knihovna je implementována v jazyce C, která od verze 2.0 poskytuje i rozhraní pro C++.

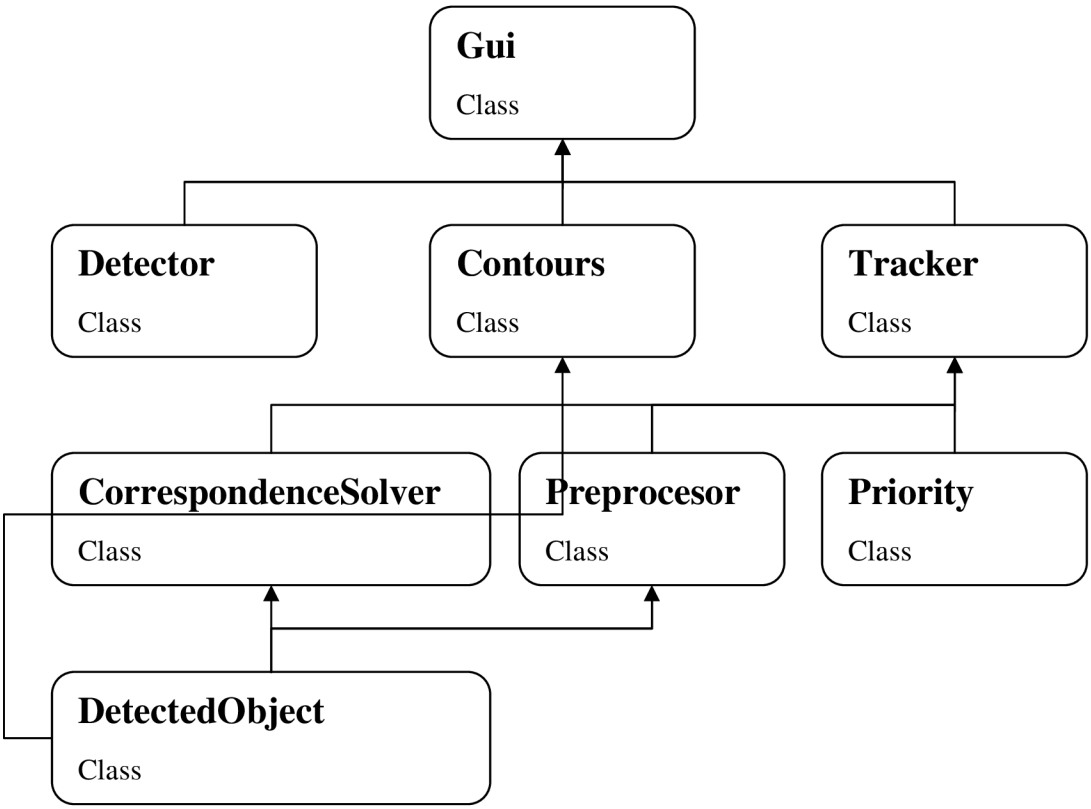
Detekci a sledování pohybujících se objektů ovlivňuje množství parametrů. Tyto parametry se mohou v závislosti na scéně, ve které sledování provádím, měnit. Chci umožnit uživateli tyto parametry měnit. Protože ovládání z příkazové řádky by bylo pro uživatele komplikované, a navíc by nebyla možná změna parametrů za běhu, rozhodl jsem se pro ovládání aplikace vytvořit grafické rozhraní. K tvorbě grafického rozhraní používám knihovnu Qt verze 4.6.2 [15]. Tato knihovna je napsána v C++ a kompatibilní jak s operačním systémem Windows tak Linux. Ovládání aplikace popisuji v příloze.



Obrázek 38: Aplikace pro detekci pohybujících se automobilů

6.1 Třídy aplikace

Aplikaci jsem rozdělil do několika tříd (Obrázek 39). Snažil jsem se, aby každá třída zapouzdřovala jednu část aplikace. Parametry a metody tříd jsou popsány v programové dokumentaci, která je součástí elektronické přílohy. Detaily implementace lze najít ve zdrojových textech, které jsou také součástí elektronické přílohy.



Obrázek 39: Třídy a jejich vztahy (šípky nereprezentují dědičnost)

Třida Gui

Tato třída představuje grafické rozhraní aplikace. Stará se o načítání a ukládání snímků video sekvence, informování ostatních tříd o změně parametrů, předávání výsledků mezi třídami a zobrazování výsledků.

Třída Detector

Třída Detector se zabývá detekcí pohybu. Třída udržuje jako svůj vnitřní stav model pozadí. Vstupem třídy je snímek v barevném prostoru RGB. Snímek je předzpracován (Kapitola 5.1), následně jsou vypočteny LBP příznaky. V dalším kroku je provedena hrubá detekce pohybu pomocí porovnání LBP histogramů bloků popředí a pozadí (Kapitola 5.2). V pixelech, ve kterých byl detekován pohyb, je provedena jemná detekce pomocí diference pixelů v HSV barevném prostoru (Kapitola 5.3). Na pixelech, které byly při jemné detekci označeny jako pixely s pohybem, se provede kontrola na přítomnost stínu (Kapitola 5.4). Nakonec třída aktualizuje svůj model pozadí. Detektor vrací binární obraz s detekovaným pohybem. Obraz je zvětšen na původní velikost.

Třída Contours

Vstupem této třídy je binární obraz s detekovaným pohybem. Třída pomocí algoritmu pro detekci kontur nalezne spojité oblasti. Pro spojité oblasti pak vytvoří bounding boxy. Pro každý bounding box vytvoří objekt. Třída vrací seznam detekovaných objektů.

Třída Tracker

Tato třída se zabývá sledováním průchodu objektů scénou. Udržuje seznam objektů, které se vyskytly v předchozích snímcích. Snaží se spárovat objekty, které byly nalezeny v aktuálním snímku, s objekty v předchozích snímcích. Vstupem třídy je seznam objektů detekovaných v aktuálním snímku, výstupem je seznam objektů, které se mají vykreslit.

Třída preprocesor

Tato třída se využívá pro předzpracování objektů. Spojuje vnořené objekty, objekty, které se velkou mírou překrývají, a objekty, které se nacházejí blízko sebe. Také odstraňuje malé objekty a objekty s velmi rozdílným poměrem výšky a šířky.

Třída CorrespondenceSolver

Tato třída vypočte míru korespondence dvou objektů na základě polohy středů a jejich velikostí.

Třída Priority

Tato třída udržuje preference objektu. Udržuje seznam, ve kterém je obsaženo, s jakými objekty objekt koresponduje a jakou mírou s nimi koresponduje. Třída se využívá při párování objektů k posílání nabídek.

Třída DetectedObject

Třída představuje detekovaný objekt. Obsahuje seznam s výskyty objektů. Položka seznamu je tvořena číslem snímku, polohou středu, výškou a šířkou objektu. Objekt dále obsahuje Kalmanův filtr, který predikuje polohu objektu v následujícím snímku.

6.2 Paralelní zpracování

Ve své práci kladu velký důraz na rychlost zpracování. Protože v dnešní době většina počítačů obsahuje více jádrové procesory, je vhodné využít všech jader ke zpracování úlohy. Některé části úlohy se nedají zpracovávat paralelně, naopak jiné části lze zpracovat paralelně velmi snadno.

Ve své aplikaci jsem se rozhodl paralelizovat části, ve kterých se pracuje s obrazem. Konkrétně se jedná o podvzorkování obrazu, výpočet LBP příznaků, porovnávání LBP histogramů bloků, detekci pohybu pomocí diference pixelů v HSV barevném prostoru, detekci stínů a převod obrazu na původní velikost.

K paralelizaci jsem použil knihovnu OpenMP. Tato knihovna v sobě obsahuje direktivy a knihovní funkce. Pomocí direktiv určuji, která část programu se bude zpracovávat sekvenčně a která paralelně. Díky tomuto mohu provádět paralelizaci inkrementálně (paralelizuji postupně části kódu). Knihovní funkce umožňují například zjistit počet procesorů (jader), nastavit počet vláken, zjistit počet vláken provádějících zpracování.

Pokud překladač nezná direktivy OpenMP nebo se nemá překládat do paralelní verze (například kvůli ladění programu), překladač ignoruje direktivy OpenMP a kód je přeložen do sekvenční verze. Volání funkcí OpenMP je vhodné ohraničit direktivou ifdef _OPENMP. Tato

direktiva způsobí, že příkazy obsažené v tomto bloku se vykonají, pouze pokud se překládá za pomoci OpenMP.

Pokud je vláken více, než kolik mohou jádra naráz zpracovávat, dochází k přepínání vláken. Přepnutí vlákna stojí výpočetní výkon a zpomaluje se tím zpracování. Dnešní procesory mohou na jednom jádru zpracovávat současně jedno nebo dvě vlákna (záleží na typu procesoru). Ve své aplikaci nastavuji počet vláken, která paralelně zpracovávají program, na počet jader.

7 Experimentální výsledky

7.1 Hodnoty prahů pro odstranění stínů

Jak jsem uvedl v kapitole 5.4, pro odstranění stínů jsem využil evoluční strategie. Pro vyhledání optimálního prahu jsem využil čtyři anotované sady z různých video sekvencí.

Obrázek 40: Příklad anotované sady - vstupní obraz, model pozadí v HSV, anotovaný binární obraz.

Nejdříve jsem hledal optimální prahy nad každou sadou samostatně, následně jsem hledal optimální práh pro všechny sady.

Sada	Hue	Saturation	Value I/B	Value B/I	Fittness
1	73	-17	0,181	1,829	7,993
2	75	-17	0,197	1,875	8,135
3	69	-30	0,251	1,917	25,046
4	69	-28	0,301	1,918	24,871
Všechny	73	-20	0,439	1,617	6,425

Tabulka 1: Hodnoty prahů zjištěných pomocí evoluční strategie

Pro srovnání výsledků evoluční strategie jsem vyhodnotil několik hodnot prahů, které jsem zvolil. Vyhodnocení jsem prováděl nad všemi anotovanými sadami.

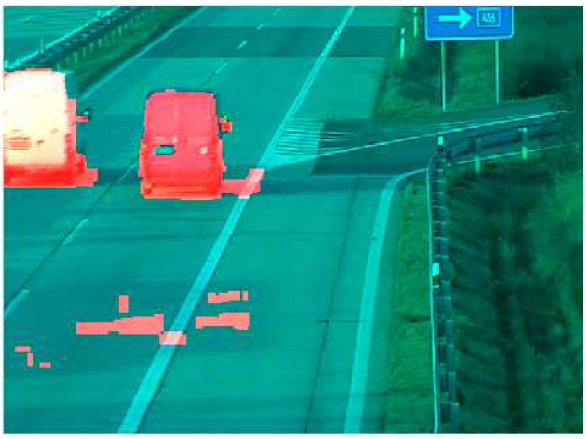
Hue	Saturation	Value I/B	Value B/I	Fitness
70	20	0,439	1,617	3,451
70	0	0,439	1,617	4,992
70	-30	0,439	1,617	5,031
60	-20	0,439	1,617	6,168
80	-20	0,439	1,617	6,168
70	-20	1,000	1,617	6,400
70	-20	0,439	1,000	6,175

Tabulka 2: Ručně nastavené prahy.

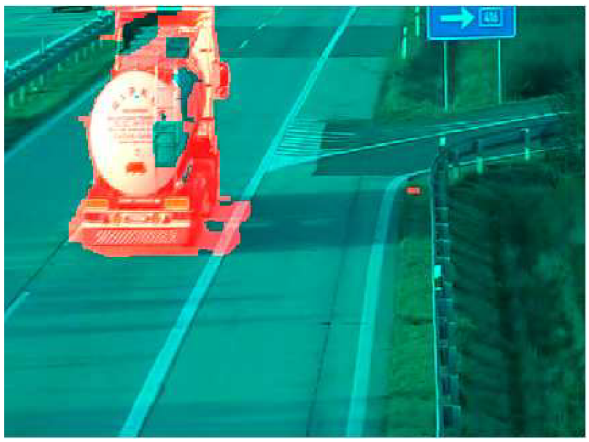
Pro ověření kvality nalezených prahů jsem provedl validaci na zdrojových video sekvencích. Jako metriku jsem použil vizuální porovnání. Na obrázcích 41 až 44 vidíte, jak kvalitně jsou odstraněny stíny při použití prahů nalezených evoluční strategií.

Obrázek 41: Kvalitní odstranění stínů

Obrázek 42: Odstranění stínů s drobnými vadami



Obrázek 43: Špatně odstraněný stín



Obrázek 44: Odstraněny velké části objektu

7.2 Postprocesing obrazu pomocí trackeru

Jak vidíte na obrázcích 42 až 44, výstup detektoru není vždy ideální. Díry v objektu odstraní algoritmus pro detekci kontur. Avšak s vadami jako je chybně detekovaný stín nebo rozdělení objektu na více částí se musí vyrovnat tracker. K odstranění vad je v trackeru určena část zabývající se předzpracováním.

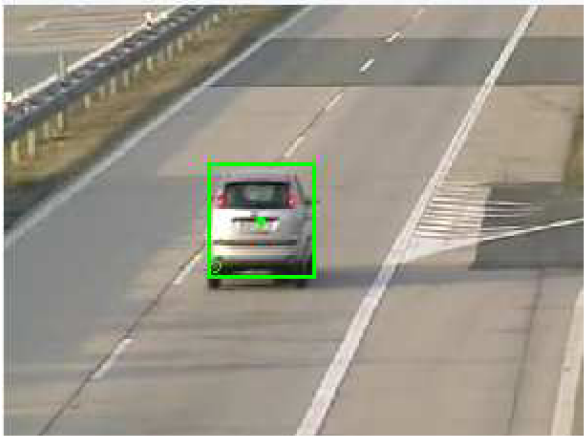
Slučování objektů:

Slučování objektů lze ovlivnit třemi parametry. Nejdůležitějším parametrem je vzdálenost objektů od sebe. Dalšími parametry jsou míra překrytí v ose x a míra překrytí v ose y.

Pokud je parametr pro minimální vzdálenost objektů nastaven na velkou hodnotu, může docházet k nevhodnému spojování objektů (Obrázek 47). Pokud je hodnota velmi malá, nemusí být

objekt, který byl detekcí stínů roztříštěn, spojen do jednoho objektu. Experimenty jsem zjistil, že pro většinu případů je vhodná hodnota mezi 5 až 10 pixely.

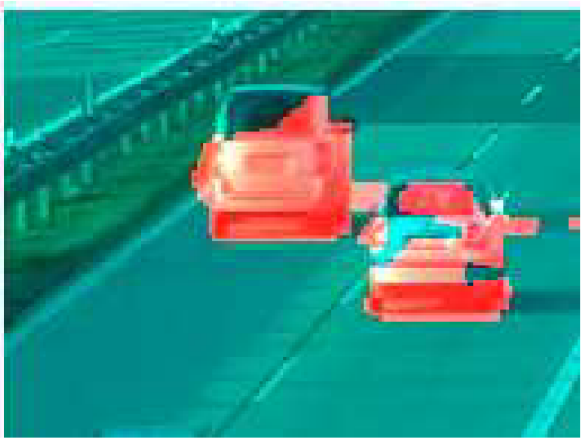
Obrázek 45: Výstup detektoru



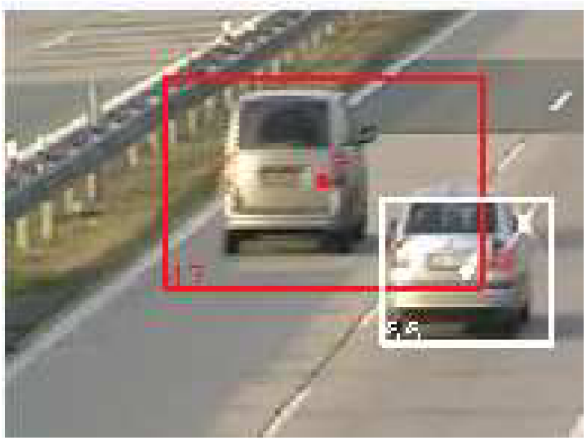
Obrázek 46: Vzdálenost objektů od sebe nastavena na 10 pixelů.

Obrázek 47: Vzdálenost objektů od sebe nastavena na 100 pixelů. Dochází k nevhodnému spojení objektů.

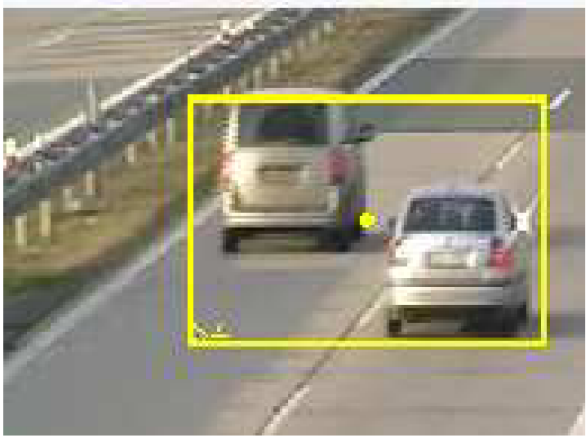
Vhodné hodnoty pro míru překrytí jsou v rozmezí 0,6 až 0,9. Pokud je míra překrytí malá, dochází ke spojování objektů, které se málo překrývají. Toto může vést ke spojení dvou odlišných objektů (Obrázek 50).



Obrázek 48: Výstup detektoru



Obrázek 49: Správně označené objekty



Obrázek 50: Špatně označené objekty. Objekty byly spojeny do jednoho objektu, protože míra překrytu byla nastavena pouze na 0,4.

Odstranění malých objektů:

Zbytky stínů mohou být detekovány jako objekty. Takovýto objekt většinou pokrývá malý počet pixelů. Při odstranění malých objektů se bere v úvahu, kolik pixelů pokrývá bounding box objektu nikoliv objekt samotný.

Při jakém počtu pixelů, lze považovat objekt za dostatečně velký, záleží především na zpracovávané video sekvenci. Pokud jsou automobily ve video sekvenci malé (kamera byla daleko od snímané silnice), bude nutné nastavit počet pixelů na menší hodnotu, aby nedocházelo k nechtěnému odstranění skutečných objektů. Naopak pokud automobily pokrývají velký počet pixelů, tak i pozůstatky stínů budou pokrývat větší počet pixelů.

Dalším parametrem ovlivňujícím toto nastavení je samotné rozlišení video sekvence. Pro video s rozlišením 640 x 480 doporučuji jako minimální hodnotu 300 pixelů.

Odstranění objektů se špatným poměrem výšky a šířky:

Některé zbytky stínů pokrývají více pixelů, a proto nejsou odstraněny. Tento problém se pokouším řešit pomocí poměru výšky a šířky. Vycházím z předpokladu, že neexistují auta, která jsou velmi vysoká a úzká nebo nízká a široká.

Jak jsem při testování zjistil, tento předpoklad není zcela správný. Jednou z možností porušení tohoto předpokladu je chyba při detekci. Ale daleko častěji je tato podmínka porušena při vstupu nebo výstupu automobilu do scény. V tomto okamžiku je ve scéně detekována pouze část automobilu, a předpoklad o poměru výšky a šířky přestává platit.

Obrázek 51: Chybné odstranění objektů na základě poměru výšky a šířky

Abych potlačil tento problém, rozhodl jsem se při odstranění prověřit opětovně velikost objektu. Objekt je ze zpracování odstraněn, pokud má špatný poměr výšky a šířky a zároveň pokrývá plochu menší než je desetinásobek minimální plochy pro objekt. Toto řešení však není zcela dokonalé a problém pouze zmírňuje.

7.3 Vyhodnocení úspěšnosti trackingu

Testování jsem provedl nad sadou snímků z různých video sekvencí. V kompletní sadě se vyskytují i snímky, kde jsou auta velmi vzdálená od kamery, překrývají se, a obecně lze říci jen velmi těžko detekovatelná. Do druhé sady jsem z kompletní sady vybral snímky, kde se vyskytují objekty dobře vzájemně separovatelné a v rozumné vzdálenosti od kamery. Příklady snímků vidíte na obrázcích 52 a 53.

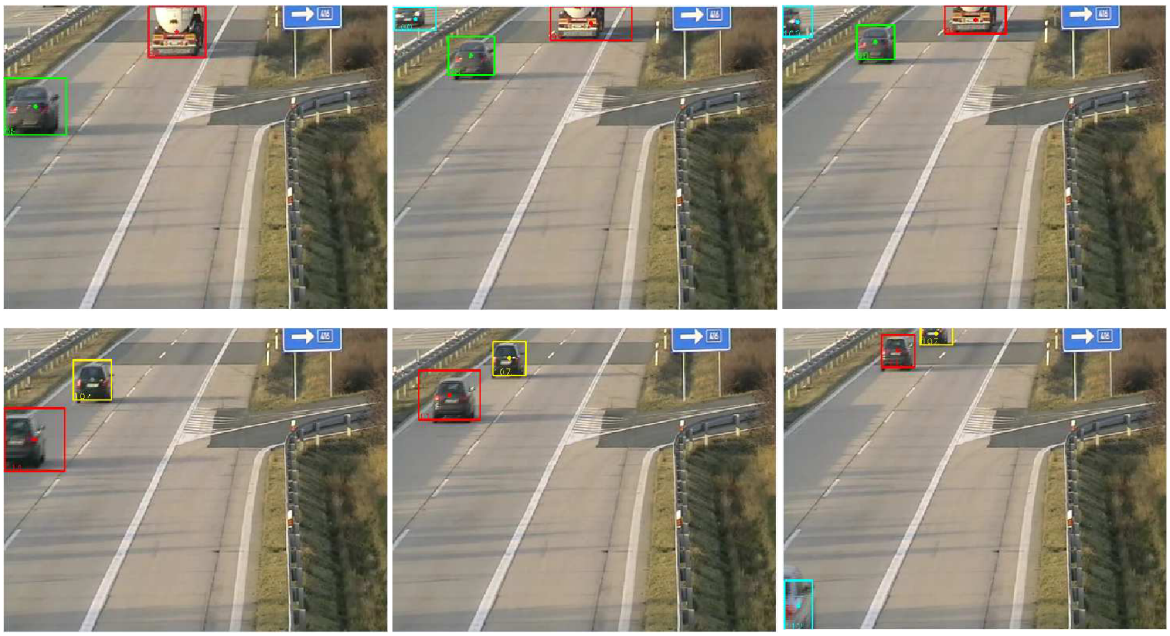
Obrázek 52: Dobře seperavatelné objekty ve vhodné vzdálenosti

Obrázek 53: Velmi vzdálené automobily a vzájemně se překrývající automobily

Sada	Kompletní	Vybraná
Počet snímků v sadě	203	108
Správně detekované objekty	296	140
Posunutý střed objektu (Obrázek 55)	10	7
Chybná velikost (Obrázek 56)	54	31
Překrytí více objektů (Obrázek 57)	87	6
Nedetekovaný objekt (Obrázek 59)	96	18
Klamná detekce (Obrázek 60)	7	5
Úspěšnost detekce	60%	83%

Tabulka 3: Úspěšnost trackingu

Pokud se dva objekty nepřekryjí nebo nepřiblíží velmi blízko k sobě, jsou sledovány téměř se 100 % úspěšností (Obrázek 54).

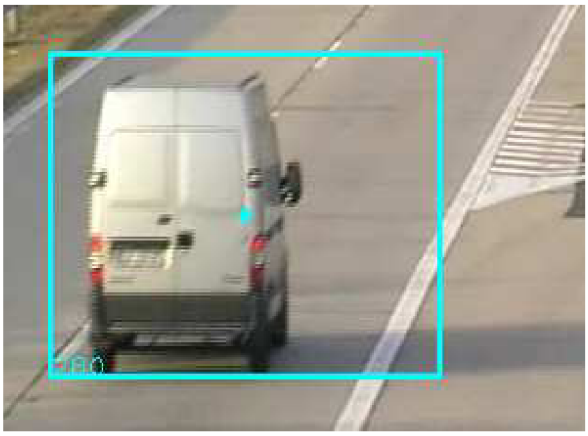


Obrázek 54: Výsledky trackingu

Posunutý střed objektu je vada, kdy se střed detekovaného objektu zcela jasně nachází mimo skutečný střed objektu, přičemž velikost detekovaného objektu odpovídá skutečné velikosti objektu. Posunutý střed objektů je způsoben špatně predikovanými hodnotami, které poskytuje Kalmanův filtr. Tato chyba se vyskytuje poměrně zřídka a je během několika snímků korigována.

Obrázek 55: Posunutý střed objektu

Chybná velikost objektu je vada, při které je velikost detekovaného objektu menší nebo větší než skutečná velikost objektu. Chybná velikost objektu je způsobena hlavně chybným odstraněním stínů. Ve většině případů není stín zcela odstraněn a jeho zbytek způsobí zvětšení velikosti objektu. Chybná velikost je v malé míře způsobena, podobně jako u posunutí středu objektu, špatně predikovanými hodnotami.



Obrázek 56: Chybná velikost objektu

Další chybou je vzájemné překrytí objektů ve scéně. Při této chybě tracker milně považuje více objektů za jeden objekt. Protože tracker nemá 3D informaci, je celkem problematické tento nedostatek odstranit. Odstranění by vyžadovalo buď zcela jiný přístup k detekci, například detekci hran a následné porovnání hran mezi aktuálním a referenčním snímkem, nebo dodatečné vyhodnocení části obrazu a blízkého okolí, kde byl detekován pohyb pomocí komplexnějšího algoritmu počítačového vidění (implicit shape model, skenování obrazu klasifikátorem).

Obrázek 57: Překrytí více objektů.

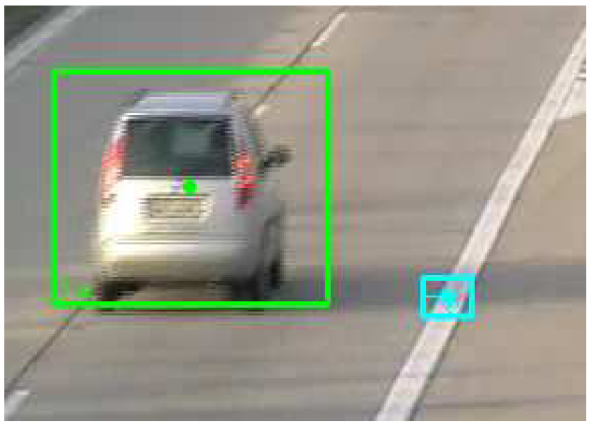
Nedetekovaný objekt je chyba, při které není trackerem skutečný objekt nalezen. Nedetekování objektu může být způsobeno více faktory. Prvním faktorem je, že objekt nemusí být vůbec zachycen detektorem pohybu. Dalším faktorem je odstranění malých objektů. Na obrázku 56 vidíte, že kamera snímá poměrně rozsáhlou oblast. Ve velké vzdálenosti od kamery jsou objekty detekovány, ale tracker s nimi již nepracuje, protože byly ze zpracování vyloučeny.

Obrázek 58: Výstup detektoru

Obrázek 59: Chybné odstranění skutečných objektů.

Posledním faktorem je odstranění objektu ze zpracování, kvůli nevhodnému poměru výšky a šířky. Tento problém nejčastěji nastává při vstupu objektů do scény.

Klamná detekce je chyba, při které je tracker nalezne ve skutečnosti neexistující objekt. Klamné detekce byly způsobeny stíny, které přežily jak odstranění pomocí detektoru tak odstranění pomocí předzpracovací části trackeru.



Obrázek 60: Klamně detekovaný objekt

U vybrané sady dosahuje detektor úspěšnosti kolem 82%, u kompletní sady 60%. Vyhodnocení jsem provedl nad sadou snímků, které jsem ručně vybral z video sekvence. Pokud bych vyhodnocení provedl nad celou video sekvencí, v kompletní sadě by se vyskytovalo více vhodných snímků a tím pádem by i výsledná úspěšnost byla lepší.

Další vyhodnocení jsem provedl vizuálně na video sekvencích. Vyvodil jsem následující závěry. Problémy občas nastávají pouze při vstupu a výstupu do scény. Pokud se objekty překryjí nebo se výrazně přiblíží, jsou spojeny do jednoho objektu. K rozdělení dochází, pokud se objekty dostatečně vzdálí. Jeden z objektů si zachová původní identifikátor, druhému objektu je přidělen jiný identifikátor.

Největší zhoršení detekce způsobují zbytky stínů. Jsou buď falešně detekovány jako objekty nebo způsobují spojování objektů do jednoho objektu.

7.4 Rychlost zpracování

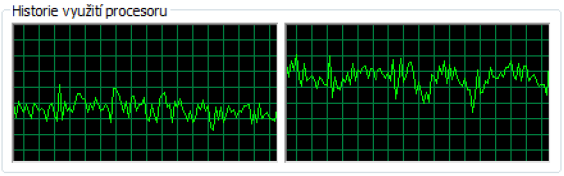
Ve své práci kladu velký důraz na rychlost zpracování. Ideální je zpracovat vstup v reálném čase. Dnes má většina video sekvencí 25 snímků za sekundu. Některé mají 30 snímků za sekundu. V tabulce 3 vidíte, jaké rychlosti zpracování dosahuje má aplikace. Výkon byl měřen na dvoujádrovém Intel Core 2 Duo s frekvencí 2,4 GHz, 4G RAM a operačním systémem Windows 7

Rozlišení	OpenMP	Snímků za sekundu	Zrychlení
640 x 480	Ano	14	1,27
640 x 480	Ne	11	
320 x 240	Ano	32	1,18
320 x 240	Ne	27	

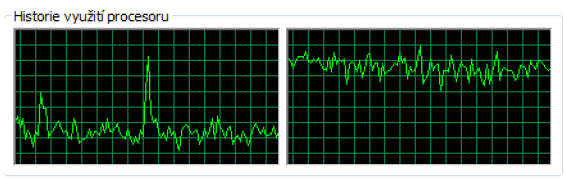
Tabulka 4: Urychlení pomocí paralelního zpracování.

Jak z tabulky 3 vyplývá, paralelní řešení urychlí čas zpracování. K dosažení vyšších hodnot zrychlení by bylo potřeba převést více sekvenčních částí na paralelní, jako je například hledání kontur a tracking.

Jak můžete vidět na obrázcích 61 a 62, kromě zvýšení rychlosti zpracování přináší paralelizace i lepší rozložení výkonu na jednotlivá jádra.



Obrázek 61: Zatížení jader při použití paralelního zpracování.



Obrázek 62: Zatížení jader při použití sekvenčního zpracování.

Závěr

Cílem diplomové práce bylo navrhnout vhodnou metodu pro sledování automobilů. Při návrhu jsem musel vyřešit mnoho problémů. Přesto mohu konstatovat, že zadaný cíl se podařilo naplnit. Tracker pohybujících se automobilů nedosahuje stoprocentní úspěšnosti. Pro zvýšení úspěšnosti trackeru je nutné pokračovat ve vývoji aplikace.

V teoretické části práce jsem popsal několik metod detekce pohybu. Největší pozornost jsem věnoval metodě Local Binary Patterns. Další stěžejní teoretickou částí práce je popis metod pro odstranění stínů, v níž jsem popsal několik rozdílných metod a nastínil jsem jejich výhody a nevýhody.

Detekci pohybu jsem rozdělil na hrubou a jemnou. Pro hrubou detekci využívám metodu LBP. V místech, kde byl hrubou detekcí nalezen pohyb, zpřesním místo pohybu pomocí diference pixelů v HSV barevném prostoru.

Při sledování pohybujících se objektů nám velmi vadí stín vrhaný objektem. Pro odstranění stínů opět využívám HSV barevný prostor. Pokud hodnota pixelu v jednotlivých složkách splní podmínku stínu, je obsah pixelu považován za stín. V podmínce se vyskytují čtyři prahy. Aby uživatel nemusel složitě prahy nastavovat, vyhledal jsem optimální hodnoty prahů pomocí evoluční strategie.

Ke spojení pixelů do jednotlivých objektů využívám algoritmus pro detekci kontur. Před samotným hledáním shodných objektů mezi jednotlivými snímky provedu předzpracování na úrovni objektů. Předzpracování spojí blízké a vnořené objekty, odstraní velmi malé objekty a objekty se špatným poměrem výšky a šířky.

Nalezení shodných objektů mezi jednotlivými snímky probíhá pomocí pozic středů a velikostí objektů. Pokud je objekt nalezen v aktuálním snímku i předchozím snímku, je jeho nová pozice určena nejen skutečnou pozicí, ale i predikovanou pozicí. Zahrnutí predikované pozice pomáhá odstranit chyby vniklé při detekci. Predikci provádím Kalmanovým filtrem.

V celé práci kladu velký důraz na rychlost zpracování. Kromě použití časově nenáročných metod, jsem vhodné části kódu paralelizoval. Při rozlišení video sekvence 640x480 jsem schopen vyhodnotit 14 snímků za sekundu. U video sekvence v rozlišení 320x240 dosahuji rychlosti 32 snímků za sekundu.

Pokud se ve video sekvenci objekty často vzájemně nepřekrývají a nevyskytují se malé objekty, pohybuje se úspěšnost detekce kolem 83 %. Pokud se však ve video sekvenci vyskytují objekty, které se často překrývají, a vyskytuje se více malých objektů, klesá úspěšnost detekce v závislosti na četnosti výskytu těchto objektů.

Pokud pominu překrývající se objekty a malé objekty, největší procento chyby připadá na chybnou velikost objektu. Tato chyba je nejčastěji způsobena špatným odstraněním stínů.

Pro dosažení vyšší úspěšnosti bych doporučil vyzkoušet jinou metodu pro odstranění stínů. Samotné zkvalitnění odstranění stínů by velmi zlepšilo úspěšnost trackru. Dále doporučuji pokračovat v rozvoji trackeru. Při hledání shody objektů využít další informace, jako například barvu objektu nebo LBP příznaky objektu. Vhodné by bylo zaměřit se na překrývající se automobily.

Literatura

[1] Hochman, Z.: Sledování pohybu osob ve video sekvenci. Brno, 2008. Dostupné na URL: <http://www.stud.fit.vutbr.cz/~xhochm02/lbp/xhochm02.pdf>

[2] Španěl, M.: Rozpoznávání gest ve video sekvencích. Brno, 2003. Dostupné na URL: <http://www.fit.vutbr.cz/~spanel/dp/spanel_dp_2003.pdf>

[3] Jelínek, T.: Detekce pohybujících se objektů ve video sekvenci. Brno, 2007. Dostupné na URL: <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=5197>

[4] Ojala, T., Pietikäinen, M., Mäenpaä, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Oulu, 2002. . Dostupné na URL: <http://www.ee.oulu.fi/research/imag/texture/publications/show_pdf.php?ID=94>

[5] Heikkilä, M., Pietikäinen, M., Heikkilä, J.: A texture-based method for detecting moving objects. Oulu, 2004. . Dostupné na URL: <http://www.ee.oulu.fi/mvg/publications/show_pdf.php?ID=533>

[6] Lee, D. C.: Boosted Classifier for Car Detection. Pittsburgh, 2008. Dostupné na URL: <http://www.cs.cmu.edu/~dclee/car_boosted.pdf>

[7] Wikipedie: HSV. 2010. Dostupné na URL: < http://cs.wikipedia.org/wiki/HSV>

[8] Španěl, M.: Chyby v obraze, typy šumu, restaurace obrazu a optimální filtrace. Brno, 2009.

[9] Pirati, A., Mikić, I., Trivedi, M.M., Cucchiara, R.: Detectin Moving Shadows: Formulation, Algorithms and Evaluation. Modena, 2001. Dostupné na URL: <http://cvrr.ucsd.edu/aton/publications/pdfpapers/TRshadow.pdf>

[10] Cucchiara, R., Grana, C., Piccardi, M., Prati, A., Sirotti, S.: Improving Shadow Suppression in Moving Object Detection with HSV Color Information. Modena, 2000. Dostupné na URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=948679&userType=inst>

[11] Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V.V.: Algorithmic Game Theory, Cambridge, 2007. Dostupné na URL: <http://www.cambridge.org/journals/nisan/downloads/nisan_non-printable.pdf>

[12] Kalmanův filtr. 2010. Dostupné na URL: <http://www.cs.unc.edu/~welch/kalman/>

[13] OpenCV. 2010. Dostupné na URL: <http://opencv.willowgarage.com/wiki/>

[14] Bradski, G., Kaehler, A.: Learning OpenCV. O'Reilly, 2008.

[15] Ot 4.6. 2010. Dostupné na URL: <http://doc.qt.nokia.com/4.6/index.html>

[16] Mersenne Twister Random Number Generator, Dostupné na URL: <http://www-personal.umich.edu/~wagnerr/MersenneTwister.html>

[17] Wikipedie: Neuronové sítě. 2010. Dostupné na URL: < http://en.wikipedia.org/wiki/Neural_network>

[18] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. CVPR, 2001. Dostupné na URL: <http://research.microsoft.com/en-us/um/people/viola/pubs/detect/violajones_cvpr2001.pdf>

[19] Wikipedie: Support vector machine. 2010. Dostupné na URL: <http://en.wikipedia.org/wiki/Support_vector_machine>

[20] Wikipedie Relevance vector machine. 2010. Dostupné na URL: <http://en.wikipedia.org/wiki/Relevance_vector_machine>

[21] Horprasert, T., Harwood, D., Davis, L.S.: A statistical approach for real-time robust background subtraction and shadow detection, Proceedings of IEEE ICCV'99 FRAME-RATE Workshop, 1999. Dostupné na URL: <http://vast.uccs.edu/~tboult/FRAME/Horprasert/HorprasertFRAME99.pdf>

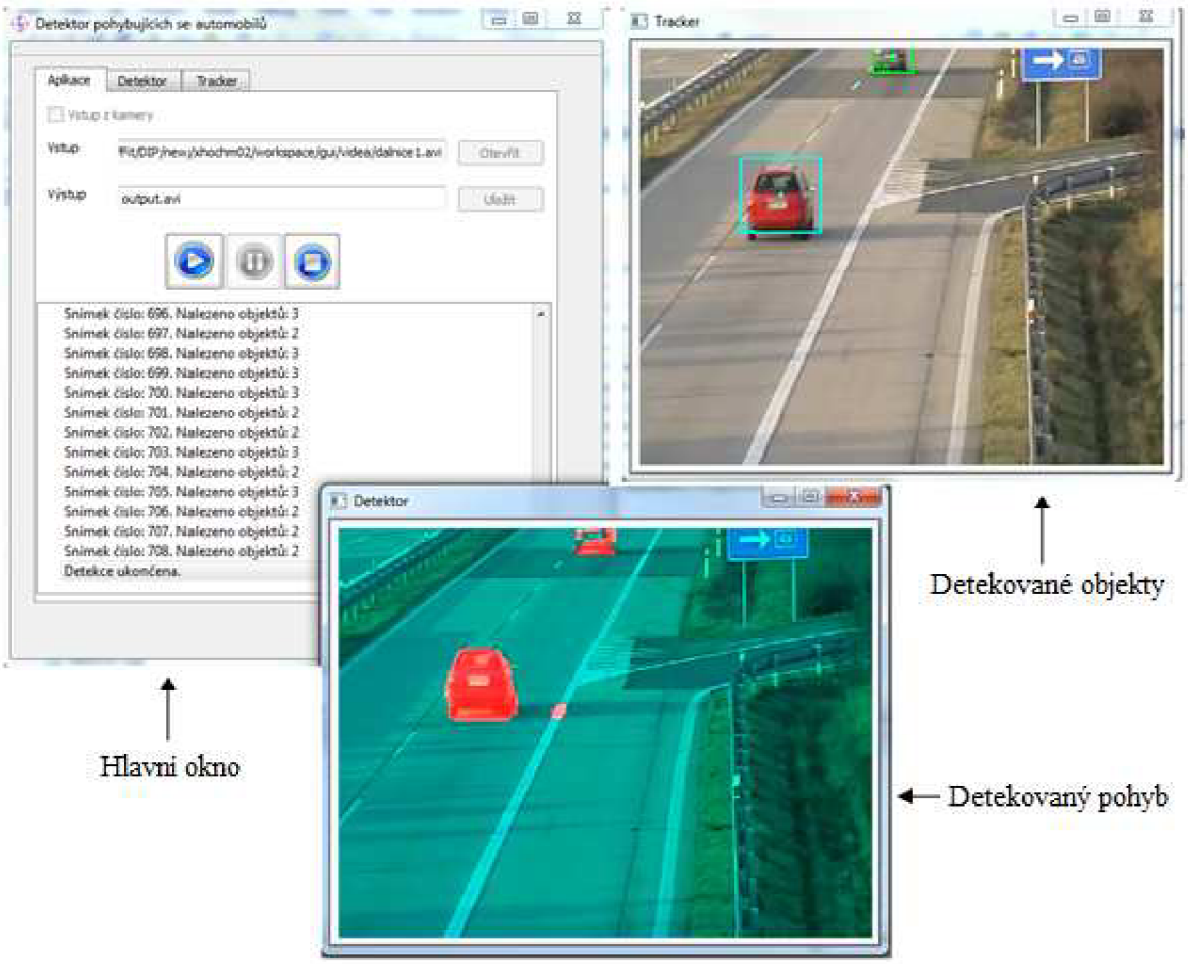
[22] Mikic, I. , Cosman, P., Kogut, G., Trivedi, M.M.: Moving shadow and object detection in traffic scenes, Proceedings of Int'l Conference on Pattern Recognition, 2000.

[23] Stauder, J., Mech, R., Ostermann, J.: Detection of moving cast shadows for object segmentation, IEEE Transactions on Multimedia, 1999. Dostupné na URL: <http://www.tnt.uni-hannover.de/papers/data/220/220_1.pdf>

Seznam příloh

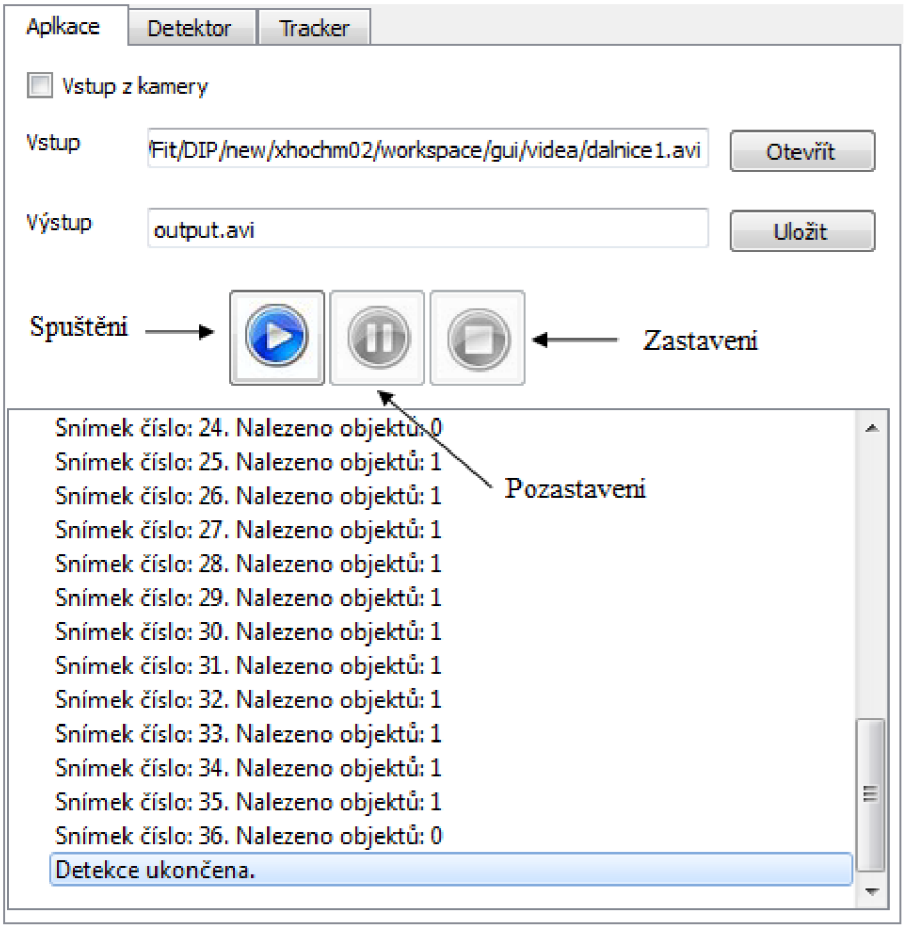
Příloha 1: Ovládání aplikace

Aplikace detektor se skládá ze tří oken. Hlavní okno slouží pro nastavování aplikace. V okně, které jsem pojmenoval Detektor, zobrazuji detekovaný pohyb. V okně Tracker zobrazuji sledované automobily.



Obrázek 63: Vzhled aplikace pro detekci pohybujících se automobilů

Hlavní okno je pomocí záložek rozděleno na tři části. Na záložce aplikace můžete nastavovat vstupní a výstupní video sekvence, spouštět, pozastavovat a ukončovat detekci. Na záložce aplikace jsou také vypisovány informace o běhu aplikace.

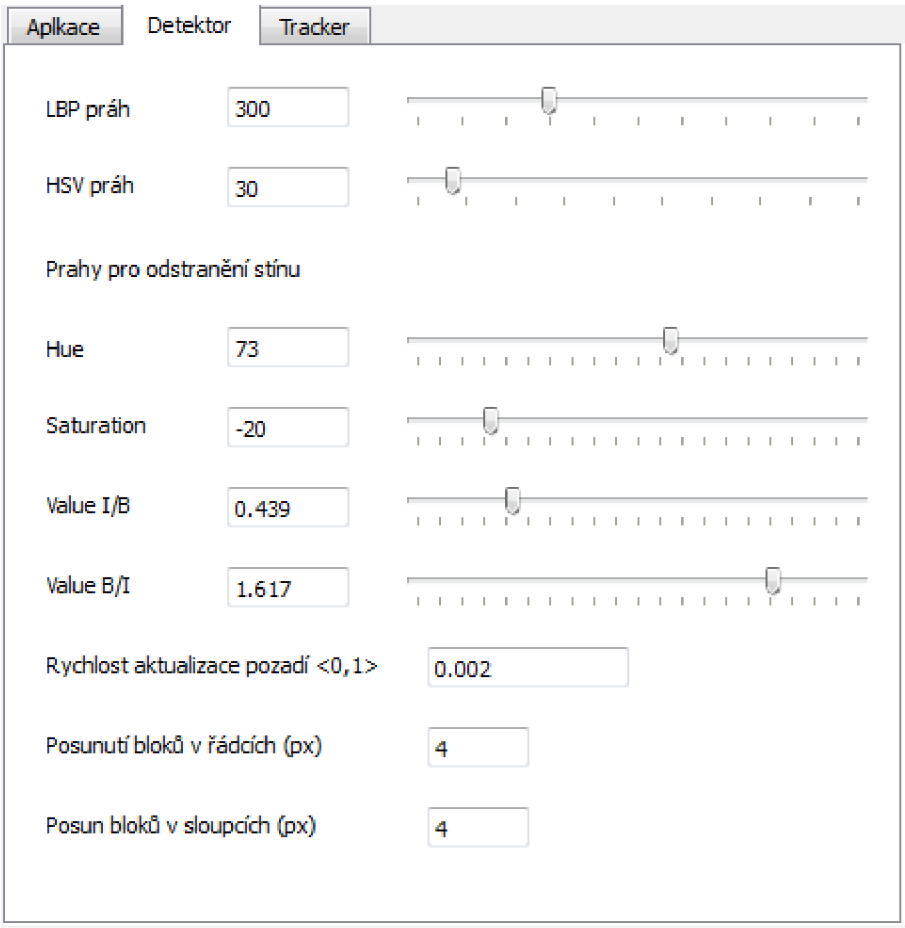


Obrázek 64: Záložka aplikace

Vstupem aplikace může být buď video soubor uložený na disku ve formátu avi, nebo web kamera. Pro zpracování vstupního video souboru, je nutné mít nainstalován kodek, pomocí kterého bylo vstupní video komprimováno. Výstupem je video soubor ve formátu avi. Do výstupu jsou ukládány snímky zobrazované v okně Tracker. Aby mohlo být video uloženo, je nutné mít nainstalován kodek DIVX.

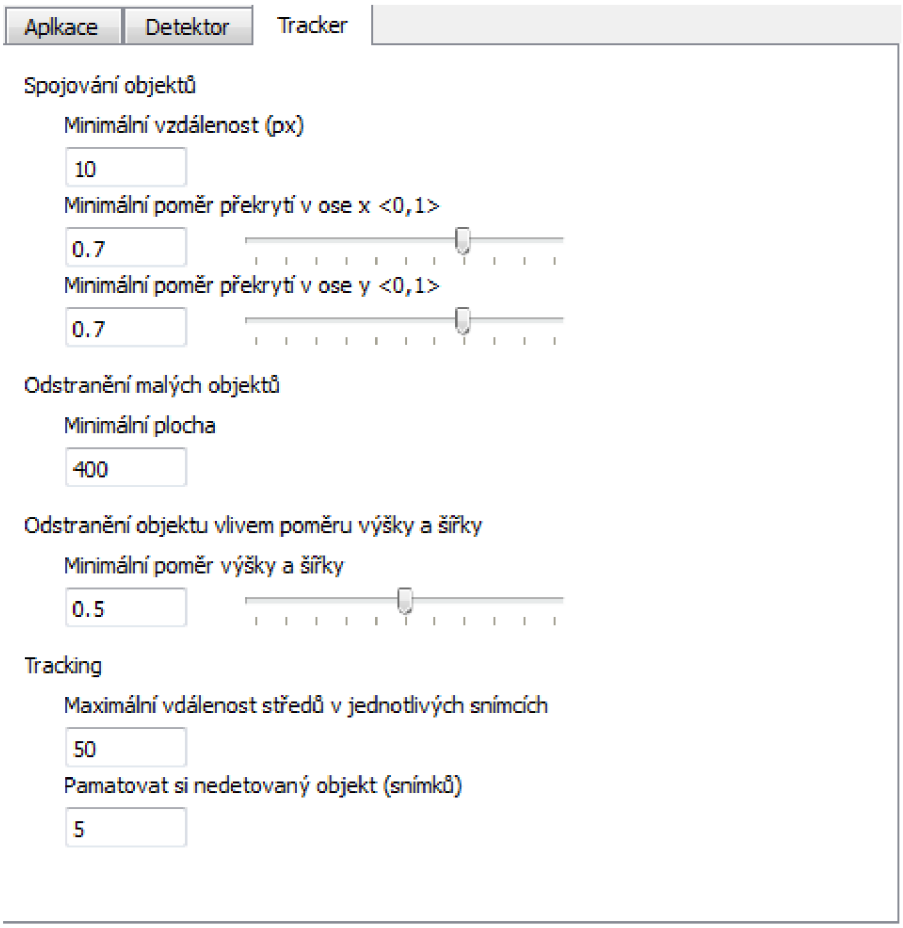
Pomocí tlačítka Spuštění spustíte detekci pohybujících se objektů. Tlačítkem pozastavení přerušíte detekci. Pokud opětovně stisknete Spuštění, detekce bude probíhat od snímku, kde jste detekci pozastavili. Pokud je detekce pozastavena, nelze měnit vstupy ani výstupy aplikace. Při stisknutí tlačítka Zastavení zastavíte detekci. Pokud opětovně stisknete Spuštění, detekce bude probíhat od začátku video sekvence (pokud jste nezměnili vstup). Pokud je detekce zastavena, lze měnit vstupy a výstupy aplikace.

Na záložce detektor lze měnit nastavení detektoru pohybu. Lze měnit prahy pro detekci pohybu pomocí LBP, diference pixelů v HSV a prahy pro odstranění stínů. Dále lze nastavit rychlost aktualizace pozadí a překrytí bloků v řádcích a sloupcích.



Obrázek 65: Záložka detektor

Na záložce tracker lze nastavit parametry pro sledování průchodu objektů scénou. Lze nastavit parametry pro předzpracování objektů (spojování objektů, odstranění malých objektů a objektů se špatným poměrem výšky a šířky) a parametry pro identifikaci objektů mezi jednotlivými snímky.



Obrázek 66: Záložka tracker

Příloha 2: Program pro nalezení optimálních prahů pro odstranění stínů

Pro řešení problému jsem se rozhodl použít evoluční strategii (ES), protože hledám dvě celočíselné hodny a dvě reálné hodnoty. ES oproti genetickému algoritmu, pracuje přímo s reálnými čísly, tudíž není potřeba tato čísla převádět do binárního kódování.

Jedinec je kódován následujícím způsobem:

$$x_i = \left(\tau_H, \tau_S, \beta, \alpha, \sigma_H, \sigma_S, \sigma_\beta, \sigma_\alpha \right)$$

kde $\alpha, \beta, \tau_S, \tau_H$ jsou hodnoty prahů a σ jsou střední odchylky pro jednotlivé prahy. Hodnoty σ se používají při mutaci jedince.

Mutace:

U ES se mutace provádí pomocí normálního rozložení. Existuje několik variant mutace. Já jsem zvolil variantu, při které má každý parametr svou vlastní střední odchylku σ normálního rozložení. Postup mutace je následující:

- $\sigma_i' = \sigma_i e^{\tau' N(0,1) + \tau N_i(0,1)}$
- $x_i' = x_i + \sigma_i' N_i(0,1)$

Nejdříve je upravena hodnota σ. $N(0,1)$ je náhodné číslo vygenerované pomocí normálního rozložení, které se použije pro všechny σ. $N_i(0,1)$ je náhodné číslo vygenerované pomocí normálního rozložení, které je unikátní pro každou hodnotu σ. τ, τ' jsou učící parametry definovány následovně:

- $\tau = \frac{1}{\sqrt{2n}}$
- $\tau' = \frac{1}{\sqrt{2\sqrt{n}}}$

kde n je počet parametrů.

Pokud je hodnota $\sigma_i' < \varepsilon_0$, pak $\sigma_i' = \varepsilon_0$.

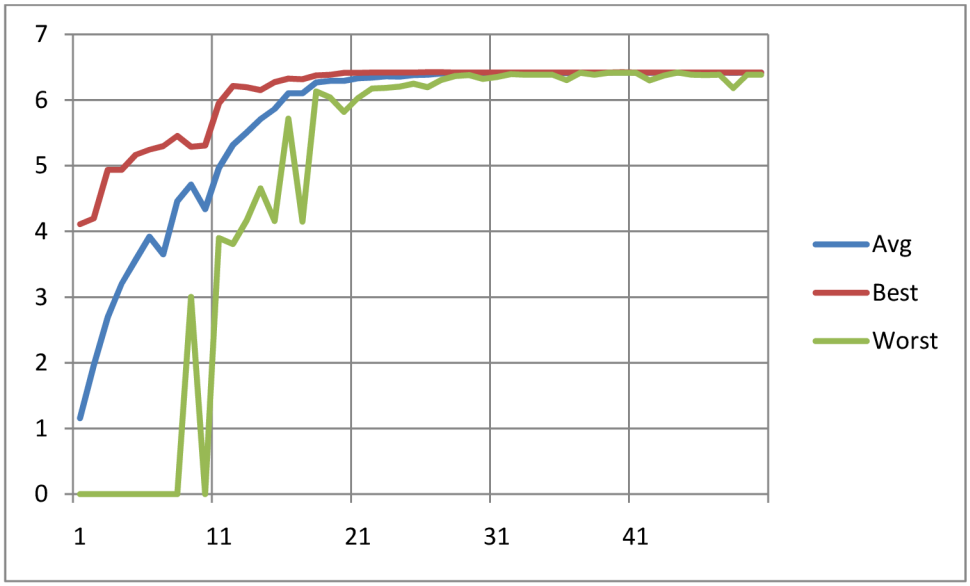
Křížení:

Náhodně jsou vybráni dva jedinci z rodičovské populace. Parametry kříží pomocí diskrétního (uniformního) křížení. Náhodně vygeneruji uniformní masku a na základě masky vyberu příslušné parametry od každého rodiče.

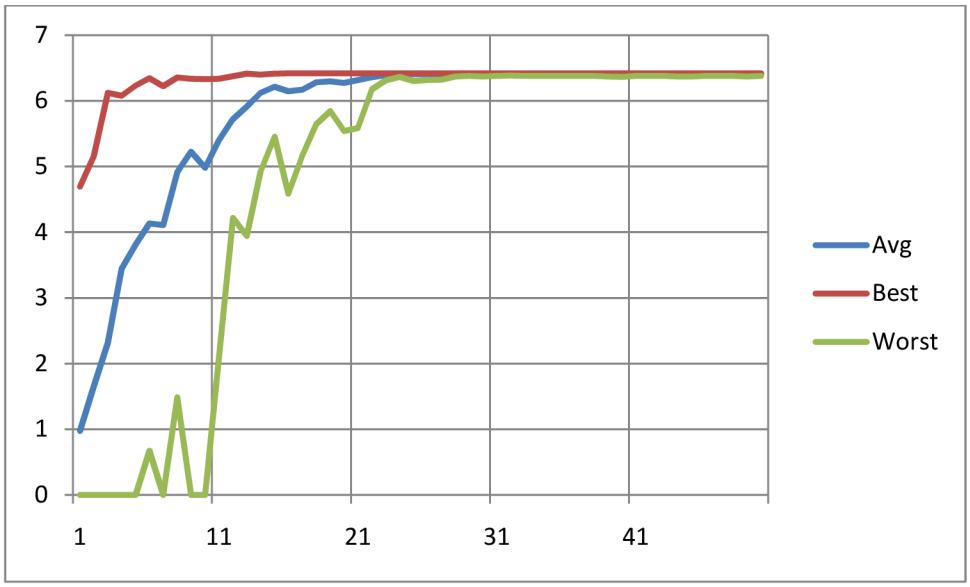
σ kříží průměrem.

Výsledky:

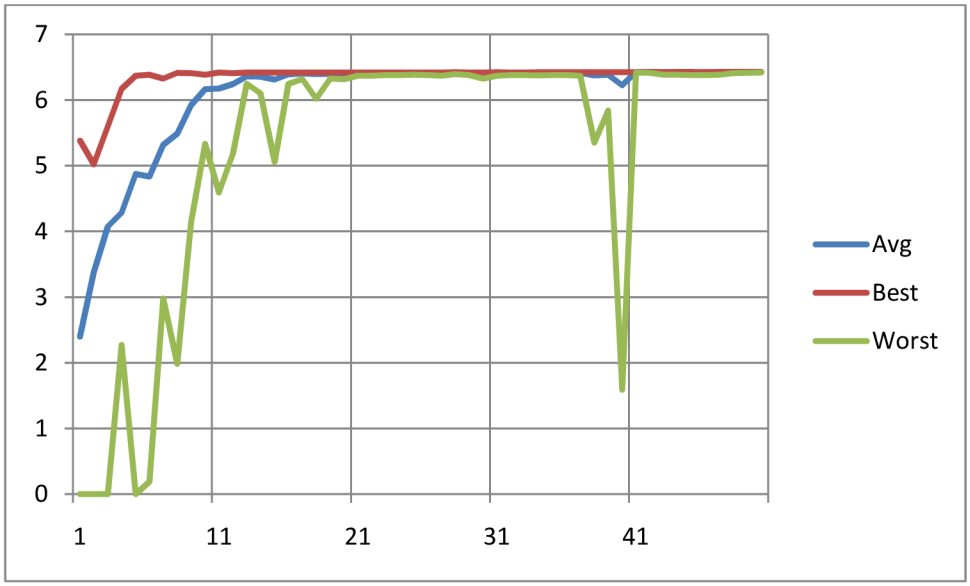
Na Grafech 1 až 3 můžete vidět průběh evoluce. Počet rodičů jsem nastavil na 10, počet potomků na 30 a počet generací na 50. Obnova populace probíhala čárkovou variantou. K vyhodnocení výsledků jsem provedl 10 běhů evoluce.



Graf 1: Průběh evoluce



Graf 2: Průběh evoluce



Graf 3: Průběh evoluce

Nejlepší fitness funkce 6,425 dosáhlo nastavení prahů: $\alpha = 1{,}61767$, $\beta = 0.439782, \tau_S = -20, \tau_H = 73$

Ovládání programu:

Program se ovládá z příkazové řádky. V adresáři, kde se nachází spustitelný soubor, se musí nacházet podadresáře background, compare a sources. Adresář background obsahuje snímky pozadí, adresář compare obsahuje vzory a adresář sources obsahuje zdrojové snímky.

Parametr	Význam
-g <číslo>	Počet generací. Defaultně 100.
-p <číslo>	Velikost populace rodičů. Defaultně 10.
-d <číslo>	Velikost populace potomků. Defaultně 20.
-b <cesta>	Soubor, do kterého se zapisují nejlepší jedinci z jednotlivých generací. Defaultně bestSubject.csv
-r <cesta>	Soubor, do kterého se zapisuje postup evoluce. Defaultně progress.csv
-i <cesta>	Soubor obsahující názvy jednotlivých snímků. Názvy snímků v složkách background, compare a sources se musí shodovat a navíc musí být uvedeny v tomto souboru. Defaultně input.txt
-s	Postup evoluce se nebude vypisovat na standardní výstup
-m	Populace se bude obnovovat pomocí plusové varianty. Defaultně probíhá obnova pomocí čárkové varianty
-h	Nápověda