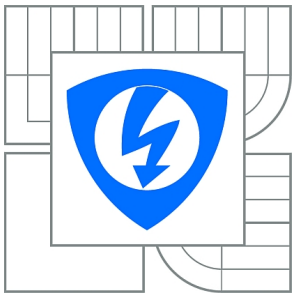

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY


FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION


# SLEDOVÁNÍ POHYBUJÍCÍCH SE OBJEKTŮ

MOVING OBJECT TRACKING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS


AUTOR PRÁCE — MARTIN SEHNOUTKA

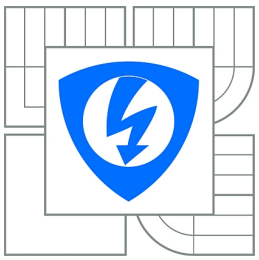
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOSLAV RICHTER, Ph.D.

SUPERVISOR



BRNO 2015



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav automatizace a měřicí techniky**

# Bakalářská práce

bakalářský studijní obor
**Automatizační a měřicí technika**

| | | | |
|---|---|---|---|
| ***Student:*** | Martin Sehnoutka | ***ID:*** | 154865 |
| ***Ročník:*** | 3 | ***Akademický rok:*** | 2014/2015 |

**NÁZEV TÉMATU:**

## Sledování pohybujících se objektů

**POKYNY PRO VYPRACOVÁNÍ:**

Vytvořte prostředí pro sledování pohybu objektů pohybujících se ve snímané ploše.
1) Nastudujte problematiku sledování pohybujících se objektů pomocí kamer.
2) Vytvořte databázi vzorových snímků/videí, na kterých se pohybuje větší množství objektů a posun objektů mezi snímky výrazně nepřesahuje velikost objektu.
3) Navrhněte a naprogramujte algoritmy pro sledování objektů při pohybu.
4) Na základě detekce a sledování objektů vytvořte mapu směru a rychlosti pohybu v daném místě scény.
5) Stanovte možnost využití mapy pohybu pro predikci směru a rychlosti pohybu objektů.

**DOPORUČENÁ LITERATURA:**

Hlaváč V., Šonka M.: Počítačové vidění,Grada, Praha 1992, ISBN 80-85424-67-3
Faugeras O.: Three-Dimensional Computer Vision, The MIT Press 1993

| | | | |
|---|---|---|---|
| ***Termín zadání:*** | 9.2.2015 | ***Termín odevzdání:*** | 25.5.2015 |

***Vedoucí práce:*** Ing. Miloslav Richter, Ph.D.
***Konzultanti bakalářské práce:***

**doc. Ing. Václav Jirsík, CSc.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

## ABSTRAKT

Tato práce se zabývá využitím počítačového vidění pro účely sledování pohybu objektů ve videozáznamu. V první části jsou algoritmy počítačového vidění popsány teoreticky. Další část práce obsahuje návrh mapy pohybu ve scéně. Všechny tyto postupy byly využity při implementaci programu pro trasování objektů. Rozbor funkčnosti programu je uveden v závěru.

## KLÍČOVÁ SLOVA

Sledování pohybu, Kalmanův filtr, Mapa pohybu, OpenCV

## ABSTRACT

This thesis deals with application of computer vision for purposes of object tracking in a video record. In first part of this document, algorithms of computer vision are described theoretically. Next part contains design of map of movement in a scene. All these procedures were used during implementation of program for tracking of moving objects. Discussion of program functionality is in the last chapter.

## KEYWORDS

Motion tracking, Kalman filter, Map of movement, OpenCV

SEHNOUTKA, Martin *Sledování pohybujících se objektů*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2015. 51 s. Vedoucí práce byl Ing. Miloslav Richter, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Sledování pohybujících se objektů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....

podpis autora

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Miloslavu Richterovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.


Brno ..... .....

podpis autora

# OBSAH

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# ÚVOD

Sledování pohybu objektů se řadí do oboru počítačového vidění, jehož cílem je automatizovat získávání informací z obrazového signálu. Samotné počítačové vidění se skládá z poznatků mnoha jiných oborů jako je fyzika, matematika nebo informatika. Problém zpracování obrazu je v tom, že neexistují jednoznačné postupy pro jednotlivé úlohy. Jinými slovy, není možné definovat univerzální řešení. Přesto je uplatnění tohoto oboru veliké a zahrnuje oblasti jako jsou automatizace, lékařství nebo dopravní systémy.

Cílem této bakalářské práce je shrnout základní poznatky z oboru počítačového vidění, které je následně možné aplikovat na úlohu sledování objektů při pohybu a zaznamenávání jejich trasy. Dále je zde nastíněna možnost predikce pohybu a trasování objektů i v případě, že jsou zakryty překážkou. V neposlední řadě je navržen způsob mapování pohybu ve scéně, který je dále využit při predikci pohybu. Všechny tyto algoritmy jsou následně implementovány do programu, který testuje jejich použití na reálných záznamech. Tyto záznamy byly pořízeny s ohledem na různý charakter pohybu i tvaru objektů a dávají možnost otestovat všechny funkce programu.

V první kapitole tohoto dokumentu jsou popsány metody počítačového vidění, které budou dále využívány. V druhé kapitole jsou popsány záznamy, které byly pořízeny pro účely testování. Ve třetí kapitole je navržena mapa směru a rychlosti pohybu a její využití. Ve čtvrté kapitole je popsána realizace programu v jazyku C++. Nakonec je provedeno zhodnocení dosažených výsledků.

# 1 ROZBOR PROBLEMATIKY

## 1.1 Popis úlohy

Úlohou je najít metody, kterými lze sledovat pohyb objektů ve scéně snímané kamerou. Tato úloha je velmi obecná a není možné najít univerzální řešení pro všechny možné situace. Základem je nalezení objektů v jednotlivých snímcích a následné sjednocení objektů mezi snímky.

V nejjednodušším případě by se všechny objekty pohybovaly odděleně od sebe, měly malý posun mezi snímky a ztrácely se až při opuštění zorného pole kamery. Ovšem za běžných situací se objekty navzájem slučují, například při míjení se, nebo naopak ztrácejí, pokud se přesunou za statickou překážku, například v zorném poli je sloup. Všechny takovéto úvahy o možných stavech je nutné zahrnout do metody sledování pomocí počítače.

## 1.2 Postup zpracování obrazu

Celou úlohu zpracování obrazu je možné rozdělit na několik po sobě následujících částí. Prvním krokem je snímání obrazu. V tomto kroku je vstupní optická veličina převedena do číselných dat, která budou dále zpracována v počítači. Dalším krokem je předzpracování obrazu. To může zahrnovat odstranění šumu, zvýraznění hran a podobné operace, které mají usnadnit další zpracování. Třetím krokem je segmentace obrazu, to znamená oddělení objektů zájmu od pozadí. V případě této práce budou objekty zájmu pohybující se objekty. Posledním krokem je porozumění obrazu. [1]

## 1.3 Segmentace obrazu

Segmentace je proces oddělení popředí, objektů zájmu, od pozadí. Pro jednotlivé obrazy je možné použít metody jako je prahování nebo sledování hran. Ve videu ze statické kamery je možné použít odčítání pozadí, také známé jako detekce popředí. Tato metody předpokládá, že je dostupný obraz pozadí. Problém ovšem nastává, pokud se pozadí může měnit, například vlivem počasí, denní doby apod. Výsledkem segmentace je binární obraz. [2]

Nejjednodušší způsob implementace segmentace je odečtení aktuálního snímku $I(x, y, t)$ od snímku pozadí $B(x, y)$. Následně je každý bod obrazu, ve kterém je rozdíl větší než prahová hodnota $Th$, označen jako popředí a ostatní jako pozadí. Matematicky se dá postup zapsat takto:

$$|I(x, y, t) - B(x, y)| > Th \tag{1.1}$$

Obraz pozadí může být získán prostým průměrem z $N$ posledních prvků. Případně je možné použít ještě jiné metody, jako je například medián.

$$B(x,y) = \frac{1}{N}\sum_{i=1}^{N} I(x,y,t-i) \tag{1.2}$$

Na obrázku 1.1 je zobrazen model pozadí, který byl získán postupným průměrováním předchozích snímku ve videu. Jak je vidět, model se adaptuje tak rychle, že jsou v něm vidět pomalu se pohybující objekty, což samozřejmě není žádoucí.

Obr. 1.1: Průměrované pozadí

### 1.3.1 Kumulativní průměr Gaussových rozdělení

Tato metoda umožňuje adaptivní segmentaci obrazu, to znamená, že se postupně přizpůsobuje změnám v obraze. Každý obrazový bod pozadí $B(x,y)$ je charakterizován normálním rozdělením $N(\mu,\sigma^2)$. Střední hodnota se postupně mění podle vztahu:

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

Kde $I_t$ je jasová hodnota nového pixelu a $\alpha$ je váha nového přírůstku. Tuto váhu můžeme volit v závislosti na požadované rychlosti přizpůsobení nebo naopak na požadované stabilitě. Rozptyl se počítá obdobně.

Každý obrazový bod je potom klasifikován jako popředí, pokud vyhovuje následující nerovnici:

$$\frac{|I_t - \mu_t|}{\sigma_t} > Th \tag{1.4}$$

Tato metoda může být dále upravena, například výpočtem průměru pouze z obrazových bodů, které byly vyhodnoceny jako pozadí [3].

### 1.3.2 Směs Gaussových rozdělení

Tato metoda je v anglické literatuře známá jako Mixture of Gaussians (MOG). Na rozdíl od předchozí metody zavádí pro každý obrazový bod více Gaussových rozdělení.

Předpokládáme, že průběhem času se bude pozadí měnit. Například se budou ohýbat stromy ve větru nebo na řece se budou pohybovat vlny. Proto se pro každý objekt v pozadí, popředí a případně i stín zavede vlastní Gaussovo rozdělení. Celkový počet těchto rozdělení je označen jako $K$. Pravděpodobnostní funkce pro jeden obrazový bod může být zapsána jako:

$$P(i_t) = \sum_{i=1}^{K} \omega_{i,t} N(i_t - \mu_{i,t}, \Sigma_{i,t}) \tag{1.5}$$

Kde $\omega_{i,t}$ značí váhu (nebo také výšku) daného rozdělení. Rozdělení s vysokou vahou a nízkým rozptylem budou pravděpodobně patřit k pozadí, jelikož se v obraze objevují často a příliš se nemění.

Aby byl bod označen za pozadí, musí vyhovět následujícímu kritériu. Nejprve jsou všechna rozdělení ohodnocena na základě poměru váhy $\omega_i$ a směrodatné odchylky $\sigma_i$. Potom je prvních B rozdělení, která vyhoví následující podmínce, označeno jako pozadí.

$$\sum_{i=1}^{B} \omega_i > Th \tag{1.6}$$

Každému obrazovému bodu musí být přiřazeno jedno rozdělení a zároveň musí být aktualizovány parametry daného rozdělení. Jednotlivá rozdělení jsou postupně podle výše uvedeného hodnocení testována na podmínku:

$$\frac{|I_t - \mu_t|}{\sigma_t} > Th \tag{1.7}$$

První vyhovující rozdělení je označeno jako příslušné pro daný bod $I_t$. Pokud nevyhovuje žádné rozdělení, je poslední v pořadí nahrazeno novým se střední hodnotou $I_t$, nízkou vahou a vysokým rozptylem. [3]

Výsledný binární obraz získaný touto metodou zobrazuje obrázek 1.2.

## 1.4 Ohraničení objektů v obraze

Pro ohraničení objektů v binárním obraze jsou vyhledávány izolinie, také zvané kontury. Kontury jsou křivky, které spojují místa se stejnou funkční hodnotou. V případě trojrozměrné obrazové funkce $I(x, y)$ spojují místa se stejnou hodnotou jasu. Tedy vytváří řez obrazovou funkcí pomocí roviny rovnoběžné s rovinou x,y. Příkladem běžně používaných kontur jsou vrstevnice na mapě. [4]

Obr. 1.2: Segmentace pozadi

## 1.5 Morfologické transformace

Matematická morfologie geometrizuje úlohu zpracování obrazu. Zaměřuje se na tvar objektů a transformace, které tento tvar zachovávají. Příkladem použití může být odstranění šumu, zjednodušení tvaru, ztenčování, zesilování apod. Vstupní obrazem je v tomto případě binární obraz (obecně může být i šedo-tónový).

Transformace jsou realizovány jako relace mezi binárním obrazem I a strukturním elementem B v dvourozměrném prostoru $E^2$.

### 1.5.1 Dilatace

Dilatace množiny I se strukturním elementem B se vypočte jako vektorový součet I a B. Její aplikace se projeví jako rozšíření objektů v obraze a zaplnění malých děr. Příklad dilatace na masce získané segmentací pozadí je zobrazen na obrázku 1.3

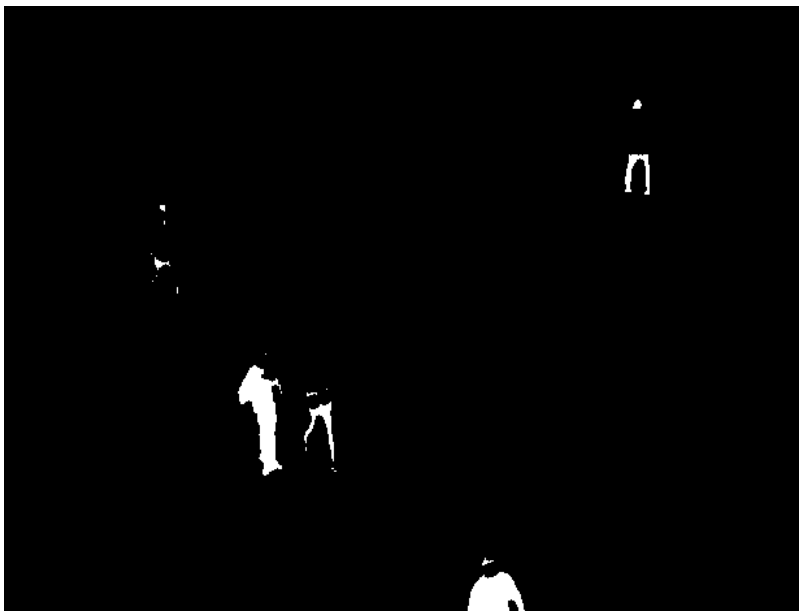
$$I \oplus B = \{d \in E^2 : d = i + b, i \in I, b \in B\} \tag{1.8}$$

### 1.5.2 Eroze

Eroze se vypočítá jako vektorový rozdíl množin I a B. Používá se pro odstranění malých objektů a zjednodušení těch větších. Ukázka je na obrázku 1.4.

$$I \ominus B = \{d \in E^2 : d + b \in I \forall b \in B\} \tag{1.9}$$

Obr. 1.3: Dilatace výše ukázané masky popředí.



Obr. 1.4: Eroze masky popředí.

### 1.5.3 Otevření a uzavření

Kombinací eroze a dilatace je možné dosáhnout dalších morfologických operací. Eroze následovaná dilatací se nazývá otevření, naproti tomu dilatace následovaná erozí se nazývá uzavřením. Otevření se používá pro oddělení objektů spojených úzkou spojnicí. Uzavření se používá pro zaplnění úzkých mezer a malých děr. Opakované použití těchto operací nemá smysl, jelikož výsledek již zůstane nezměněn.[1]

V této práci je možné využít otevření pro oddělení spojených objektů. Naopak by mohlo být výhodné použít i uzavření pro zaplnění objektů s dírami nebo mírně rozpojených objektů.

## 1.6 Detekce významných bodů

Při sledování pohybu objektů v obraze je výhodné najít body, které je možné jednoznačně identifikovat a trasovat. Takovýmito body zájmu můžou být například rohy. Opakem vhodných bodů jsou například jednobarevná plocha, u které není možné jednoznačně určit pohyb v žádné ose, nebo hrana, u které je možné určit pohyb pouze v jedné ose.

Formálně je možné roh definovat jako místo, kde se střetnou dvě hrany s různým směrem nebo obecněji jako bod, který ve svém okolí vykazuje veliký gradient jasu. [5]

### 1.6.1 Harrisův rohový detektor

Harrisův rohový detektor je vylepšením Moravcova rohového detektoru. Moravcův rohový detektor počítá sumu z absolutních hodnot odchylek jasové funkce pro nejbližší pixely:

$$M(x,y) = \frac{1}{8} \sum_{k=x-1}^{x+1} \sum_{j=y-1}^{y+1} |f(k,j) - f(x,y)| \tag{1.10}$$

Harrisův detektor počítá sumu kvadrátů odchylek. Základem je posunutí výřezu obrazu $W \in I$ o $\Delta x, \Delta y$. Poté je spočtena suma kvadrátů odchylek mezi obrazem I a posunutým výřezem W. Po rozvoji analytického zápisu do Taylorovy řady a zanedbání členů vyšších řádů je vzorec pro výpočet:

$$S(x,y) = [\Delta x, \Delta y]\mathbf{A}(x,y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{1.11}$$

Kde matice **A** je:

$$\mathbf{A}(x,y) = \begin{bmatrix} \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial^2 I(x_i,y_i)}{\partial x^2} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial I(x_i,y_i)}{\partial x} \frac{\partial I(x_i,y_i)}{\partial y} \\ \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial I(x_i,y_i)}{\partial y} \frac{\partial I(x_i,y_i)}{\partial x} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial^2 I(x_i,y_i)}{\partial y^2} \end{bmatrix} \tag{1.12}$$

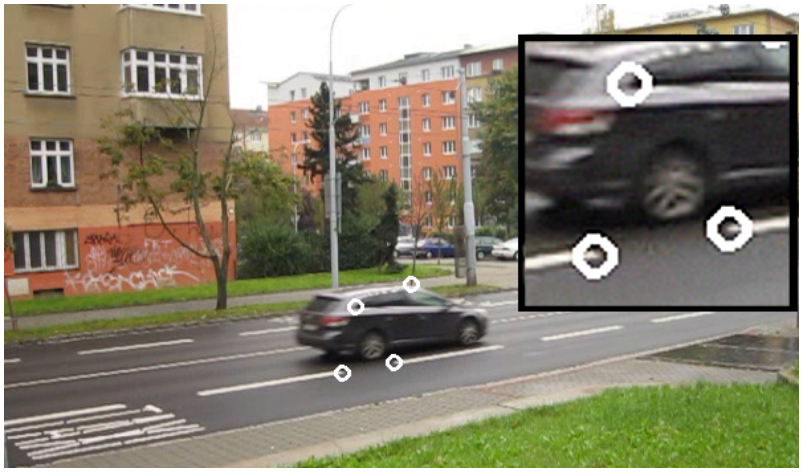
Pro matici **A** jsou následně vypočítána vlastní čísla a můžou nastat tři možné stavy:

- Obě vlastní čísla jsou malá. To znamená, že v daném místě se nenachází ani roh, ani hrana. Jde o plochou část jasové funkce.
- Jedno číslo je malé a druhé je veliké. V daném místě je hrana.
- Vlastní čísla jsou obě veliká. Na místě se nachází hledaný roh.

### 1.6.2 Shi-Tomasi detektor

Tento detektor je dalším vylepšením Harrisova detektoru. Aby byl bod uznán jako roh, musí menší z vlastních čísel matice **A** být větší než prahová hodnota Th. [6]

Ukázka detektoru významných bodů v okolí pohybujícího se auta je zobrazena na obrázku 1.5.



Obr. 1.5: Detekce rohů

## 1.7 Sledování pohybu

Pro analýzu pohybu neexistují univerzální postupy, proto existuje více metod, které ovšem pracují jen za určitých předem daných podmínek. Tato práce se zabývá zpracováním záznamu, který byl zachycen statickou kamerou a posun objektů mezi jednotlivými snímky je malý vůči rozměru samotného objektu. Cílem analýzy je sledovat trajektorii objektů.

### 1.7.1 Optický tok

Cílem výpočtu optického toku je znázornění pohybu objektů na snímku za časový interval *dt*. Ten může například znamenat periodu, se kterou byly pořízeny snímky. Pro zjištění optického toku zavádíme dva předpoklady:

- Osvětlení objektů se v čase nemění.
- Sousední body (body jednoho objektu) se pohybují stejným směrem.

První předpoklad je důležitý, jelikož změnou jasu by mohlo dojít k chybné detekci pohybu. Druhý předpoklad se hodí například pro rozpoznání jednotlivých objektů od sebe.

Pro odvození výpočtu optického toku opět zavedeme jasovou funkci I(x,y,t). Korespondenci bodu, který se za čas *dt* posunul o *dx, dy* lze zapsat rovnicí:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{1.13}$$

Cílem je najít vektor rychlosti:

$$\mathbf{c} = \left( \frac{dx}{dt}, \frac{dx}{dt} \right) = (u, v) \tag{1.14}$$

Provedením výpočtu optického toku můžeme najít v obraze čtyři druhy pohybu:

- Translace v konstantní vzdálenosti
- Translace do dálky nebo přibližování
- Rotace kolem osy pohledu
- Rotace kolmá na osu pohledu

Optický tok jsem zkoušel využít pro detekci pohybu významných bodů. Tyto body jsem následně rozřadil mezi jednotlivé objekty a chtěl jsem je využít jako ukazatel směru a velikosti pohybu objektu. Bohužel u některých objektů se mi nedařilo spolehlivě nacházet významné body, které by dále šly trasovat, proto jsem od této teorie upustil a místo ní použil Kalmanův filtr pro predikci dalšího pohybu. Obrázek 1.6 ukazuje detekci pohybu významných bodů na video-záznamu s pohybujícími se vozidly.

### 1.7.2 Kalmanův filtr

Je filtr, který využívá sérii měření zatížených šumem a z nich vypočítává odhad přesné hodnoty. Využívá tedy znalost předchozího vývoje měřených veličin k určení budoucího vývoje. Základem filtru je model systému, který je popsán vektorem $\mathbf{x}$, a model nepřesných měřených hodnot $\mathbf{z}$.

Formálně tedy máme model systému:

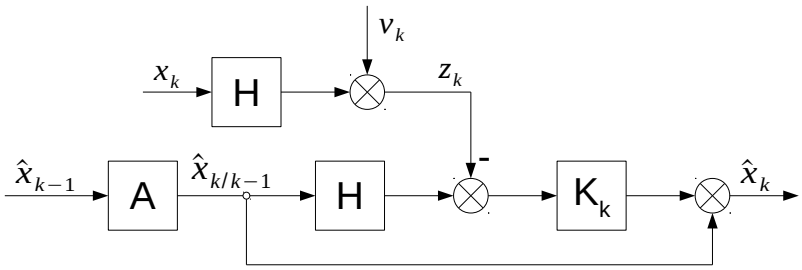
$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k \tag{1.15}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{1.16}$$

Obr. 1.6: Výpočet optického toku pro významné body

Kde **x**, **z** jsou výše zmíněné modely stavu systému a měření, **w** a **v** jsou vektory šumu, u kterých předpokládáme nulovou střední hodnotu a Gaussovo rozdělení. Matice **A** je maticí přenosu stavů a vyjadřuje tedy vztah mezi aktuálním a budoucím stavem. **H** je matice pozorování (měření) a vyjadřuje vztah mezi měřenými veličinami a stavem systému. Dolní index k u jednotlivých členů rovnice vyjadřuje krok výpočtu, což u videozáznamu znamená čas, ve kterém byl daný snímek pořízen.

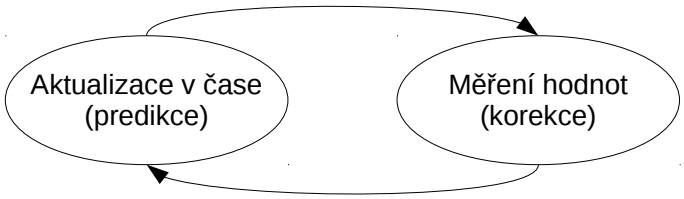
Schématicky lze problém modelovat takto: Na vstupu je korigovaná hodnota stavu z minulé iterace, z té je přes matici přenosu stavů vypočten odhad následujícího stavu. Tento odhad je porovnán s reálným měřením a přes Kalmanovo zesílení je vypočtena korigovaná hodnota odhadu.




Obr. 1.7: Schématické znázornění výpočtu predikce a korekce

Jak již bylo naznačeno, výpočet se skládá ze dvou fází. První je časová aktualizace, což je predikce dalšího stavu, druhá fáze je aktualizace měření, jinak řečeno korekce. [5]

Ke každé fázi náleží sada rovnic, ze kterých je vypočten stav systému. Jednotlivé rovnice jsou uvedeny v tabulce 1.1.




Obr. 1.8: Cyklus predikce a korekce měření pomocí Kalmanova filtru

Tab. 1.1: Výpočet Kalmanova filtru

| Časová aktualizace | Aktualizace měření |
|---|---|
| $\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1}$ | $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1}$ |
| $\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$ | $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right)$ |
| | $\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_k^-$ |

Dále je tedy zavedena matice $\mathbf{P}$, což je chybová kovariance způsobená šumem $\mathbf{w}$ a chybou odhadu, matice $\mathbf{Q}$, která je určena pouze kovariancí šumu $\mathbf{w}$ a nakonec matice $\mathbf{R}$, která obdobně popisuje šum $\mathbf{v}$.

Na začátku výpočtu jsou určeny výchozí hodnoty matic $\mathbf{P}$, $\mathbf{Q}$ a $\mathbf{R}$, dále je inicializován stav $\mathbf{x}$ například první měřenou hodnotou a poté je prováděno měření a korekce odhadů. Pokud není měření k dispozici, je možné věřit pouze odhadu a dále počítat jen s ním. Této možnosti je využito například, když se jednotlivé objekty překrývají a není je tedy možné jednoznačně rozlišit.

### 1.7.3 Návrh modelů pro Kalmanův filtr

Vytvořený model by měl popisovat objekt pohybující se konstantní rychlostí ve 2D prostoru. První byly navrženy stavy systému. Tyto stavy jsou inicializovány při vzniku objektu změřenou polohou a nulovou rychlostí.

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x = \dot{x} \\ v_y = \dot{y} \end{pmatrix} \tag{1.17}$$

Model se tedy skládá z dvourozměrné polohy a rychlosti ve směru x a y. Dále jsou zapsány rovnice popisující vztah mezi stavy, ze kterých jsou vyjádřeny matice přenosu

stavů. Proměnná T značí vztah mezi polohou a rychlostí. Má tedy rozměr času.

$$\begin{aligned} x_{k+1} &= x_k + T v_{x,k} \\ y_{k+1} &= y_k + T v_{y,k} \\ v_{x,k+1} &= v_{x,k} \\ v_{y,k+1} &= v_{y,k} \end{aligned}$$

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.18}$$

Dalším krokem je popis dynamického šumu soustavy **w** a z něj vznikající matice kovariance. Předpokladem je, že šum se může skládat z náhodného zrychlení a zpomalení:

$$\mathbf{w} = \begin{pmatrix} 0 \\ 0 \\ N(0, a_x^2) \\ N(0, a_y^2) \end{pmatrix} \tag{1.19}$$

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a_x^2 & a_x a_y \\ 0 & 0 & a_x a_y & a_y^2 \end{pmatrix} \tag{1.20}$$

Tímto jsou popsány stavy soustavy, ještě je potřeba popsat měření. Jelikož rychlost není možné přímo měřit, tak se bude matice měření skládat pouze z údajů o poloze.

$$\mathbf{z} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.21}$$

Měřené veličiny je nyní potřeba vztáhnout ke stavovým veličinám, aby odpovídaly rozměry jednotlivých matic.

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{1.22}$$

Měření je také zatíženo šumem, který je popsán normálním rozdělením a maticí kovariance.

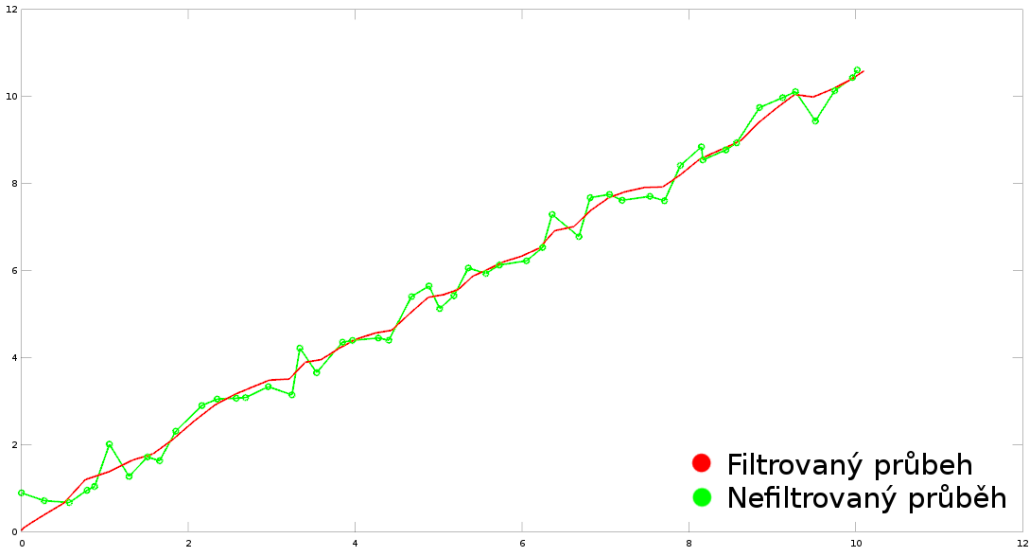
$$\mathbf{w} = \begin{pmatrix} N(0, n_x^2) \\ N(0, n_y^2) \end{pmatrix} \tag{1.23}$$

$$\mathbf{Q} = \begin{pmatrix} n_x^2 & n_x n_y \\ n_x n_y & n_y^2 \end{pmatrix} \tag{1.24}$$

Poslední matice je kovariance stavů. Ta v podstatě vyjadřuje jak moc věříme stavům oproti měření. Tato matice je inicializována velikými čísly, které znamenají, že větší váhu mají ze začátku měřené hodnoty.

$$\mathbf{P} = \begin{pmatrix} \sigma_x^2 & \sigma_{x,y} & \sigma_{x,vx} & \sigma_{x,vy} \\ \sigma_{y,x} & \sigma_y^2 & \sigma_{y,vx} & \sigma_{y,vy} \\ \sigma_{vx,x} & \sigma_{vx,y} & \sigma_{vx}^2 & \sigma_{vx,vy} \\ \sigma_{vy,x} & \sigma_{vy,y} & \sigma_{vy,vx} & \sigma_{vy}^2 \end{pmatrix} \tag{1.25}$$

Celý model jsem nejprve simuloval v jazyku pro numerické výpočty Octave. Výsledek použití je vidět na obrázku 1.9.




Obr. 1.9: Simulace Kalmanova filtru v Octave

Výše navržený model popisuje objekt pohybující se konstantní rychlosti. To je však na některých záznamech nedostatečné, a proto je potřeba model rozšířit o údaje zrychlení.

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x = \dot{x} \\ v_y = \dot{y} \\ a_x = \dot{v_x} = \ddot{x} \\ a_y = \dot{v_y} = \ddot{y} \end{pmatrix} \tag{1.26}$$

Další odvození modelu je obdobné a proto již nebude více rozebíráno. Matice se pouze rozšíří o další hodnoty pro zrychlení.

# 2 DATABÁZE VZOROVÝCH SNÍMKŮ

Pro účely testování této práce bylo vytvořeno několik záznamů, na nichž se pohybují různé druhy objektů. Záznamy byly vytvořeny s ohledem na vhodné světelné podmínky. Snahou bylo minimalizovat prvky záznamu, které by mohly narušovat segmentaci. To jsou například dlouhé stíny v odpoledních hodinách nebo déšť a sníh.

Záznamy byly pořízeny fotoaparátem Canon Powershot D10 a kamerou Panasonic.

Tab. 2.1: Parametry záznamu

| Parametr | Canon Powershot D10 | Panasonic |
|---|---|---|
| Rozlišení | 640x480 | 704x576 |
| Snímků za sekundu | 30 | 30 |

## 2.1 Obraz pozadí

Vzhledem k použité metodě segmentace je výhodné mít k dispozici obraz pozadí. Pokud tento obraz není k dispozici, jako pozadí je využit první snímek na videu. To však způsobuje vznik falešných objektů na místech, kde v prvním snímku byl objekt. V rámci této práce je využito několik postupů pro potlačení tohoto jevu. Prvním opatřením je rychlá adaptace modelu pro algoritmus MOG. Prvních několik sekund po spuštění videa je prováděno pouze vytváření modelu pozadí a nejsou spouštěny algoritmy pro detekci pohybu. Dalším opatřením je vytváření obrazu pomocí průměrování všech snímků ve videu. V rámci implementace byl vytvořen krátký program, který načte video a zprůměruje všechny jeho snímky. Tato metoda se ukázala jako vhodná pouze pro videa, kde není příliš mnoho objektů. Poslední možností je vytváření obrazu pozadí "ručně". V jednodušším případě je možné ve videu najít snímek, který právě v danou chvíli neobsahuje žádný pohybující se objekt. V opačném případě je nutné složit dohromady několik segmentů obrazu z různých snímků videa a vytvořit obraz pozadí za pomoci grafického editoru (např. GIMP).

## 2.2 Pohyb osob

První video je pořízeno z budovy Fakulty elektrotechniky a komunikačních technologií T12. Zabírá nádvoří před budovou, po kterém chodí lidé do školy. Záznam

má velmi malý kontrast, takže není snadná jeho segmentace. U jedné osoby, která se pohybuje na kraji nádvoří u trávníku, je segmentace použitými algoritmy téměř nemožná.

Obrázek 2.1 ukazuje uměle vytvořený snímek pozadí a jeden snímek ze záznamu.

## 2.3 Pohyb automobilů

Z ulice Úvoz jsou pořízeny dva různé záznamy. Jeden s menším a druhý s větším provozem. Na obou záznamech působí rušivě sloupy, které drží troleje. V prvním záznamu to není takový problém, jelikož se za sloup schová téměř celé auto a díky predikci pohybu je na druhé straně opět nalezeno. U druhého záznamu je auto sloupem přepůleno, takže se po segmentaci jeví jako dva objekty. U obou záznamů byl obrázek pozadí pořízen přímo z videa, bez použití průměrování. Obzvláště u hustého provozu průměrovaní všech snímků nemělo dobré výsledky.

Obrázky 5.2 a 2.3 ukazují snímky z videí. Na druhém obrázku je vidět trolejbus, který po segmentaci zabírá téměř celý snímek, a tak působí velmi rušivě.

## 2.4 Objekty různého druhu

Na křižovatce ulic Kounicova a Hrnčířská byly pořízeny 3 záznamy s různým přiblížením. Na záběru celé křižovatky 2.4 jde vidět pohyb jak lidí, tak automobilů. Další záznam se zaměřuje více na chodník 2.6 a přechody. Poslední pak zabírá především křižovatku s automobily 2.5. Tímto záznamem projíždí tramvaj, jejíž pohyb se sleduje velmi těžko, protože zabírá celou plochu záznamu a nikdy není vidět celá.

## 2.5 Srovnání jednotlivých záznamu

Obrázky 2.1 až 2.6 zobrazují snímek pozadí (nalevo) a snímek s pohybujícími se objekty (napravo).

Obr. 2.1: Záznam nádvoří před budovou T12

Obr. 2.2: Záznam ulice Úvoz s menším provozem

Obr. 2.3: Záznam ulice Úvoz s hustším provozem

Obr. 2.4: Křižovatka ulic Kounicova a Hrnčířská

Obr. 2.5: Křižovatka ulic Kounicova a Hrnčířská z blízka

Obr. 2.6: Záběr na křižovatku a chodník

# 3 MAPA SMĚRU A RYCHLOSTI POHYBU

Jak bylo naznačeno v semestrální práci, cílem je vytvořit mapu znázorňující pohyb v dané scéně. V ideálním případě by mapa obsahovala spojité křivky pohybu, nicméně obraz videa je vzorkovaný v časové i prostorové rovině. To znamená, že i mapa bude obsahovat pouze konečné množství bodů, ve kterých budou zaznamenávány vektory směru.

## 3.1 Návrh

Základem pro návrh i následnou implementaci je matice. Maximální smysluplný rozměr, který může nabývat, je rozměr videa. Například ve vzorových videích je to 640 sloupců a 480 řádků. Avšak takto vysoké rozlišení mapy nemá příliš význam, protože samotné objekty mají rozměr řádově vyšší, než jeden pixel, takže by se dalo říci, že určité segmenty mapy budou mít vždy stejný směrový vektor, protože bude náležet stejným objektům. Jako příklad může posloužit záběr na silnici, kde se auta vždy pohybují ve stejném pruhu a vždy zabírají několik stovek pixelů. Z této úvahy vychází i návrh mapy. Ta je rozdělena na čtvercové segmenty o velikosti několika pixelů. Tuto velikost je možné nastavit v programu. Jednotlivé směrové vektory jsou pak přepočítány doprostřed tohoto segmentu.

Samotné vektory mají fyzikální rozměr rychlosti, neboť zobrazují změnu polohy za jednu vzorkovací periodu T.

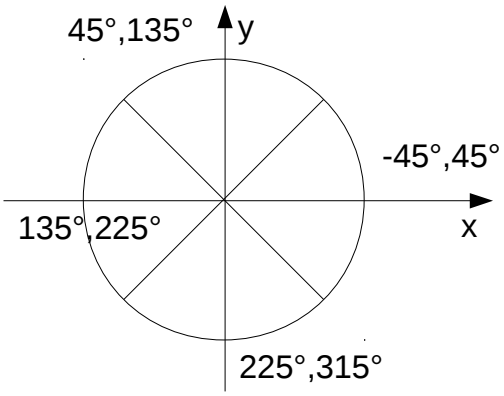
$$\overrightarrow{v} = (v_x, v_y) = \left(\frac{\Delta x}{T}, \frac{\Delta y}{T}\right) \tag{3.1}$$

Obdobně jako u Kalmanova filtru by i mapa mohla zobrazovat vektory zrychlení, případně i další veličiny, ale tím už se tato práce nezabývá.

### 3.1.1 Vrstvy

Jako další vylepšení mapy pohybu bylo navrženo rozdělení mapy na 4 tzv. vrstvy. Každá vrstva zobrazuje směr pohybu objektu, který se na předchozích snímcích pohyboval určitým směrem. To znamená, že pokud se objekt přibližuje zprava, tak se směrové vektory promítnou do vrstvy, která znázorňuje pohyb objektů zprava doleva. Toto rozdělení bylo zavedeno, protože některé snímky automobilů na silnici jsou zaznamenány zboku a tudíž se překrývají auta jedoucí opačnými směry. Kdyby mapa zůstala pouze jednovrstvá, tak by se výsledné směrové vektory vynulovaly a nešlo by ji dále využít.

Počet vrstev a jejich pootočení vzhledem k vodorovné ose je samozřejmě libovolné, nicméně pro tuto práci byly zvoleny 4 vrstvy s tím, že každá z nich je určena pro jeden ze základních směrů doprava, doleva, nahoru a dolů s intervalem hodnot ±45°. Toto uspořádání je zobrazeno na obrázku 3.1 a svým uspořádáním připomíná dobře známý součtový člen.




Obr. 3.1: Rozdělení vrstev mapy podle úhlu

## 3.2 Zdroje dat

Další záležitostí potřebnou pro návrh mapy bylo zvolení zdroje dat, kterými bude mapa plněna. Jako první možnost se nabízí přímo filtrovaná trasa objektu. Ovšem kdyby byla zvolena tato možnost, nebylo by možné tvořit mapu bez použití Kalmanova filtru pro trasování objektů. Jak bude uvedeno v další kapitole, mapa je použita jako základ pro trasování objektů právě na místo Kalmanova filtru. Proto byla vybrána možnost plnění mapy pomocí dat z výpočtu optického toku, který byl blíže rozebrán v úvodní teoretické kapitole. Optický tok navíc umožňuje plnění mapy pomocí většího množství dat (několik významných bodů pro objekt) a experimentálně bylo vyzkoušeno, že není tolik náchylný na šum.

### 3.2.1 Ukládání dat

Předpokladem je, že výsledný vektor rychlosti bude pro daný segment vypočítán jako aritmetický průměr jednotlivých změřených vektorů, kde $n$ značí počet změřených vektorů.

$$\overrightarrow{v} = \left( \frac{v_{x1} + v_{x2} + ... + v_{xn}}{n}, \frac{v_{y1} + v_{y2} + ... + v_{yn}}{n} \right) \tag{3.2}$$

Jelikož jsou měření průběžná, je nutné uchovávat si v paměti počet zápisů, které již byly provedeny, a pomocí něj poté přepočítat průměr. $k$ značí pořadí snímku ve videu, $n$ opět počet měření, $\overrightarrow{v[k]}$ průměrný vektor v kroku $k$ a $\overrightarrow{v_m}$ změřený, nový vektor rychlosti.

$$\overrightarrow{v[k]} = \frac{n-1}{n}\overrightarrow{v[k-1]} + \frac{1}{n}\overrightarrow{v_m} \tag{3.3}$$

## 3.3 Využití pro predikci pohybu

Jedním ze základních prvků výsledného programu je tzv. algoritmus trasování popsaný v kapitole 4.2.3. Tento algoritmus slouží pro predikci a korekci polohy objektu. První implementovanou variantou byl Kalmanův filtr, který polohu předpovídá na základě předchozího vývoje stavů objektu (poloha, rychlost, zrychlení). Mapa je využita v obdobném smyslu. Jestliže se objekt nachází na určité pozici, tak je jeho další poloha předpovězena na základě údaje z mapy.

Výpočet předpokládané polohy je tedy možný za pomoci rovnice:

$$\overrightarrow{\hat{x}[k]} = \overrightarrow{x[k-1]} + T \cdot \overrightarrow{v_{ij}} \tag{3.4}$$

kde $\overrightarrow{\hat{x}[k]}$ značí odhad pro další krok, $\overrightarrow{x[k-1]}$ měření z minulého snímku a $\overrightarrow{v_{ij}}$ údaj z mapy na souřadnicích i,j.

## 3.4 Zjednodušení a implementace

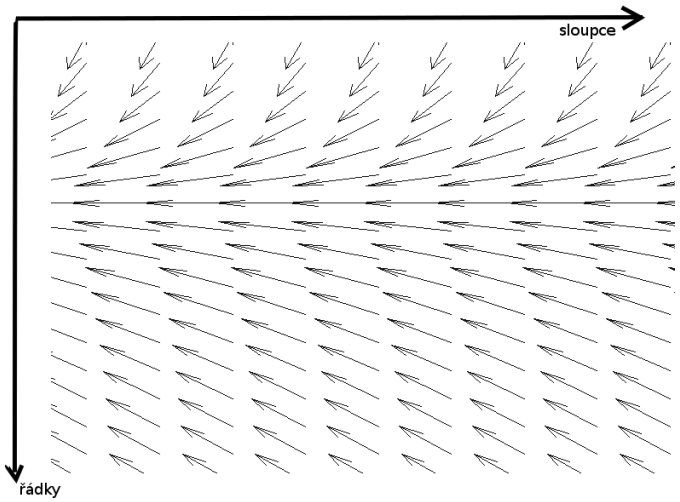
Jak je vidět z rovnice 3.4 a 3.1, při výpočtu směrového vektoru je změna polohy vydělena periodou vzorkování T a naopak při predikci, tj. využití mapy, je touto hodnotou rychlost opět vynásobena. Z toho vyplývá, že je možné při realizaci mapy tyto operace úplně vynechat a do mapy ukládat pouze změnu polohy $\Delta\overrightarrow{p}$ . Ve výsledku je tedy jedna vrstva mapy uložena jako matice $\mathbf{V}$:

$$\mathbf{V}_i = \begin{pmatrix} \Delta\overrightarrow{p}_{11} & \Delta\overrightarrow{p}_{12} & \cdots & \Delta\overrightarrow{p}_{1j} \\ \Delta\overrightarrow{p}_{21} & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \Delta\overrightarrow{p}_{i1} & \Delta\overrightarrow{p}_{i2} & \cdots & \Delta\overrightarrow{p}_{ij} \end{pmatrix} \tag{3.5}$$

Celá mapa je uložena jako vektor:

$$\mathbf{M} = (\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_4) \tag{3.6}$$

Výsledná mapa pohybu na silnici se zúžením by mohla například vypadat jako na obrázku 3.2, který byl pouze vygenerován v programu Matlab. Konkrétní realizace mapy a její vizualizace v programu je popsána v kapitole 4.2.3.




Obr. 3.2: Návrh mapy pohybu

# 4 REALIZACE ALGORITMŮ

## 4.1 OpenCV

OpenCV je knihovna zaměřená na zpracování obrazu v reálném čase. Byla vydána pod licencí BSD, která umožňuje volné použití jak pro komerční, tak pro akademické účely. Zdrojový kód je napsaný v jazyku C++, ale použití je možné i v jazycích jako Python a Java.[7]

V této práci je použita verze OpenCV 2.x, nicméně již je dostupná beta verze OpenCV 3.

### 4.1.1 Struktura knihovny

Celé OpenCV je rozděleno do několika modulů. Jednotlivé moduly zahrnují funkce pro určitou oblast zpracování statického obrazu, videa, případně moduly pro algoritmy umělé inteligence nebo hardwarovou akceleraci. Aby se předešlo konfliktům v pojmenování jednotlivých tříd a funkcí, jsou všechny komponenty knihovny dostupné ve jmenném prostoru *cv*.

Základním prvkem je modul *core*, který obsahuje definice tříd používaných pro ukládání snímků, bodů v obraze, geometrických tvarů a dalších základních typů. Snímky jsou ukládány jako typ matice, jejichž zpracování je možné v různých bitových hloubkách, s různým počtem kanálů a v celočíselném nebo desetinném zápisu.

Dalším modulem je *imgproc*, který obsahuje funkce pro předzpracování obrazu pomocí různých filtrů, převodů mezi barevnými formáty, funkce pro morfologické transformace a detekci významných bodů použitých pro trasování objektů.

Funkce pro zpracování videa a trasování objektů obsahuje modul *video*.

O zobrazení grafického uživatelského rozhraní (GUI) se stará modul *highgui*.

### 4.1.2 Správa paměti

Jelikož zpracování videa může být velmi náročné na paměť, stará se knihovna také o správu paměti. Všechny vytvořené matice, instance třídy *cv::Mat*, si uchovávají údaj o tom, kolik proměnných se na ně odkazuje. Při použití běžného přiřazení se matice nekopíruje, pouze se nastaví reference na již existující prvek. Destruktor, který dealokuje paměť je volán až když na instanci neukazuje ani jedna proměnná. Zkopírování matice je tedy nutné provádět pomocí volání funkcí, které jsou k tomu určené.

### 4.1.3 Ošetření výjimek

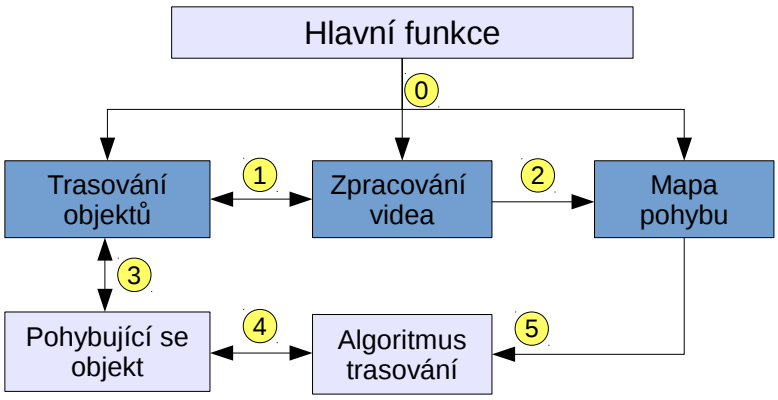
OpenCV zavádí vlastní typ výjimek *cv::Exception*, které jsou volány při kritických chybách. Například když žádaný algoritmus nekonverguje nebo matice mají nevhodný rozměr pro násobení.

### 4.1.4 Alternativní řešení

Dalším možným prostředkem pro realizaci programů počítačového vidění je prostředí Matlab. V tomto prostředí je možné psát programy ve stejnojmenném jazyku. Matlab obsahuje veliké množství nástrojů pro práci s maticemi, signály a obrazem. Nevýhodou jsou vysoké pořizovací náklady za licenci.

## 4.2 Návrh programu

Vzhledem k tomu, že je program implementován v jazyku C++, bylo využito objektově orientovaného přístupu. Architekturu programu zobrazuje blokové schéma 4.1. Hlavní funkce programu je naznačena stejnojmenným blokem. V tomto bloku jsou vytvořeny instance jednotlivých tříd, které jsou znázorněny dalšími bloky ve schématu. V poslední, třetí vrstvě jsou třídy, ke kterým nepřistupuje přímo hlavní funkce, ale jsou volány z objektů jiných tříd.




Obr. 4.1: Celkový návrh programu

### 4.2.1 Hlavní funkce - main

V této funkci je získán vstup od uživatele, následně je vytvořena struktura objektů jako na obrázku 4.1 a nakonec je spuštěno přehrávání.

Prvním krokem je vypsání úvodního textu a výzva pro uživatele k zadání názvu souboru s videem. Od názvu videa jsou následně odvozena jména ostatních souborů, tak jak je to popsáno v další sekci. Pro vyhodnocení názvu a oddělení od přípony je použit následující regulární výraz:

```
regex pieces_regex("(.+)\\.(\\w+)$");
```

Jinak řečeno: Soubor a cesta k němu se musí skládat minimálně z jednoho libovolného znaku, po kterém následuje tečka a minimálně jeden znak dlouhá přípona. Přípona musí být složena z alfanumerických znaků, ostatní jsou nepřípustné.

Následuje otevření všech potřebných souborů, samozřejmě s ošetřením případných problémů. Po vytvoření všech objektů přechází program do smyčky, kde postupně prochází všechny snímky ve videu. Během přehrávání videa je možné program ovládat třemi klávesami, jak je popsáno v tabulce 4.1.

Tab. 4.1: Ovládání programu

| **Klávesa** | **Funkce** |
|---|---|
| q | quit - zastavit a ukončit |
| p | pause - pozastavit |
| c | continue - pokračovat |

### 4.2.2 Soubory potřebné pro přehrávání

O uchování názvů všech souborů se stará třída *TrackerFiles*. Tabulka 4.2 zobrazuje přehled souborů používaných při přehrávání videa. Pokud není k dispozici mapa, je vytvořena nová. Ostatní soubory jsou povinné.

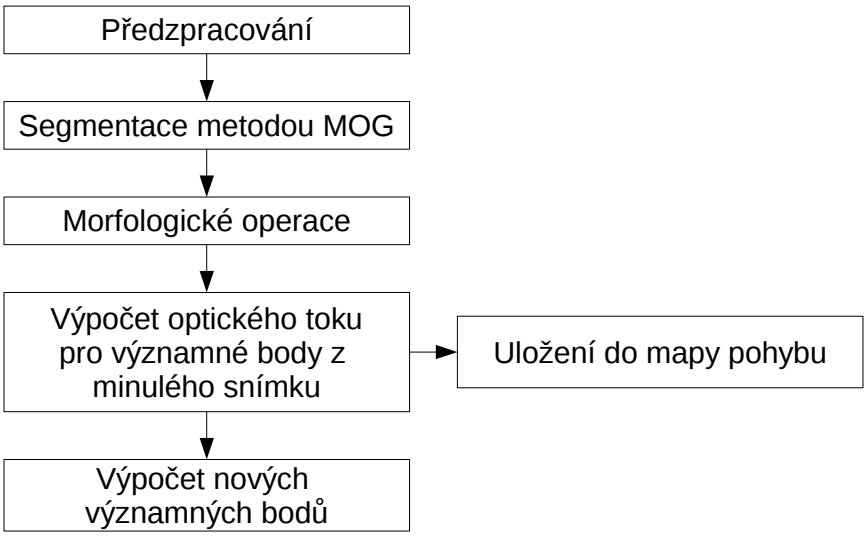
Tab. 4.2: Soubory přidružené k videu

| **Soubor** | **Přípona** |
|---|---|
| Video | Zadá uživatel. |
| Obraz pozadí | .jpg |
| Mapa pohybu | .txt |
| Maska skrytých objektů | .mask.jpg |

### 4.2.3 Popis tříd

#### Zpracování videa

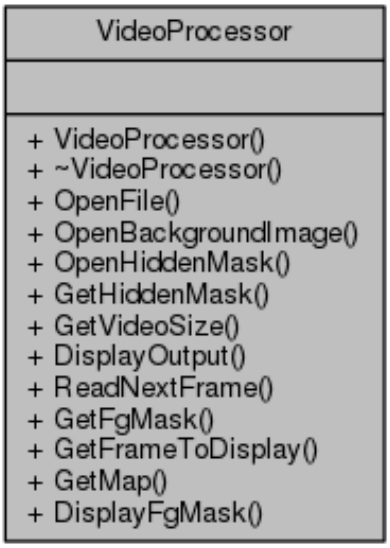
Tato třída je určena k segmentaci jednotlivých snímků videa. Vstupem do této třídy (spojnice 0 na obrázku 4.1) je samotný soubor videa, obraz pozadí a maska určující, kde může být objekt skrytý. Načítání dalších snímků je řízeno metodou *ReadNextFrame*, která provede několik operací přímo nad snímkem. Mezi ně patří především předzpracování, segmentace, výpočet optického toku a detekce významných bodů. Posloupnost jednotlivých kroků zobrazuje diagram 4.2. Z hlavní funkce je rovněž volána metoda *DisplayOutput*, která zobrazuje výstupní obraz s označenými objekty. Jako doplňková metoda při ladění programu slouží *DisplayFgMask*, která zobrazí masku popředí po segmentaci a morfologických operacích.




Obr. 4.2: Posloupnost kroků v metodě pro čtení dalšího snímku

Rozhraní pro třídu *TrackerCore* tvoří metody pro získání ukazatele na masku skrytých objektů, ukazatele na obraz segmentovaného popředí a ukazatele na aktuální snímek. Tato vazba je na obrázku 4.1 znázorněna pod číslem 1.

Při volání konstruktoru je rovněž vytvořena instance třídy *Map*, která je plněna vektory optického toku (vazba 2) mezi jednotlivými snímky. Ukazatel na tuto instanci je možné získat metodou *GetMap*, což je využito pro nastavení ukazatele na mapu v abstraktní třídě trasovacího algoritmu.




Obr. 4.3: UML diagram třídy VideoProcessor

### Mapa pohybu

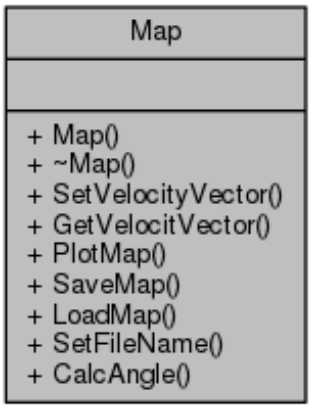
Tato třída implementuje návrh mapy rychlosti a směru pohybu objektů, který byl proveden v kapitole 3. Základní datová struktura je trojrozměrné pole, které uchovává čtyři matice směrových vektorů. Každý z těchto vektorů je reprezentován strukturou *MapCell*:

```
struct MapCell
{
    std::complex<double> Velocity;
    unsigned int Counter;
};
```

Jak je vidět, směrový vektor je implementován pomocí komplexního čísla. Šablona komplexního čísla ze standardní knihovny byla vybrána, protože definuje základní matematické operace s dvojrozměrným číslem a tudíž je možné rychlostní vektor průměrovat stejně jednoduše jako skalární čísla:

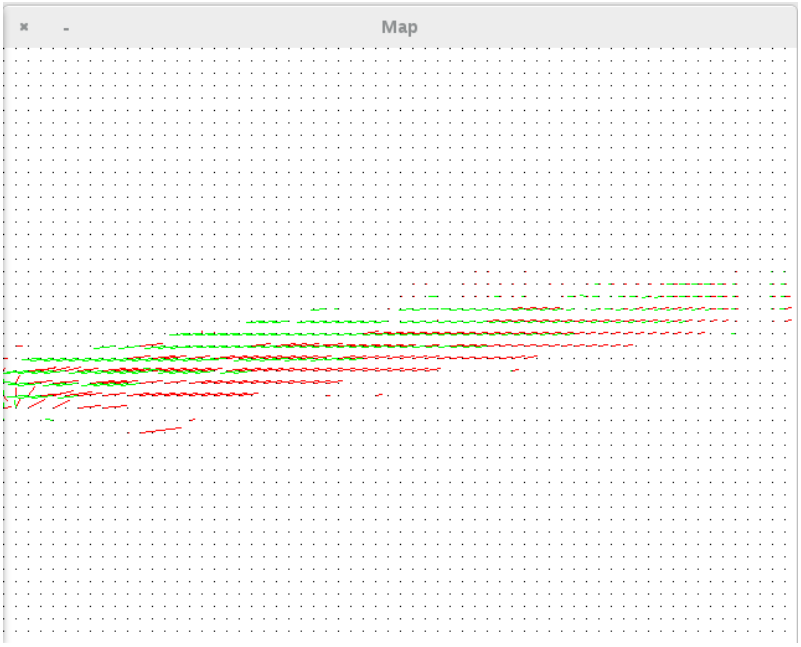
```
const double A = double(N-1)/N;
const double B = double(1)/N;
double tc = (N > 1) ? 0.02 : 0;

auto iir_filter = [A,B,tc](auto x_n_1, auto u_n)
{
    return (A-tc)*x_n_1 + (B+tc)*u_n;
};
```

Rozhraní definované třídou zobrazuje obrázek 4.4.




Obr. 4.4: UML diagram třídy Map

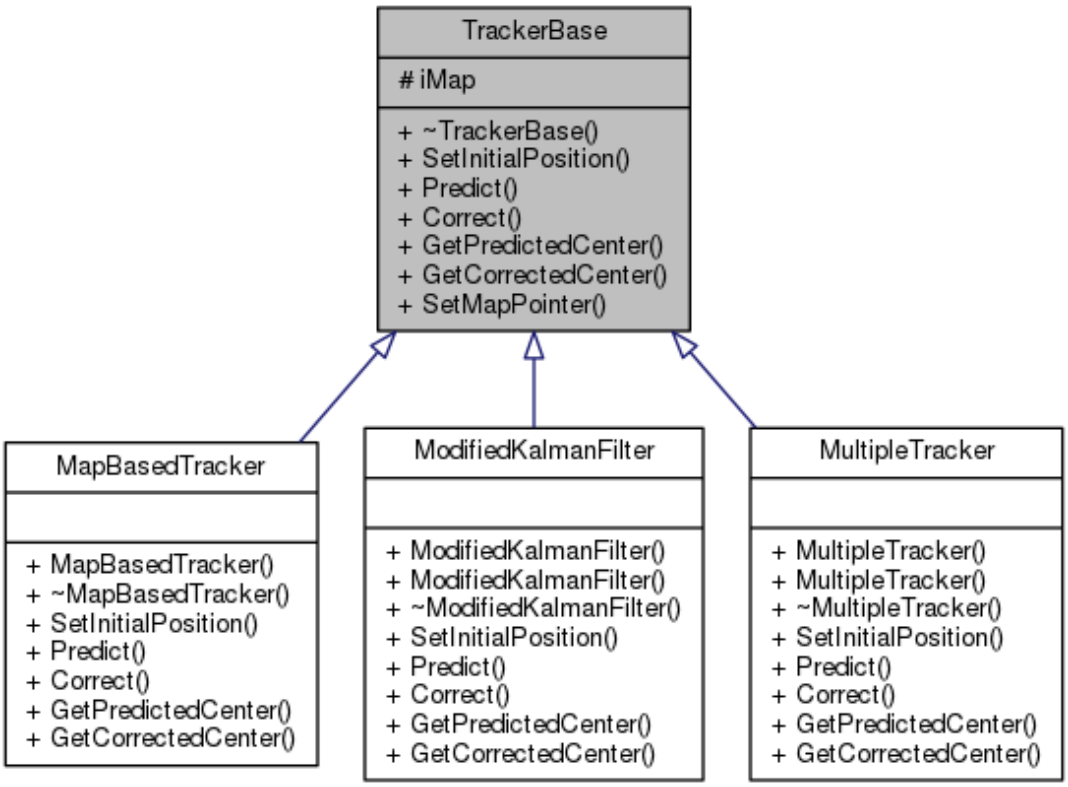
Mapu je možné uložit do souboru a následně ji ze souboru opět načíst. Hodnoty jsou uloženy v prosté textové podobě. Vkládání nových měření je možné pomocí metody *SetVelocityVector*, naopak čtení vektoru na specifikovaném místě je možné pomocí *GetVelocityVector*. Pomocnou metodou při vytváření programu byla *PlotMap*, která zobrazí mapu jako obrázek v dalším okně vedle videa. Na obrázku 4.5 je vidět mapa pohybu na záznamu ulice Úvoz. Jednotlivé barvy vektorů odpovídají různým směrům pohybu objektu.




Obr. 4.5: Vykreslení mapy

### Algoritmus trasování

Vzhledem k tomu, že během této práce byly navrženy dvě metody predikce a korekce pohybu, byla vytvořena abstraktní třída definující základ pro algoritmy trasování. Abstraktní třída se jmenuje *TrackerBase* a z ní jsou odvozeny třídy pro trasování s pomocí mapy *MapBasedTracker*, dále s použitím Kalmanova filtru *ModifiedKalmanFilter* a kombinace obou přístupů *MultipleTracker*. Rozhraní všech tříd slouží pro výpočet predikce a to i opakované, výpočet korekce a získání těchto hodnot. Konkrétní implementaci znázorňuje UML diagram 4.6.




Obr. 4.6: Rodokmen třídy TrackerBase

Implementace jednotlivých metod u třídy *MapBasedTracker* je výrazně jednodušší než u Kalmanova filtru. Predikce pouze načte hodnotu vektoru z mapy a tu přičte k aktuální předpovídané, nikoliv korigované, poloze, tím je zajištěna funkce opakované predikce. Korekce nastaví polohu na hodnotu, která byla změřena.

Jak bylo zmíněno v kapitole o návrhu modelu pro Kalmanův filtr 1.7.3, je možné využít dva různé modely. Typ modelu je pro všechny objekty stejný, ale může se měnit pro různá videa.

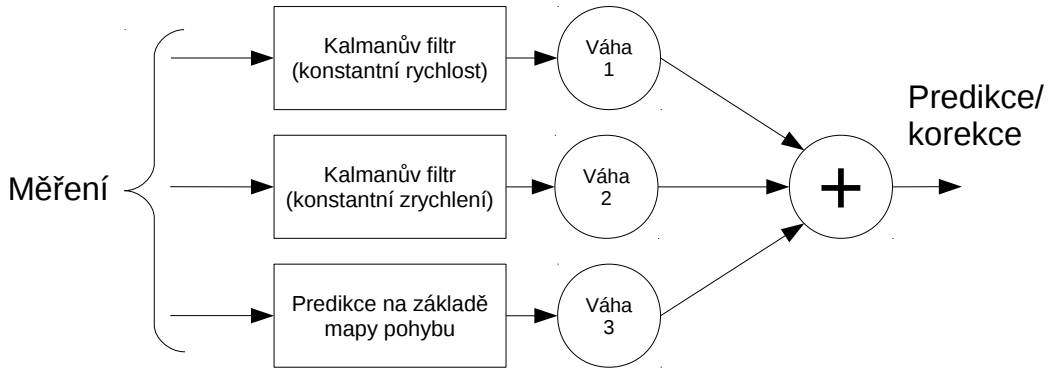
```
enum KalmanFilterType
{
```

```
    E_constant_velocity,
    E_constant_acceleration
};
```

Při inicializaci objektu je vytvořena instance třídy *cv::KalmanFilter*. Jednotlivé matice jsou vyplněny podle teoretického návrhu. Zbytek metod v podstatě jen upravuje rozhraní třídy v knihovně OpenCV tak, aby odpovídalo třídě *TrackerBase*. Jako příklad zde uvádím metody pro korekci a predikci.

```
void ModifiedKalmanFilter::Predict(double aAngle)
{
    iBaseFilter.predict();
}

void ModifiedKalmanFilter::Correct(cv::Point aCenter)
{
    cv::Mat_<float> measurement(2,1) ;
    measurement << aCenter.x, aCenter.y;
    iBaseFilter.correct(measurement);
}
```

Třída *MultipleTracker* kombinuje oba tyto přístupy. Vytváří instance obou filtrů a jejich výstupy následně kombinuje pomocí váženého průměru.




Obr. 4.7: Schématické znázornění spojení více filtrů

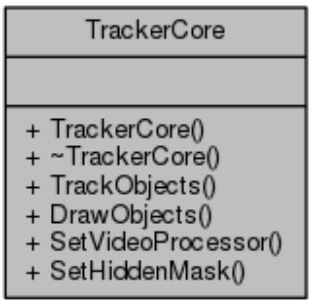
### Pohybující se objekt

Každý objekt, který je ve videu detekován způsobí vznik nové instance třídy *MovingObject* nebo od ní odvozené třídy *MovingMultiObject*. Členské proměnné definují

identifikátor objektu, jeho geometrické rozměry, záznam trasy, vlastní instanci trasovacího algoritmu a proměnné označující objekt jako skrytý. Multi-objekt navíc obsahuje vektor objektů, které jsou v něm obsaženy.

### Trasování objektů

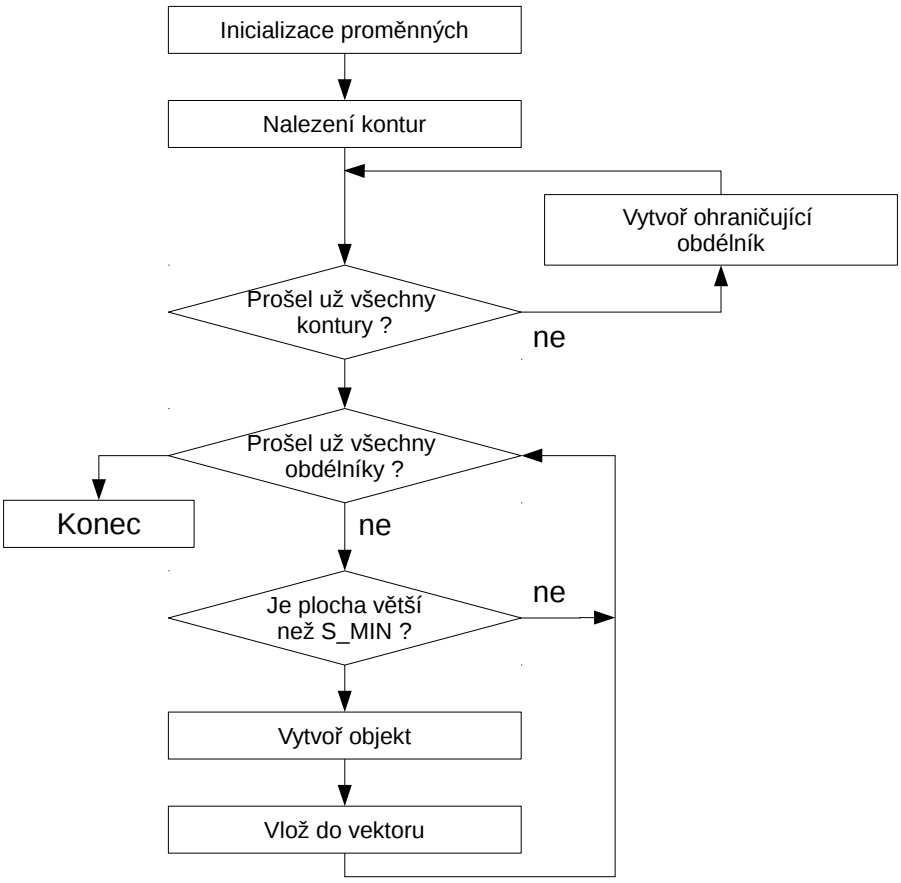
Tato třída řídí proces trasování. Tato úloha je ve skutečnosti redukována na nalezení trasy mezi dvěma po sobě jdoucími snímky. Rozhraní třídy je poměrně jednoduché (obrázek 4.8), ale jednotlivé metody jsou složitější.



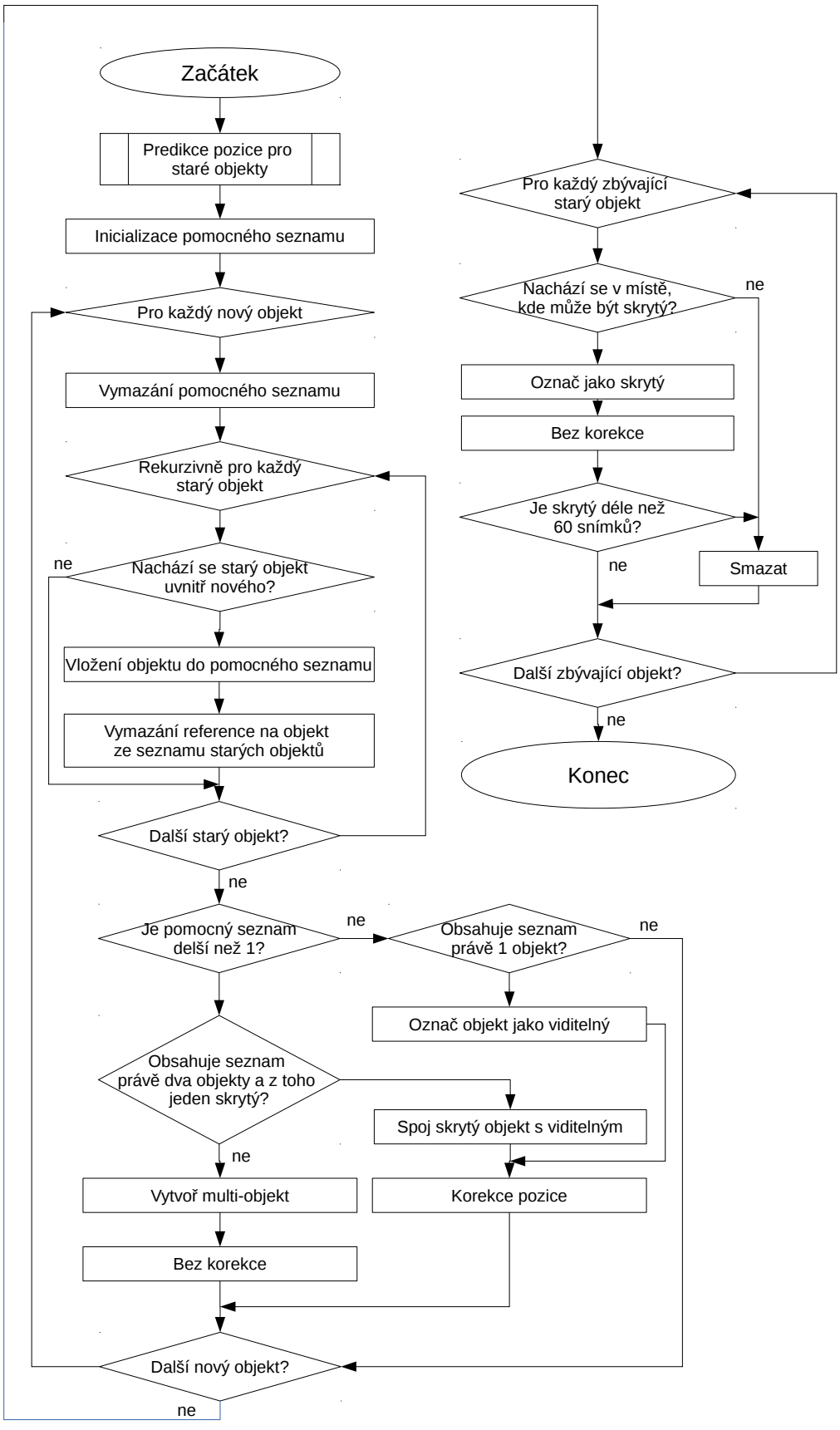

Obr. 4.8: UML diagram třídy TrackerCore

První z těchto metod je *InitObjects*, která provede nalezení objektů v novém segmentovaném snímku, který přebírá od třídy *VideoProcessor*. Diagram algoritmu metody je zobrazen na obrázku 4.9. Nalezení kontur a ohraničení obdélníky je realizováno pomocí funkcí z OpenCV. Proměnná S_MIN je nastavena podle předpokládané velikosti objektů ve videu. Např. pro auta je větší než pro osoby.

Nejdelší metodou v této třídě je *PairObjects*, která porovnává vektor objektů z minulého snímku a vektor objektů v novém snímku a hledá trasy objektů mezi snímky. První jsou k novým objektům přiřazeny staré objekty z minulého snímku. Pokud odpovídá jeden objekt jednomu, tak se pouze zaznamená trasa. Pokud jeden objekt odpovídá více starým, tak vzniká tzv. multi-objekt. Nakonec se projdou všechny zbývající, ztracené, staré objekty. Buď se objekt stane tzv. skrytým, pokud odpovídá masce skrytých objektů, nebo je smazán, pokud masce neodpovídá případně je skrytý už moc dlouho. Do detailu popisuje tento algoritmus obrázek 4.10.




Obr. 4.9: Metoda pro inicializaci objektů ve snímku




Obr. 4.10: Metoda pro sjednocení objektů

## 4.3 Překlad a sestavení

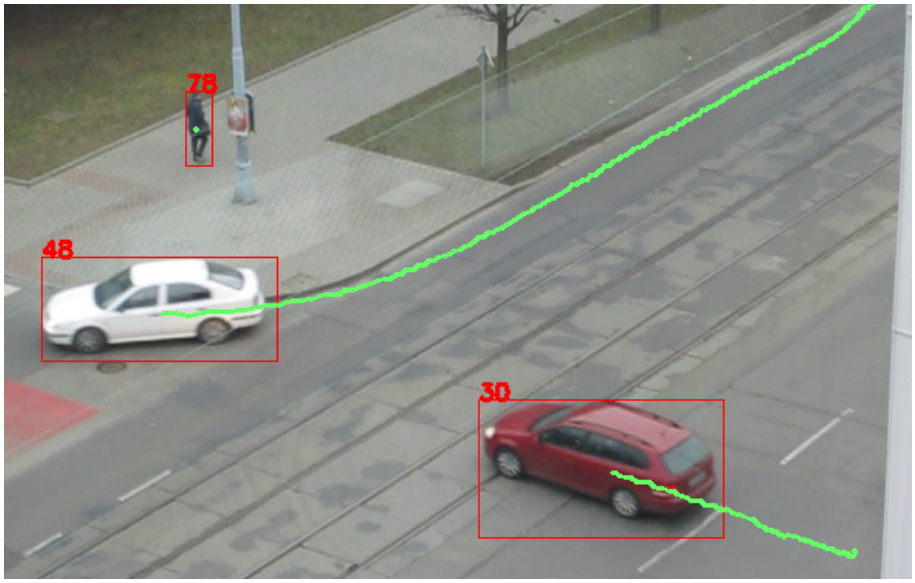
Pro správu a sestavení projektu je využit svobodný software CMake, který je také vydán pod licencí BSD. Jeho výhodou je nezávislost na zvoleném vývojovém prostředí. Tato práce byla vyvíjena v systému GNU/Linux, kde je možné použít CMake k vygenerování Makefile. Obdobně je možné vygenerovat projekt pro další známá prostředí jako je XCode (Apple) nebo Visual Studio (Microsoft). [8].

Pro překlad byl použit Clang++ s definovaným standardem C++14. Nicméně zdrojové kódy obsahují i alternativní řešení pro překlad bez podpory standardu C++11 a C++14. Pokud je využita tato možnost, preprocesor vypíše upozornění na použití alternativních částí programu.

# 5 ZÁVĚR

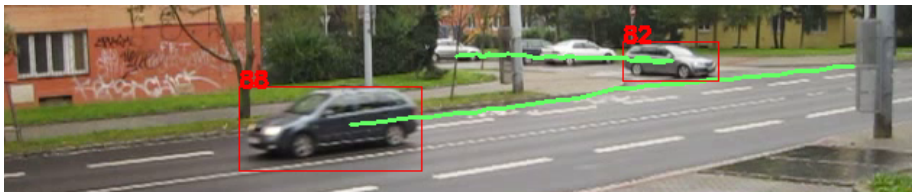
Cílem práce bylo nastudovat zpracování videozáznamu pomocí algoritmů počítačového vidění, navrhnout způsob mapování pohybu ve scéně a pořídit databázi testovacích záznamů. Tyto body zadání byly shrnuty v 1. až 3. kapitole. Jako klíčový postup pro sledování pohybu byla zvolena metoda segmentace pomocí směsi Gaussových rozdělení (MOG) a modelování a předpovídání pohybu pomocí Kalmanova filtru.

Dalším úkolem bylo realizovat teoretické postupy vytvořením programu pro sledování pohybu objektů ve scéně. Jako prostředek byla zvolena knihovna algoritmů počítačového vidění OpenCV (Open Computer Vision), která poskytuje rozhraní pro jazyk C++. Implementaci programu popisuje kapitola 4. Podařilo se vytvořit program, který je za určitých podmínek schopen sledovat pohyb ve scéně. Podmínkami jsou především stálé světelné podmínky, minimum rušivých elementů v obraze a statická poloha kamery. V nejjednodušším případě se všechny objekty pohybují odděleně a daleko od sebe, takovýto záznam je zobrazen na obrázku 5.1. Složitější scéna je zaznamenána na ulici Úvoz, kde auta míjejí sloupy kolem silnice a také se navzájem překrývají 5.2. Na tomto videu se uplatňuje především predikce pohybu, díky které je správně trasován automobil i když je skrytý za sloup a následně, když se míjí s vozidlem na hlavní cestě.



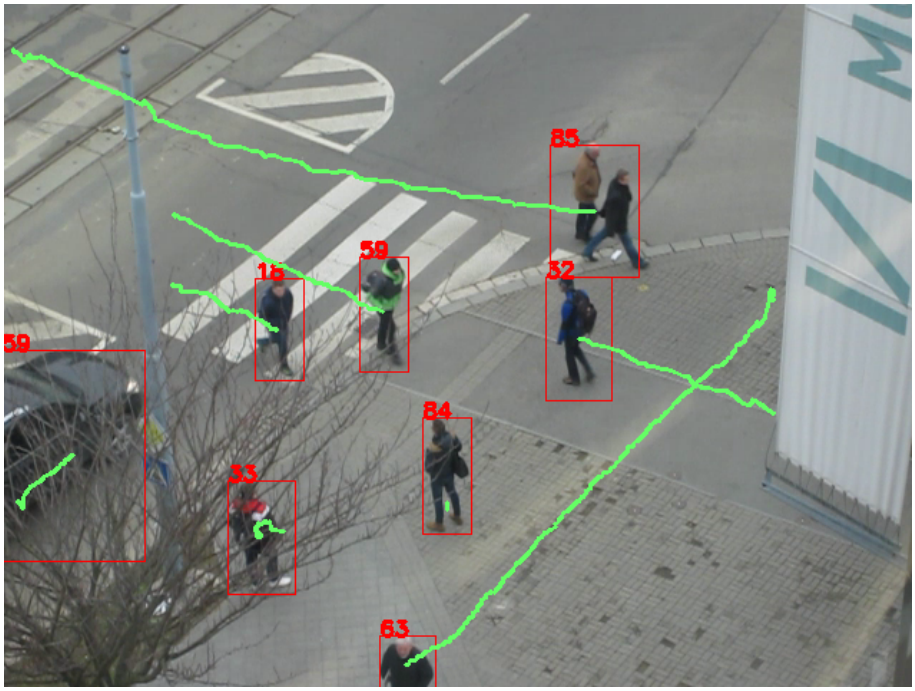

Obr. 5.1: Oddělené objekty daleko od sebe

Větší množství objektů se pohybuje na záznamech pořízených před budovou Moravské zemské knihovny. Na těchto záznamech jsou rušivé prvky v podobě stromů




Obr. 5.2: Uplatnění predikce pohybu

částečně překrývajících silnici a sloupů pro troleje. Program při správné funkci zaznamenává obrázek 5.3. Všechny objekty jsou správně odděleny a je za nimi označena trasa pohybu. Ovšem u záznamů před knihovnou se u některých objektů vyskytují chyby segmentace. Jako příklad je zde uveden obrázek 5.4, na kterém je automobil přepůlen sloupem trolejového vedení, a obrázek 5.5, kde automobil při čekání na přechodu zanikl v pozadí kvůli příliš rychlé adaptaci modelu pozadí.

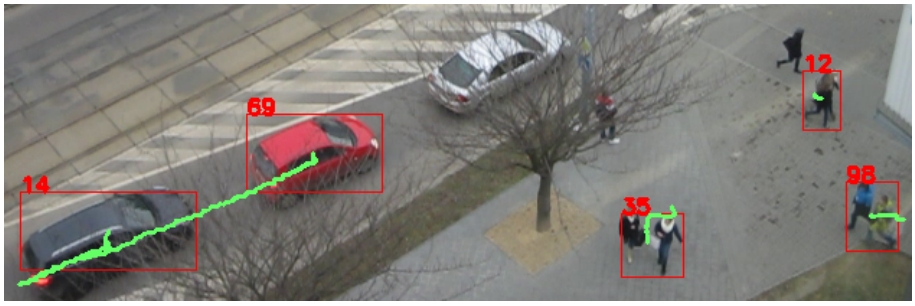



Obr. 5.3: Větší množství pohybujících se objektů

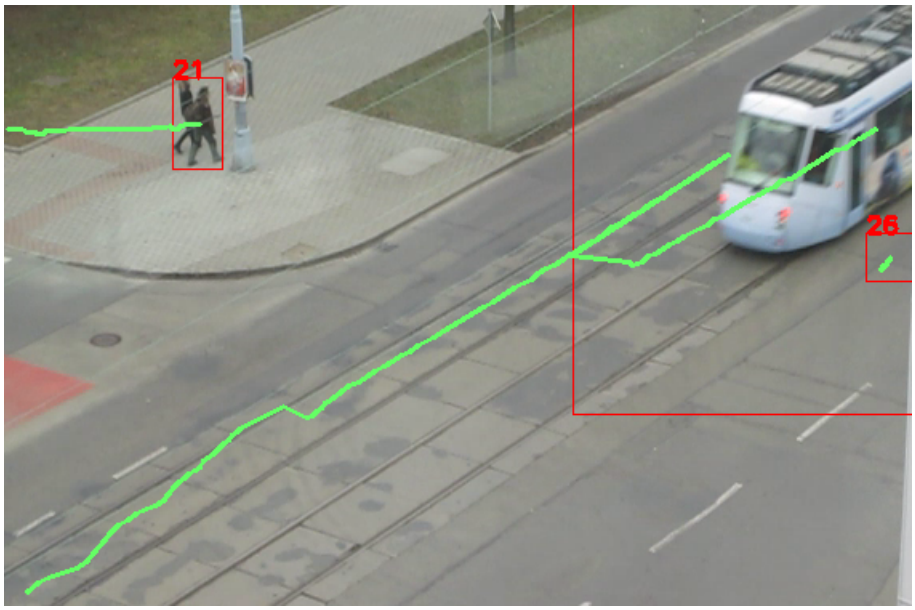
Velmi rušivě působí v záznamech také městská hromadná doprava. Trolejbusy a tramvaje většinou zabírají celou scénu nebo její převážnou část. I přesto je program schopen do jisté míry trasovat pohyb těchto vozů 5.6.

Program ve formě zdrojových kódů a souboru pro automatizaci sestavení je umístěn v příloze spolu s testovacími záznamy a vyexportovanými záznamy, na kterých je vidět trasování pohybu bez nutnosti spouštět program.


Obr. 5.4: Chybná segmentace způsobená sloupem v záznamu




Obr. 5.5: Příliš rychlé přizpůsobení pozadí




Obr. 5.6: Trasování příliš velkého objektu

Dalším rozvojem práce by mohlo být vylepšení segmentace, která je v současném

stavu velmi náchylná na nastavení na míru dané scéně a světelným podmínkám. Taktéž je velmi rušivý jakýkoliv otřes kamery, například způsobený silným větrem při natáčení. Způsob predikce pohybu rovněž obsahuje prostor pro další rozvoj práce. Především pro použití několika modelů pohybu by mohl být navržen algoritmus, který by sám vybral nejvhodnější model v danou chvíli. Prostor pro vylepšení se rovněž nachází ve využití výpočetního výkonu grafické karty, jejíž podpora je už v OpenCV implementována.

# LITERATURA


[1] HLAVÁČ, Václav a Milan ŠONKA. *Počítačové vidění*. Praha: Grada, 1992, 272 s. ISBN 80-854-2467-3.

[2] Background subtraction. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-10-31]. Dostupné z: `www.en.wikipedia.org/wiki/Background_subtraction`

[3] PICCARDI, M. Background subtraction techniques: a review. *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)* [online]. IEEE, 2004, s. 3099-3104 [cit. 2014-10-31]. DOI: 10.1109/ICSMC.2004.1400815. Dostupné z: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1400815`

[4] Contour line. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-11-12]. Dostupné z: `http://en.wikipedia.org/wiki/Contour_line`

[5] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 3rd ed. Toronto: Thomson, 2008, xxv, 829 s. ISBN 978-0-495-08252-1.

[6] BRADSKI, Gary R. *Learning OpenCV*. Sebastopol: O'Reilly, c2008, xvii, 555 s. ISBN 978-0-596-51613-0.

[7] OpenCV. *Open source computer vision* [online]. [cit. 2014-10-31]. Dostupné z: `www.opencv.org`

[8] Overview: About CMake. *CMake* [online]. 2015 [cit. 2015-04-10]. Dostupné z: `http://www.cmake.org/overview/`

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

| | |
|---|---|
| I | Image - snímek |
| B | Background - pozadí |
| Th | Threshold - prahová hodnota |
| MOG | Mixture of Gaussians - metoda segmentace |
| GIMP | The GNU image manipulation program - program pro úpravu obrázků |
| GNU | GNU's Not Unix - GNU Není Unix, rekurzivní zkratka |
| Matlab | Matrix Laboratory - maticová laboratoř, program pro pro vědeckotechnické výpočty |
| CV | Computer Vision - počítačové vidění |
| BSD | Berkeley Software Distribution - operační systém a k němu náležící licence pro svobodný software |
| GUI | Graphical User Interface - Grafické uživatelské rozhraní |
| UML | Unified Modeling Language - grafický jazyk pro vizualizaci a dokumentaci programů |

# SEZNAM PŘÍLOH

# A OBSAH PŘILOŽENÉHO CD

## A.1 Dokumentace

Nachází se ve složce *dokumentace*. Jedná se o dokumentaci generovanou programem Doxygen přímo ze zdrojových kódů programu. Dokumentace je ve formátu html.

## A.2 Zdrojový kód

Ve složce *zdrojovy_kod* se nachází soubor main.cpp, CMakeList.txt a složka modules, která obsahuje všechny potřebné třídy. Poslední složka se nazývá *tests* a obsahuje testy jednotlivých tříd, které byly použity při vývoji programu a nejsou tedy přímo součástí hlavního programu.

## A.3 Vzorová videa

Příloha obsahuje dvě složky: *videa_vstupni* a *videa_vystup*. První složka obsahuje soubory s video záznamem a všechny ostatní přidružené soubory potřebné ke spuštění detekce pohybu zabalené v archivu pro každé video zvlášť. Druhá složka obsahuje videa vygenerovaná z programu. Tato videa jsou přiložena jako ukázka funkce programu.