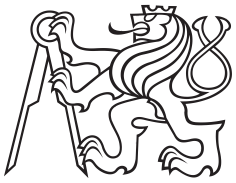
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Detekce pohybujících se objektů v obraze z pohyblivé kamery

Filip Tefr

Vedoucí: Ing. Jan Chudoba
Studijní program: Kybernetika a robotika
Květen 2019

Poděkování

Chtěl bych poděkovat panu ing. Janu Chudobovi za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 24. května 2019

Abstrakt

Cílem této bakalářská práce je vytvořit algoritmus, který bude schopen detekovat pohybující se objekty dynamicky se pohybující kamerou. Algoritmus byl vytvořen v programovacím jazyce C++ s využitím volně šiřitelné knihovny OpenCV. Metoda byla optimalizována pro objekty malé velikosti vzhledem k velikosti obrazu, aby mohla být kamera umístěna na dronu či helikoptéře. Hlavním stavebním kamenem algoritmu je metoda výpočtu optického toku, která se snaží odhadnout vektor rychlosti daného pixelu. Celá metoda je rozdělena do čtyř nezávislých bloků, které zjednodušují implementaci.

Klíčová slova: počítačové vidění, kalibrace kamery, detekce pohybu dynamickou kamerou, optický tok

Vedoucí: Ing. Jan Chudoba
Skupina inteligentní a mobilní robotiky,
Jugoslávských partyzánů 3,
CIIRC,
Praha 6

Abstract

The goal of this bachelor thesis is to create an algorithm, which can detect moving object from a dynamic moving camera. The algorithm was developed in programming language C++ using freely available library OpenCV. The method was optimazed for small object relative to the size of an image. The main method used in these thesis is optical flow, which tries to estimate velocity vector of given pixel. The algorithm was divided into four independent parts, which simplify the implementation.

Keywords: computer vision , camera calibration, dynamic camera movement detection, optical flow

Title translation: Moving Object Detection in an Image from a Moving Camera

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Cílem této práce je vytvořit algoritmus oblasti počítačového vidění, který bude schopen detekovat pohybující se objekty běžnou kamerou. Kamera se bude moci pohybovat, proto se bude muset vliv pohybu kamery na detekci eliminovat. Metoda bude optimalizována pro objekty malé velikosti vzhledem k velikosti obrazu, jelikož objekty budou zachyceny z výšky. Algoritmus bude implementován s využitím volně dostupné knihovny OpenCV [3] v programovacím jazyce C++. Hlavním stavebním kamenem algoritmu bude metoda výpočtu optického toku [8], která se snaží odhadnout pohyb jednotlivých pixelů mezi dvěma obrázky. Možné využití výsledku práce může být např. hlídaní rozsáhlých území z výšky autonomními helikoptérami či drony, které ušetří práci lidským hlídkám.

Práce bude rozdělena na několik částí. V kapitole 2 budou uvedeny teoretické předpoklady k vytvoření algoritmu (např. dostupné metody či kalibrace kamery). V kapitole 3 bude popsán postup kalibrace kamery v OpenCV a také detailně popsán použitý algoritmus. V závěru 4 práce budou provedeny praktické experimenty a bude diskutována funkčnost a úspěšnost algoritmu.

Kapitola 2

Teoretická část

Metody pro detekci pohybu kamerou lze rozdělit na dva typy úloh dle pohybu použité kamery. V první skupině metod používáme statickou kameru (kamera je stále na jednom místě). Tomuto tématu se např. věnuje článek *Image Difference Threshold Strategies and Shadow Detection* [10], kde je použita metoda obrázkové diference (modifikovaná verze je popsána v kapitole 2.1.1). K tomuto účelu lze také použít odečítání pozadí [9], která se snaží ze sekvence obrázků odebrat pozadí (statické objekty). Tyto metody lze např. použít ke sledovaní hustoty dopravy statickou kamerou.

Ve druhé skupině úloh se kamera pohybuje. Tento úkol je značně složitější než předchozí. Zde se vyskytují dva nezávislé pohyby - pohyb kamery a pohyb samotných objektů. Tyto pohyby jsou smíchány dohromady, a proto se musí při detekci pohybu rozlišit. Z tohoto důvodu nejsou metody z předchozí skupiny pro tento účel vhodné, nebo je nutné je značně modifikovat. Tomuto úkolu se např. věnuje článek *Detecting moving objects using a single camera on a mobile robot in an outdoor environment* [6], kde je diskutováno použití detektoru příznaků (kapitola 2.1.1) a optického toku (kapitola 2.1.2).

2.1 Dostupné metody

Metody na řešení problému detekce pohybu v obraze pohybující se kamerou lze rozdělit na 2 hlavní skupiny. První skupinu tvoří metody, které pracují se dvěma popř. více po sobě jdoucími snímky. Snímky je třeba srovnat tak, aby

se pohybující objekty vyskytovaly na snímcích na stejném místě (kompenzovat pohyb kamery). Na tyto snímky je poté použita metoda obrázkové diference (viz kapitola 2.1.1).

Do druhé skupiny lze zařadit metody, které pracují na jiném principu než je obrázková diference (hledání příznaků, objektů nebo dále použitý optický tok). Pro správnou funkci všech níže uvedených metod je třeba kompenzovat zkreslení způsobené kamerou. K tomuto účelu slouží proces kalibrace (viz kapitola 2.2).

2.1.1 Metoda obrázkové diference

Každý obrázek velikosti w a h lze reprezentovat maticí $A \in M^{w,h}$. Proto je v některých případech možné pracovat s obrázky podobně jako s maticemi. Toho využívá následující metoda založená na odčítání odpovídacích si pixelů ve dvou po sobě jdoucích obrázků. Nechť $A, B, C \in M^{w,h}$ jsou obrázky, M je matice, která říká, zda se daný pixel pohybuje.

Algoritmus 1 Obrázková diference

Require: Images A, B
Ensure: Move map M
  $C \leftarrow |A - B|$
  for i = 0 to ImageWidth **do**
    for i = 0 to ImageHeight **do**
      if $C_{i,j} \geq treshold$ **then**
        $M_{i,j} \leftarrow true$
      end if
    end for
  end for

Jak je vidět, algoritmus provede rozdíl obrázků A a B. Pixel i, j je klasifikován jako pohyblivý, je- li hodnota rozdílu větší než předem stanovená hodnota (treshold).

Tato metoda funguje spolehlivě jen za předpokladu, že obrázky jsou dobře srovnané, tzn. nepohybující se objekty leží v obrázku na stejném místě, tzn. že se kamera nepohybuje. Pokud tento předpoklad splněn není, lze korespondenci snímků zajistit např. některou z níže uvedených metod.

2D Korelace

Míru podobnosti dvou obrázků lze definovat diskrétní 2D korelací.

Definice 2.1. Nechť f, $g \in M^{w,h}$, pak definujeme korelaci $(f * g)_{x,y} = \sum_{i=0}^{w-1} \left(\sum_{j=0}^{h-j} f_{i,j}\, g_{i+x,j+y} \right)$

Tam, kde korelace nabývá svého maxima, jsou si obrázky nejvíce podobné. Ke srovnání obrázků je tedy nutné vypočítat korelaci dvou obrázků a následně starý obrázek posunout o argument maxima x_{max}, y_{max} výše uvedené funkce. K detekci pohybu na těchto obrázcích je pak možné použít např. metodu obrázkové diference (viz kapitola 2.1.1). Touto metodou lze detekovat pouze posun, nikoliv rotaci či další transformace. Její velkou nevýhodou je také velká výpočetní náročnost.

Detektor příznaků

Funkce metody je založena na hledání dobře sledovatelných bodů ve dvou po sobě jdoucích obrázcích. Jelikož mezi příznaky nalezené v obrázcích neexistuje žádná vazba, je nutné je spárovat, tzn. přiřadit k sobě korespondující příznaky. Po spárování bodů dochází k velkému množství chyb, proto je nutné brát na chyby ohled.

Obrázek 2.1: Ukázka detektoru příznaků [1]

Existuje velké množství různých detektorů (např. *SIFT* [7], *SURF* [2]), které lze k této funkci použít. Na obrázku 2.1 je ukázka detektoru *SURF*. S využitím těchto detektorů a nalezených dvojic lze najít obecnou transformaci, kterou

[1] Obrázek převzat z dokumentace OpenCV `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html`

lze srovnat použité obrázky. K nalezení této transformace je možné použít např. algoritmus RANSAC (Random Sample Consensus) [14], který je odolný vůči chybám. Výhodou této metody je detekovaní složitějších transformací (nejen posun, ale i rotace, zkosení či změna měřítka).

2.1.2 Optický tok

Tato metoda bude dále použita, proto popíšu všechny její principy podrobněji. Metoda se snaží odhadnout vektor rychlosti pro dané pixely. Z velikosti těchto vektorů lze zjistit, jak moc se daný pixel pohnul. Pokud bude pohyb velký, lze ho vyhodnotit jako reálný. Princip metody je následující [15]: souřadnice jednotlivých pixelů označíme $[x\ y]^T$, intenzitu pixelu v čase t $I(x, y, t)$. Za čas Δt se pixel $[x\ y]^T$ posune o $[\Delta x\ \Delta y]^T$, tedy můžeme psát

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.1}$$

Předpokládejme, že pohyb pixelů je malý. Intenzitu pixelu můžeme aproximovat Taylorovým polynomem prvního řádu:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{2.2}$$

Z výše uvedených rovnic plyne

$$\begin{aligned} \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t &= 0 \\ \frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} &= 0 \\ \frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y &= -\frac{\partial I}{\partial t} \end{aligned} \tag{2.3}$$

kde V_x a V_y jsou složky vektoru rychlosti ve směru osy x resp. y. Označíme-li příslušné derivace obrázku I_x, I_y a I_t, pak výše uvedenou rovnici můžeme přepsat následovně

$$\nabla I.\vec{V} = -I_t \tag{2.4}$$

Což nám dává výslednou rovnici s neznámým vektorem $\vec{V}$. Jelikož výše uvedená rovnice má 2 neznámé, nelze tuto rovnici řešit bez dalších podmínek. Volbou těchto podmínek dostáváme různé varianty metody.

Metoda, která bude dále použita, byla publikována *Brucem D. Lucasem* a *Takeo Kanadem* [8]. Metoda využívá skupiny (tzv. okna) sousedních pixelů v obrázku. Označíme-li souřadnice pixelů v okně q_i, pak dle rovnice 2.4 platí

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t \\ (q_1)I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &. \\ &. \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{2.5}$$

Uvedené rovnice lze přepsat jako soustavu lineárních rovnic $Ax = b$, kde

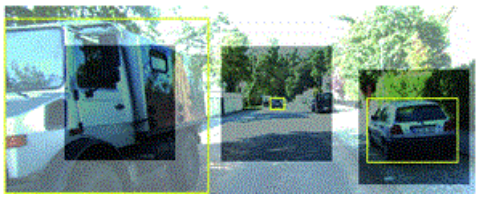
$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_y(q_2) & I_y(q_2) \\ . & . \\ . & . \\ I_y(q_n) & I_y(q_n) \end{bmatrix}, \; b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ . \\ . \\ -I_t(q_n) \end{bmatrix}, \; x = \begin{bmatrix} V_x \\ V_y \end{bmatrix}. \tag{2.6}$$

Jedná se o předurčenou soustavu, kterou lze přibližně řešit metodou nejmenších čtverců. Výsledný vektor rychlosti je roven

$$x = \begin{bmatrix} V_x \\ V_y \end{bmatrix} = (A^T A)^{-1} A^T b. \tag{2.7}$$

2.1.3 Detektor objektů

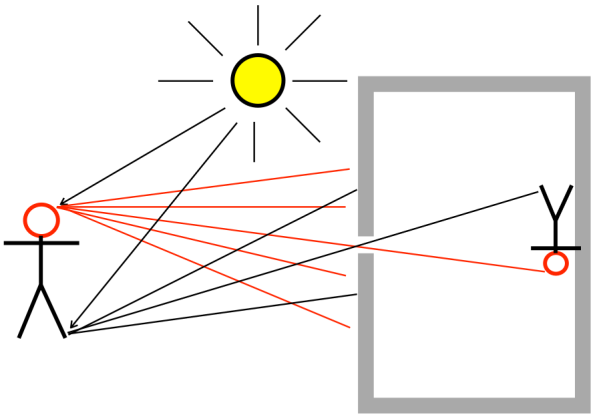
Kromě příznaků (viz kapitola 2.1.1), lze v obraze detekovat samotné objekty (budovy, auta, lidi, ...). K tomuto účelu lze použít některou již sestavenou neuronovou síť (např. R-CNN [5]), naučenou na detekci těchto objektů. K tomuto účelu je potřeba velké množství obrázků (trénovací množiny), na kterých jsou objekty ručně označeny. Ukázka funkce neuronové sítě je zobrazena na obrázku 2.2 [4]. Tato síť se použije na nalezení objektů na snímcích a podobně jako u předchozího algoritmu je mezi sebou spárovat. Tato metoda je odolnější proti falešným detekcím než předchozí metody, avšak je možné použít ji jen na specifické (předem známé) druhy objektů. Její nevýhodou je také její velká výpočetní náročnost, proto se touto metodou nebudu zabývat.



Obrázek 2.2: Detekce aut neuronovou sítí [4]

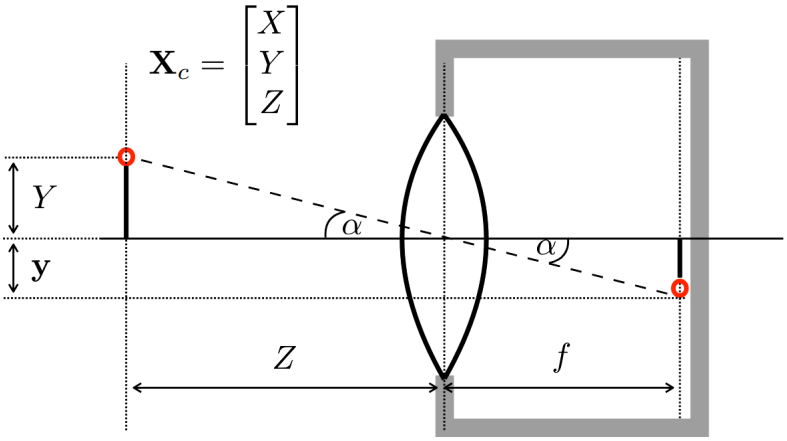
2.2 Kalibrace kamery

Aby bylo možno některou z výše uvedených metod použít, je nejdříve nutné kameru zkalibrovat, což znamená zjistit parametry kamery jako je např. ohnisková vzdálenost a kompenzovat radiální a tangetální zkreslení. Jako model kamery použijeme tzv. dírkovou kameru [16], která dobře reprezentuje většinu moderních kamer. Tento model se skládá z bodového zdroje světla, zobrazovaného objektu a samotné kamery.



Obrázek 2.3: Dírková kamera [16]

Ke zvýšení viditelnosti zobrazovaného objektu v kameře lze zvětšit průměr dírky, což sice zlepší viditelnost objektu, ale také ho rozostří. To je způsobeno tím, že paprsky, které by měly dopadat na stejné místo, jíž dopadají jinam z důvodu větší dírky. Ke zaostření obrazu je třeba použít spojnou čočku, která ale do obrazu vnáší další zkreslení. S využitím obrázku 2.4



Obrázek 2.4: Dírková kamera se spojnou čočkou [16]

odvodíme zobrazovací rovnici kamery. Označíme-li souřadnice bodu v prostoru X, Y, Z a souřadnice bodu v kameře x, y, pak platí:

$$\begin{aligned} \tan\alpha = \frac{Y}{Z} &= \frac{y}{f} \\ \lambda x &= fX \\ \lambda y &= fY \\ \lambda &= Z \end{aligned} \tag{2.8}$$

Kde λ je škálovací faktor reprezentující nejednoznačnost transformace (všechny body na spojnici zobrazované bodu a středu čočky se zobrazí do stejného bodu). Rovnice 2.8 lze zapsat maticově

$$\lambda \begin{bmatrix} \frac{x}{\lambda} \\ \frac{y}{\lambda} \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.9}$$

Jelikož žádná čočka není dokonalá (ohnisková vzdálenost není ve všech směrech konstantní), přepíšeme rovnici 2.9 následovně

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.10}$$

kde f_x a f_y jsou ohniskové vzdálenosti ve směrech os x resp. y a bod $[o_x\ o_y]^T$ je optický střed čočky. Dále je třeba kompenzovat radiální zkreslení, které bude hledáno ve tvaru [1]

$$\begin{bmatrix} x_{opravene} \\ y_{opravené} \end{bmatrix} = \begin{bmatrix} x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{bmatrix} \tag{2.11}$$

a tangentální zkreslení ve tvaru

$$\begin{bmatrix} x_{opravené} \\ y_{opravené} \end{bmatrix} = \begin{bmatrix} x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{bmatrix} \tag{2.12}$$

kde r je vzdálenost bodu od optického středu a

$$[k_1\ k_2\ k_3\ p_1\ p_2] \tag{2.13}$$

jsou parametry zkreslení použité kamery. Návod, jak tyto parametry zjistit, bude uveden v kapitole 3.2.

Kapitola 3

Implementační část

Všechny algoritmy, které budou použity k detekci pohybu, budou implementovány s využitím knihovny OpenCV [3]. OpenCV je multiplatformní volně šířitelná knihovna, která obsahuje mnoho algoritmů a funkcí, které si týkají počítačového vidění a strojového učení. Většina funkcí[1] podléhá licenci BSD, z čehož plyne, že je možné knihovnu použít zdarma i komerčně. Obsahuje funkce k načítaní a ukládání obrázků (core.hpp), jejich zpracování (imgproc.hpp), či grafické prostředí určené k interakci s uživatelem (highGUI.hpp). API knihovny je dostupné v mnoha programovacích jazycích (C++, Python, Java či MATLAB). Ke zrychlení je také možné využít GPU akceleraci. Tímto se však tato práce nebude zabývat. K implementaci bude použito nativní C++ API z důvodu požadavku na zpracování obrazu v reálném čase.

[1] Např. detektory *SURF* či *SIFT* jsou šířeny pod jinou licencí

3.1 Ukázkový kód OpenCV

Následující kód provede načtení obrázku a jeho zobrazení v okně. Příklad je převzat z dokumentace OpenCV [2].

:

```
1  #include <opencv2/core/core.hpp>
2  #include <opencv2/highgui/highgui.hpp>
3  #include <iostream>
4
5  using namespace cv;
6  using namespace std;
7
8  int main( int argc, char** argv )
9  {
10     if( argc != 2)
11     {
12      cout <<"␣Usage:␣display_image␣
           ImageToLoadAndDisplay" << endl;
13      return -1;
14     }
15
16     Mat image;
17     // Načtení obrázku ze souboru
18     image = imread(argv[1], CV_LOAD_IMAGE_COLOR);
19
20   // Kontrola vstupu
21     if(! image.data )
22     {
23         cout <<  "Could␣not␣open␣or␣find␣the␣image" <<
              std::endl ;
24         return -1;
25     }
26
27   // Vytvoření okna pro zobrazení obrázku
28     namedWindow( "Display␣window", WINDOW_AUTOSIZE );
29     // Zobrazení obrázku v~okně
30     imshow( "Display␣window", image );
31
32   // Čekání na stisk klávesy
33     waitKey(0);
34     return 0;
35 }
```

[2] https://docs.opencv.org/2.4/doc/tutorials/introduction/display_image/display_image.html

3.2 Kalibrace kamery v OpenCV

V sekci 2.2, byl popsán teoretický postup, jak lze kameru zkalibrovat. Jak již bylo řečeno, dobrá kalibrace kamery je důležitým krokem při zpracování obrazu. Kalibraci lze provádět manuálně či automaticky. Proces většinou spočívá ve vytvoření snímků určitého objektu z různých vzdáleností a úhlů. Při manuální kalibraci se uživatel dle svého uvážení snaží přiřadit určitým bodům na objektu souřadnice původního bodu ve 3D světě. Tento proces je pracný, a není moc přesný, protože je ovlivněn lidským faktorem. Z těchto důvodů je vhodné tento proces zautomatizovat.

S využitím knihovny OpenCV bude navržen program s následujícími funkcemi:

- Načtení obrázků ze složky, nebo jejich vytvoření web kamerou
- Nalezení parametrů kamery (2.10) a matice kamery (2.13)
- Uložení těchto parametrů do YAML souboru pro pozdější použití

Kalibraci kamery lze rozdělit na několik úkonů, které budou nadále popsány. Ještě je třeba si zavést 2 souřadnicové systémy, ve kterém budeme body popisovat - světový souřadnicový systém pevně spojen se zemí a souřadnicový systém kamery spojen s obrázkem.

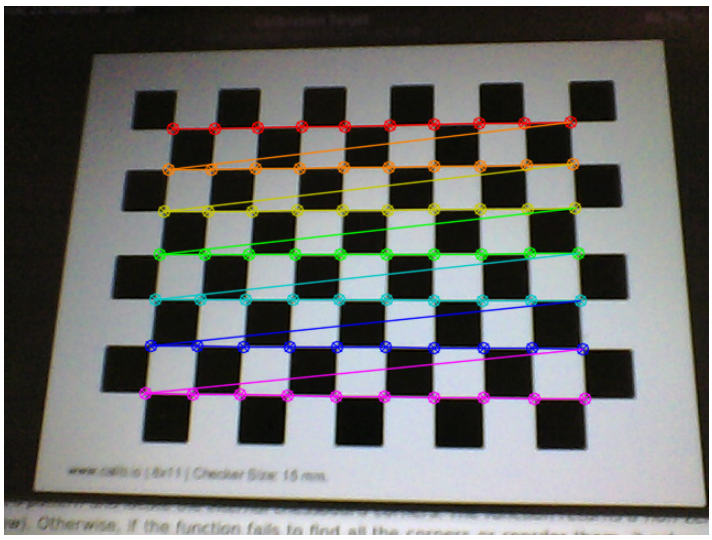
3.2.1 Snímky kalibračního obrazce

Aby bylo možné zautomatizovat přiřazení souřadnic bodů v souřadnicovém systému kamery a reálného světa, je nutné při kalibraci pracovat s přesně definovanými objekty. OpenCV umí pracovat se dvěma objekty - šachovnice a kružnice. V této práci se bude pracovat se šachovnicí.

Prvním bodem kalibrace je vytvoření dostatečného množství snímků stejné šachovnice. Je nutné, aby se snímky lišily nejen vzdáleností od šachovnice, ale aby byly vyfoceny s různých úhlů. Dobré je také pracovat s obdélníkovou šachovnicí lichých rozměrů.

3.2.2 Přiřazení souřadnic

Jako body, které budeme hledat pro kalibraci, si vybereme rohy šachovnice. K nalezení těchto bodů v obrázku nám mohou pomoci 2 funkce v OpenCV, *cv::findChessboardCorners* a *cv::cornerSubPix*. První z vyjmenovaných funkcí nalezne souřadnice rohů v šachovnici daných rozměrů, druhá funkce slouží ke zpřesnění těchto souřadnic. Vstupní obrázek musí být černobílý. Ke konverzi barevného obrázku do černobílého lze využít funkci *cv::cvtColor*. Souřadnice rohů šachovnice ve světovém souřadnicovém systém systému lze zvolit jednoduše, např [0, 0], [0, 1] atd.



Obrázek 3.1: Nalezené rohy šachovnice

3.2.3 Kalibrace kamery

Posledním bodem kalibrace je nalezení parametrů kamery (rovnice 2.13). K tomu nám poslouží funkce *cv::calibrateCamera*. Vstupem jsou již získané (kapitola 3.2.2) souřadnice bodů ve světovém souřadnicovém systému a souřadnicovém systému kamery. Výstupem této funkce jsou parametry kamery a také transformační matice použitých souřadnicových systémů.

3.2.4 Ukázkový kód

Na závěr této části uvedu část kódu, který je určen ke kalibraci kamery. Méně důležité části budou vynechány. Celý kód bude uveden v příloze.

:

```
...
for(auto path: paths){
  // Načtení obrázku
  img = imread(path);
  // Převedení na do černobíle
  cvtColor(img, grey, CV_BGR2GRAY);
  // Nalezení rohů
  if(bool ret = findChessboardCorners(grey, Size(w, h),
     cornerBuf)){
    // Zpřesnění souřadnic rohů
      cornerSubPix(grey, cornerBuf, Size(10, 10),
         Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS +
         CV_TERMCRIT_ITER, 30, 0.1));
      // Uložení souřadnic
      corners.push_back(cornerBuf);
        worldPoints.push_back(worldBuf);
    }
\\ Kalibrace kamery
calibrateCamera(worldPoints, corners, grey.size(),
   camMtx, distCoeffs, rvecs, tvecs);
\\ Uložení parametrů kamery do souboru
FileStorage fs(savePath, FileStorage::WRITE);
fs << "cameraMatrix" << camMtx;
fs << "distCoeffs" << distCoeffs;
...
```

3.3 Detekování pohybu pohyblivou kamerou

Základním stavebním kamenem celého algoritmu bude metoda optického toku popsaná v kapitole 2.1.2, kterou lze použít i při pohybu kamery. Výsledkem tohoto algoritmu je jedna skupina bodů, ve které jsou body, jejichž pohyb byl způsoben pohybem kamery a body, které se opravdu pohybují. Tyto body je třeba rozdělit. Dále budeme pracovat jen se druhou skupinou. Jelikož sledovaných bodů je stále velké množství, budou body seskupeny do skupin

dle objektů, na kterých se nacházejí. Tato skupina bude nahrazena jedním reprezentativním prvkem. Posledním bodem detekce je kontinuální sledování a vyhodnocování pohybu za účelem eliminace falešných detekcí. Z těchto důvodů bude algoritmus rozdělen na následující čtyři části:

- Generování bodů podezřelých z pohybu (kapitola 3.3.1)
- Separace bodů dle původu pohybu (kapitola 3.3.2)
- Rozdělení bodů do skupin (kapitola 3.3.3)
- Kontinuální sledování (kapitola 3.3.4)

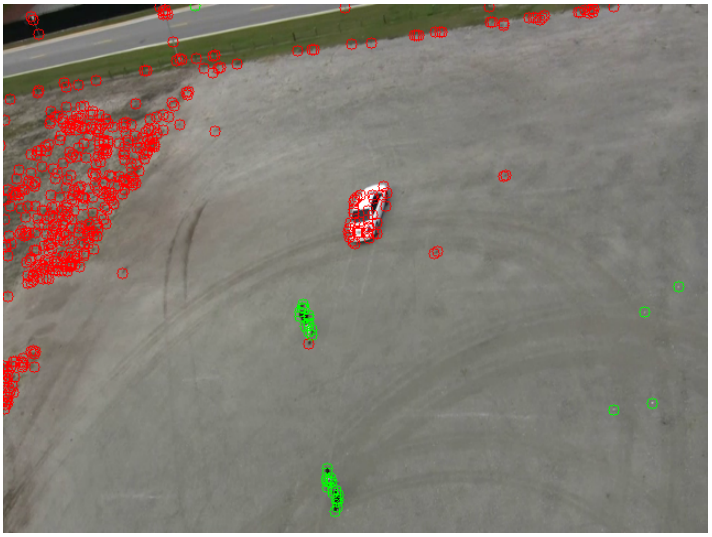
Tyto části budou řešeny odděleně. Podrobnější popis jednotlivých částí bude rozebrán v následujících kapitolách.

3.3.1 Generování bodů podezřelých z pohybu

Generování bodů bude provedeno s využitím optického toku. V OpenCV je tato metoda implementována ve funkci *cv:calcOpticalFlowLK*, jejichž vstupem jsou 2 obrázky a souřadnice bodů, které chceme sledovat. Výstupem této funkce jsou souřadnice posunutých vstupních bodů. Ne všechny body jsou vhodné ke sledování. Některé body se sledují lépe např. rohy. Ke zjištění dobře sledovatelných bodů lze využít funkci *cv::goodFeaturesToTrack*, která nám tyto body v daném obrázku poskytne. Tato funkce hledá rohy v daném obrázku metodou minimálních vlastních čísel gradientní matice [11].

3.3.2 Separace bodů dle původu pohybu

Výsledkem předchozího kroku je jedna skupina pohybujících se bodů, která zahrnuje pohyb objektů i pohyb způsoben kamerou. Pro další práci je třeba tyto skupiny oddělit. Budeme předpokládat, že množství bodů, jejichž pohyb je způsoben kamerou je značně větší, než počet bodů, které se opravdu pohybují. Separace bodů bude provedena na základě hledání majoritního pohybu. Budeme vycházet z publikace [12]. Pohyb bodů $[x, y]$ v čase $t-1$ do bodů v čase t lze aproximovat transformací, reprezentovanou maticí H ve tvaru:



Obrázek 3.2: Rozdělení bodů na dvě skupiny

$$\begin{bmatrix} \frac{x_{odhad}(t)}{\lambda} \\ \frac{y_{odhad}(t)}{\lambda} \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x(t-1) \\ y(t-1) \\ 1 \end{bmatrix} \tag{3.1}$$

kde $\lambda = 1/(h_{20}.x_{odhad}(t) + h_{21}.y_{odhad}(t) + h_{22})$. Tato transformace byla odhadnuta algoritmem RANSAC [14]. Vstupem tohoto algoritmu jsou dvě skupiny bodů, mezi kterými je třeba najít transformaci. Výstupem je matice H. Chybu této transformace odhadu lze definovat jako:

$$E(p(t-1), p_{odhad}(t), H) = ||p(t-1) - p_{odhad}(t)|| \tag{3.2}$$

Ideálně bude tato chyba nulová. Pokud bude chyba odhadu větší než E_{tresh}, lze přepokládat, že pohyb nebyl způsoben kamerou. Na základě experimentů byla stanovena hodnota, která je vhodná pro většinu videí $E_{tresh} = 0,7$. Na obrázku 3.2 je zobrazena ukázka separace bodů na dvě skupiny. Červené body reprezentují pohyb způsobený kamerou, zelené body reprezentují pohybující se objekty.

3.3.3 Rozdělení bodů do skupin

Z důvodu zjednodušení sledování bodů je třeba zredukovat jejich velký počet. Body budou sdruženy do skupin dle příslušných objektů a celá skupina bude nahrazena jedním reprezentativním bodem tzv. pivotem. Bude využito modifikované verze algoritmu *Mean-Shift Clustering*.

Definice 3.1. Nechť N je počet skupin a N_i je počet bodů v i-té skupině. Prvky i-té skupiny označíme x_k^i, kde $k = 1, \ldots, N_i$. Pak je pivot i-té skupiny $p_i = \frac{1}{N_i} \sum_{k=1}^{N_i} x_k^i$.

Dále celkový počet bodů označíme P. Všechny body jsou rozděleny do skupin s využitím pravidel 3.3 a 3.4.

$$\max_{j=1}^{N_i} ||x_j^i - p_i|| < R_{tresh} \quad i = 1, .., N \tag{3.3}$$

$$\min_{j=1}^{N} (||x_k^i - p_j||) = ||x_k^i - p_i|| \quad k = 1, .., P \tag{3.4}$$

Konstantu R_{tresh} je třeba zvolit na základě velikosti sledovaných objektů. Za předpokladu, že sledující objekty jsou malé vzhledem k velikosti obrazu, bylo zvoleno $R_{tresh} = 50$. Z rovnice 3.3 plyne, že maximální vzdálenost bodu x_j^i od pivota skupiny p_i je shora omezena R_{tresh}. Tímto omezíme maximální velikost skupiny. Rovnice 3.4 nám zajišťuje, že body leží v optimální skupině tzn. že vzdálenost bodu x_j^i od pivotu své skupiny p_i je minimální.

3.3.4 Kontinuální sledovaní

K omezení falešných detekcích je třeba kontinuálně sledovat výskyty sledovaných bodů a vyhodnotit, zda se objekty skutečně pohnuly. V kapitole 3.3.3 jsme zredukovali počet bodů, které je nutné sledovat, jen na pivoty, kterých je podobný počet jako pohybujících se objektů. K tomuto účelu byly vyzkoušeny dva způsoby - sledování bodů v určené oblasti (mřížce) a přímé sledování pivotů (objektů).

Sledování pomocí mřížky

Prvním ze způsobů, který byl aplikován na kontinuální sledovaní bodů, je rozdělení obrázku mřížkou na menší obdélníky malé velikosti. V těchto obdélnících byl počítán počet výskytů pivotů během sledování. Pokud byl pivot detekován více než G_{tresh} krát po sobě jdoucích snímcích, byl pivot klasifikován jako pohybující se a byl vykreslen. Experimentálně bylo stanoveno $G_{tresh} = 3$. Největším problémem byl přechod pivotů do sousedních obdélníků. Pokud byl pohyb příliš rychlý, bod nebyl v obdélníku více než G_{tresh} krát a pohyb nebyl vyhodnocen. Tento problém lze částečně řešit kontrolováním sousedních obdélníků, nicméně tato metoda nedávala dobré výsledky. Proto bylo přistoupeno k následující metodě.

Přímé sledování pivotů

Problém přechodu pivotů mezi obdélníky byl vyřešen přímým sledováním pivotů. Bude počítána délka souvislého výskytu pivotů v po sobě jdoucích snímcích. Jednotlivé pivoty mezi snímky bylo nejdříve nutné spárovat. Označíme pivoty ve snímcích postupně $p_i(t)$ a $p_i(t-1)$. Bylo využito transformace 3.1 ke zmenšení vlivu pohybu kamery. Pivoty $p_i(t)$ byly transformovány na body $p'_i(t) = Hp_i(t)$. Poté byly pivoty spárovány s využitím relace 3.5 (znak $\sim$ značí přiřazení):

$$p_i(t) \sim p_j(t-1) <=> |p_i(t) - p'_j(t-1| = \min_{k=1}^{N} |p_i(t) - p'_k(t-1)| < D_{tresh} \quad (3.5)$$

Relace 3.5 nám říká, že pivoty jsou spárovány, pokud je jejich vzdálenost nejmenší a zároveň menší než D_{tresh}. Konstanta D_{tresh} omezuje maximální změnu, o kterou se může pivot pohnout. Experimentálně bylo stanoveno $D_{tresh} = 20$. Následně bude počítán počet výskytů jednotlivých pivotů ve snímcích. Pokud je pivot ztracen, počet jeho výskytů je resetován. Pokud je pivot detekován více než C_{tresh} krát, je vyhodnocen pohyb. Experimentálně bylo stanoveno $C_{tresh} = 3$.

3.3.5 Přehled parametrů algoritmu

V závěru této kapitoly bude uveden přehled všech parametrů algoritmu, jejich význam a vliv změny na funkčnost algoritmu. Optimální parametry byly otestovány na videosekvenci s rozlišením 640x480 pixelů a se snímkovou frekvencí 30 snímků za sekundu.

E_{tresh}

Význam: Minimální chyba transformace H (viz rovnice 3.1) k separaci bodů
Optimální hodnota: 0.7
Vliv zmenšení: Zmenšení rychlosti objektů, kterou algoritmus detekuje; více falešných detekcí
Vliv zvětšení: Zvětšení rychlosti objektů, kterou algoritmus detekuje; méně falešných detekcí

R_{tresh}

Význam: Velikost detekovaných objektů (viz rovnice 3.3)
Optimální hodnota: 50 px
Vliv zmenšení: Zmenšení velikosti detekujících objektů
Vliv zvětšení: Zvětšení velikosti detekujících objektů

G_{tresh}

Význam: Minimální počet po sobě jdoucích detekcí pivotů v mřížce nutných k vyhodnocení pohybu (viz kapitola 3.3.4)
Optimální hodnota: 3
Vliv zmenšení: Zmenšení zpoždění algoritmu; více falešných detekcí
Vliv zvětšení: Zvětšení zpoždění algoritmu; méně falešných detekcí

D_{tresh}

Význam: Maximální rychlost objektů, která je detekována (viz rovnice 3.5)
Optimální hodnota: 20 $\frac{px}{frame}$
Vliv zmenšení: Zmenšení rychlosti objektů, která je detekována; zmenšení falešných detekcí
Vliv zvětšení: Zvětšení rychlosti objektů, která je detekována; zvětšení falešných detekcí

C_{tresh}

Význam: Minimální počet po sobě jdoucích detekcí pivotů nutných k vyhodnocení pohybu (metoda přímého sledovaní pivotů, viz kapitola 3.3.4)
Optimální hodnota: 3
Vliv zmenšení: Zmenšení zpoždění algoritmu; více falešných detekcí
Vliv zvětšení: Zvětšení zpoždění algoritmu; méně falešných detekcí

Kapitola 4

Výsledky

V závěru této práce budou navrhnuty a provedeny experimenty, které ověří úspěšnost algoritmu a určí podmínky, za kterých pracuje spolehlivě. K testu budou použity optimální parametry, které jsou uvedeny v kapitole 3.3.5.

4.1 Vliv rychlosti pohybu objektů

V tomto experimentu bude zkoumán vliv rychlosti objektů na funkčnost algoritmu. K testu bude použito uměle generované video s využití následující funkce v Matlabu. Videa byla generována v rozlišení 640x480 px a s frekvencí 30 snímků za sekundu.

:

```
1  % Creates speed test for movement tracking algorithm
2  % inputs: speed - objects speed
3  %         background - path to backround image
4  %         fileName - path, where created video is stored
5
6  function createSpeedTest(speed, background, fileName)
7      close all;
8      video = VideoWriter(fileName, 'Uncompressed␣AVI');
9      open(video);
10     img = imrotate(imread(background), 180);
```

```
11      pos = 10;
12      set(gca, 'visible', 'off');
13      for k=1:(400/speed)
14          hold on;
15          image(img);
16          plot(100, pos, 'xr', 'MarkerSize', 20,
                'LineWidth', 5);
17          hold off;
18          pos = pos + speed;
19          writeVideo(video, getframe(gcf));
20      end
21      close(video);
22      close all;
23  end
```

Funkce vytvoří video se zvolenou rychlostí pohybu objektů (v $\frac{px}{frame}$). Jeho pozadí tvoří statický obrázek a jako pohybující se objekt je zvolen kříž. Ukázka tohoto videa je na obrázku 4.1. Toto video je poté vyzkoušeno algoritmem a je vyhodnocen poměr počtu detekcí a celkového počtu snímků. Výsledky jsou uvedeny v tabulce 4.1.

rychlost $[\frac{px}{frame}]$	úspěšnost [%]
1	34.8
2	80.5
3	79.5
4	77.0
5	71.3
6	66.0
7	57.8
8	56.0
15	37.5

Tabulka 4.1: Výsledky testu vlivu rychlosti objektů na funkčnost algoritmu

Z tabulky je patrné, že nejlépe algoritmus detekuje rychlosti od 2 $\frac{px}{frame}$ do 4 $\frac{px}{frame}$, kde jsou výsledky téměř srovnatelné. Dolní mez souvisí s parametrem E_{tresh}. Horní mez je ovlivněna parametrem D_{tresh}, ale také se snímkovou frekvencí videa, jejíž vliv bude zkoumán v kapitole 4.2.

Obrázek 4.1: Ukázka testu vlivu rychlosti objektů

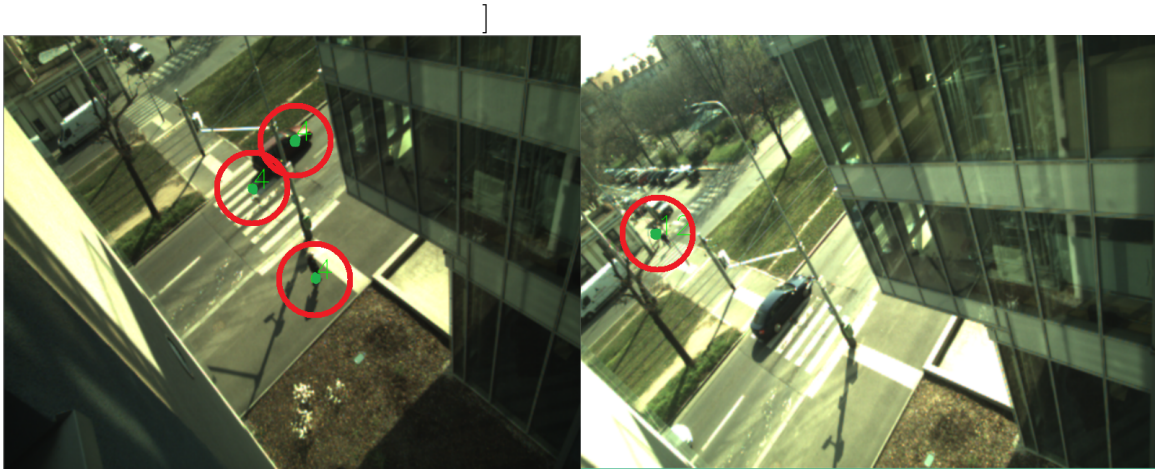
4.2 Vliv snímkové frekvence videa

Optický tok (kapitola 2.1.2), který je stavebním kamenem algoritmu určeného k detekci pohybu, pracuje správně za předpokladu, že změny mezi jednotlivými snímky nejsou příliš velké. To souvisí se snímkovou frekvencí videa. V tomto testu bude zjištěna minimální snímková frekvence, při které algoritmus ještě pracuje dostatečně spolehlivě. K testu budou použity sekvence snímků ulice, na které se pohybují auta a chodci. Ze sekvence budou postupně vyřazovány jednotlivé snímky až do doby, kdy algoritmus přestane pracovat správně. Výsledky byly porovnány s referenčním videem se 24 fps. Závěry experimentu jsou zobrazeny v tabulce 4.2.

FPS	Typy detekovaných objekty
12	chodci, auta
8	chodci, některá auta
5	chodci
4	někteří chodci
2	žádné

Tabulka 4.2: Výsledky testu vlivu fps na funkčnost algoritmu

Z tabulky je patrné, že algoritmus pracuje správně, pokud je snímková frekvence videa alespoň 12 fps. Pokud je frekvence nižší, začínají se nejdříve ztrácet detekce rychleji se pohybujících objektů (8 fps) a poté i pomalejších



Obrázek 4.2: Ukázka testu vlivu fps videa (Vlevo 12 fps, vpravo 8 fps)

objektů (4 fps). Algoritmus nedetekuje žádné objekty, pokud je snímková frekvence videa menší než 3 fps. Tento jev je pozorovatelný na obrázku 4.2. U videa s 12 fps je detekováno auto i chodci, u videa s 8 fps jen chodec.

4.3 Úspěšnost algoritmu

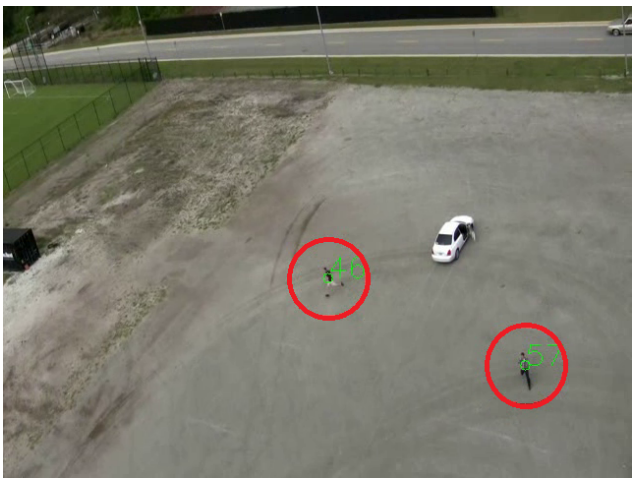
Posledním experimentem, který bude proveden, je zjištění pravděpodobnosti detekce pohybu. K tomuto účelu bude použit UCF Aerial Action Data Set [13]. Data set tvoří videa s auty a chodci zachycené z výšky. K experimentu budou použity 2 videosekvence o délce 15 s, rozlišením 640x480 px a se snímkovou frekvencí 24 fps. Videosekvence přibližně splňují podmínky, za kterých algoritmus pracuje optimálně (viz kapitoly 4.1 a 4.2). K ověření funkčnosti algoritmu je nejdříve nutné pohyb ve videu ručně označit. S těmito daty bude následně porovnán výstup algoritmu.

		Detekce algoritmu		
Číslo testu	Celkem detekcí	Správně	Falešně negativní	Falešně pozitivní
1	1200	699	501	23
2	901	638	263	61

Tabulka 4.3: Výsledky testu úspěšnosti algoritmu

K ručnímu označení dat byl vytvořen nástroj v Matlabu. V tomto nástroji je možné procházet jednotlivé snímky videa a označit pohybující se objekty.

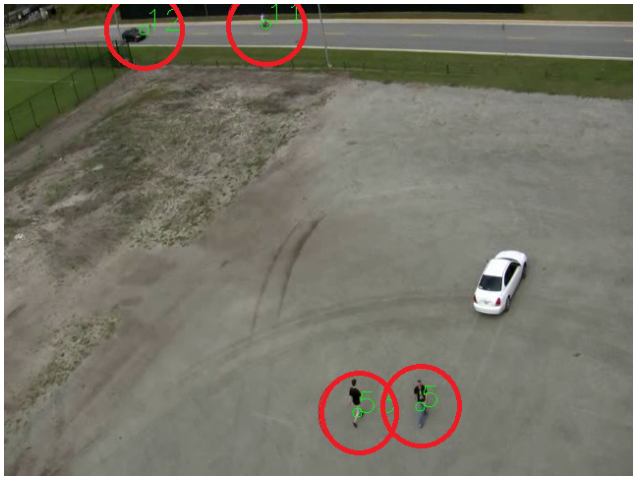
Pro zjednodušení je ručně označen jen každý pátý snímek videa. Pohyb mezi těmito snímky bude označen lineární interpolací. Detekce byla vyhodnocována pro každý snímek zvlášť. Výsledky experimentů jsou uvedeny v tabulce 4.3. V tabulce je uveden celkový (ručně označený) počet detekcí a počet, který označil algoritmus. Dále jsou zde uvedeny dva druhy chyb - falešně pozitivní a falešně negativní detekce. Falešně pozitivní detekce označil algoritmus jako pohybující se objekty, i když se ve skutečnosti nepohybovaly. Falešně negativní detekce je druh chyby, kdy metoda nenašla pohybující se objekt. Ukázky testu 1. sekvence jsou zobrazeny na obrázcích 4.3, 4.4, 4.5, ukázky testu 2. sekvence na obrázcích 4.6, 4.7, 4.8.



Obrázek 4.3: Ukázka testu úspěšnosti algoritmu (sekvence č. 1, snímek č. 1)

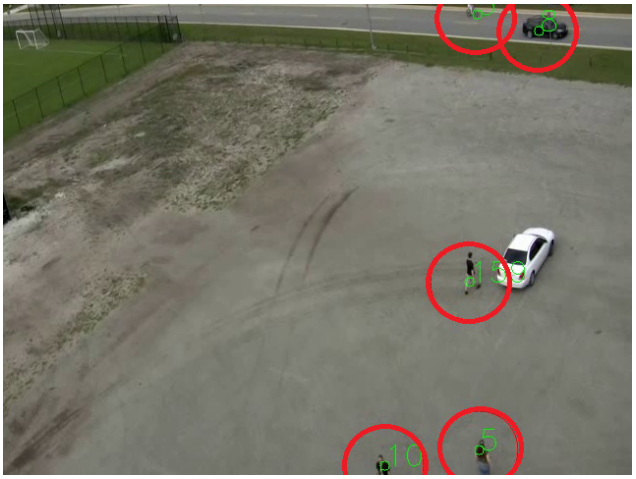
Obrázek 4.4: Ukázka testu úspěšnosti algoritmu (sekvence č. 1, snímek č. 2)

Obrázek 4.5: Ukázka testu úspěšnosti algoritmu (sekvence č. 1, snímek č. 3)



Obrázek 4.6: Ukázka testu úspěšnosti algoritmu (sekvence č. 2, snímek č. 1)

Obrázek 4.7: Ukázka testu úspěšnosti algoritmu (sekvence č. 2, snímek č. 2)



Obrázek 4.8: Ukázka testu úspěšnosti algoritmu (sekvence č. 2, snímek č. 3)

V 1. sekvenci detekoval algoritmus správně pohybující se objekt v 58 % všech případů, v druhé sekvenci 71 % případu. Množství falešně pozitivních detekcí je vzhledem k velkému počtu detekcí zanedbatelné. Je také nutné zmínit, že kvůli kontinuálnímu vyhodnocování pohybu (kapitola 3.3.4) dochází při detekci ke zpoždění - objekt není při vstupu do scény detekován okamžitě. Tato skutečnost snižuje úspěšnost algoritmu v navrženém testu.

Kapitola 5

Závěr

Na základě analýzy metod, které jsou schopny detekovat pohybující se objekty v obraze z pohyblivé kamery byl vytvořen algoritmus, který je schopen tento problém řešit. K jeho implementaci byla vybrána volně dostupná knihovna OpenCV a jako programovací jazyk byl zvolen C++. Celý algoritmus byl rozdělen na několik částí, které usnadňují jeho implementaci.

Metoda k detekování pohybu se skládá ze čtyř částí - generování bodů podezřelých z pohybu optickým tokem, separace bodů dle původu jejich pohybu, rozdělení bodů do skupin a kontinuální sledovaní.

Algoritmus je schopen zpracovat video v reálném čase, vytvořená aplikace je funkční a detekuje většinu objektů. K ověření funkčnosti metody a stanovení podmínek, za kterých pracuje spolehlivě bylo navrhnuto a provedeno několik experimentů. Algoritmus nejlépe detekuje objekty pohybující se rychlostmi mezi 2 $\frac{px}{frame}$ a 4 $\frac{px}{frame}$. V reálných podmínkách je nutné, aby zdrojová videosekvence měla snímkovou frekvenci alespoň 12 snímků za sekundu. Algoritmus je schopen detekovat pohybující se objekty s pravděpodobností asi $P \approx 64\%$ a chybou asi $F \approx 4\%$. Za chyby jsou zde považovány falešně pozitivní detekce. Aplikace funguje lépe pokud je video kvalitní, pohybující se objekty dostatečně velké a pohyb kamery je pomalý a plynulý.

Příloha A

Literatura

[1] Camera calibration with OpenCV. [Online], `https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, and B K. U. Leuven. Speeded-up robust features (surf), 2008.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Zhaowei Cai, Quanfu Fan, Rogério Schmidt Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. *CoRR*, abs/1607.07155, 2016.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[6] Boyoon Jung and Gaurav Sukhatme. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. pages 980–987, 04 0002.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[8] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[9] M Piccardi. Background subtraction techniques: A review. volume 4, pages 3099 – 3104 vol.4, 11 2004.

[10] Paul Rosin and Tim Ellis. Image difference threshold strategies and shadow detection. pages 347–356, 02 1995.

[11] Jianbo Shi and Carlo Tomasi. Good features to track. pages 593–600, 1994.

[12] Ruxandra Tapu, Mocanu Bogdan, Andrei Bursuc, and Zaharia Titus. A Smartphone-Based Obstacle Detection and Classification System for Assisting Visually Impaired People. *ICCV 2013 Workshops*, pages 444–451, December 2013.

[13] Center for Research in Computer Vision, University of Central Florida. Ucf aerial action data set. [Online], `https://www.crcv.ucf.edu/data/UCF_Aerial_Action.php`.

[14] Toshihiko Watanabe, Takeshi Kamai, and Tomoki Ishimaru. Robust estimation for camera homography by fuzzy ransac algorithm with reinforcement learning. *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 712–717, 2014.

[15] Wikipedia contributors. Optical flow, 2019. [Online], `https://en.wikipedia.org/wiki/Optical_flow`.

[16] Karel Zimmermann. Camera model and calibration. [Online], `https://cw.fel.cvut.cz/b181/_media/courses/b3b33vir/camera.pdf`, 2019.

Příloha B

Obsah DVD

Přiložené DVD obsahuje zdrojové kódy práce pro systém LaTeX , text bakalářské práce, zdrojové kódy programu v jazyce C++ a Matlabovské skripty určené k testování programu. Součástí jsou také použitá testovací videa. Adresářová struktura DVD je zobrazena v tabulce B.1.

Adresář	obsah
\doc	zdrojové kódy práce
\src	zdrojové kódy aplikace
\scripts	testovací skripty
\videos	testovací videa

Tabulka B.1: Adresářová struktura DVD

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tefr** Jméno: **Filip** Osobní číslo: **465865**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Detekce pohybujících se objektů v obraze z pohyblivé kamery

Název bakalářské práce anglicky:

Moving Object Detection in an Image from a Moving Camera

Pokyny pro vypracování:

1. Prostudujte problematiku počítačového zpracování obrazu se zaměřením na nalezení korespondence mezi obrázky pořízenými z různých míst a na detekci změn v následujících obrazech z kamery.
2. Proveďte rešerši metod pro detekci pohybu v sekvenci obrazů z kamery se zaměřením na metody vhodné pro případ, kdy se kamera sama pohybuje.
3. Na základě vyhodnocení dostupných metod a výsledků projektů na jiných pracovištích navrhněte metodu detekce změn v obraze vhodnou pro detekci objektů. Přijměte potřebná zjednodušující opatření s ohledem na řešitelnost projektu, jako omezení rychlosti pohybu kamery. Metodu optimalizujte pro situaci, kdy je velikost objektů v obraze relativně malá, např. je-li kamera instalována na letící helikoptéře.
4. Navrženou metodu implementujte s využitím dostupných knihoven pro zpracování obrazu.
5. Funkci implementované metody experimentálně ověřte na datech získaných z reálné kamery a vyhodnoťte její přesnost a schopnost detekce různých objektů.

Seznam doporučené literatury:

[1] Qian Yu and Gérard Medioni, “A GPU-based implementation of Motion Detection from a Moving Platform”, in IEEE workshop on Computer Vision on GPU, in conjunction with CVPR’08.
[2] Rodríguez-Canosa, G.R.; Thomas, S.; del Cerro, J.; Barrientos, A.; MacDonald, B. A Real-Time Method to Detect and Track Moving Objects (DATMO) from Unmanned Aerial Vehicles (UAVs) Using a Single Camera. Remote Sens. 2012, 4, 1090-1111.
[3] A. Talukder and L. Matthies, "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow," 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, 2004, pp. 3718-3725 vol.4.
[4] Thompson, William B. and Ting-Chuen Pong. “Detecting moving objects.” International Journal of Computer Vision 4 (1990): 39-57.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Chudoba, inteligentní a mobilní robotika CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **24.01.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Jan Chudoba	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

