Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2013 **MARTIN FAJFR**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin FAJFR**
Osobní číslo:	**A10N0134P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Aplikace pro automatickou detekci meteorů v souboru astrofotografií velkého rozsahu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Seznamte se s doporučenou literaturou a metodami strojového vidění pro zpracování obrazových dat.
2. Seznamte se s knihovnou OpenCV, jejím užitím a specifiky.
3. Seznamte se s knihovnou Qt, jejím užitím a specifiky.
4. Seznamte se se vstupními daty, vlastnostmi astrofotografií a objektů pro detekci.
5. Implementujte automatickou detekci meteorů s využitím knihovny OpenCV a vytvořte GUI pro obsluhu programu v Qt.
6. Proveďte vyhodnocení úspěšnosti na testovacím souboru dat.

Rozsah grafických prací:	**dle potřeby**
Rozsah pracovní zprávy:	**35-50 stránek A4**
Forma zpracování diplomové práce:	**tištěná**

Seznam odborné literatury:

Šonka, Hlaváč, Boyle: Image Processing, Analysis and Machine Vision
CENGAGE Engineering; 3 edition, 2007
ISBN 049508252X
Shapiro, Stockman: Computer Vision
Prentice Hall, 2001
ISBN 0-13-030796-3

Vedoucí diplomové práce:	**Ing. Ivan Pirner** Katedra kybernetiky
Datum zadání diplomové práce:	**24. září 2012**
Termín odevzdání diplomové práce:	**17. května 2013**

Doc. Ing. František Vávra, CSc.
děkan

L.S.

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 24. září 2012

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne

.....

vlastnoruční podpis

P O D Ě K O V Á N Í

Děkuji Ing. Ivanu Pirnerovi za vedení této diplomové práce a za jeho pomoc při řešení odborných problémů. Dále děkuji za podpoření této práce grantovým projektem SGS-2013-032.

Abstrakt

Cílem této práce je vytvořit aplikaci, která bude nasazena v Astronomickém ústavu AV ČR v Ondřejově, kde bude využita pro automatickou detekci meteorů na snímcích noční oblohy. Tato aplikace má za úkol usnadnit práci pracovníkům akademického ústavu, kteří nyní prohlížejí každý snímek zvlášť a hledají zachycené meteory. Součástí práce je úvod do reprezentace obrazových dat výpočetní technikou a seznámení s některými metodami počítačového vidění. Hlavní část práce je věnována návrhu metody pro hledání meteorů v obraze s použitím metod počítačového vidění. Tato metoda je programově implementována v uživatelské aplikaci a následně testována na množině dat velkého rozsahu. Aplikaci mají k dispozici pro použití a testování pracovníci akademického ústavu. Na závěr práce je uvedena úspěšnost navržené metody a srovnání s již existující metodou pro hledání čar v obraze.

Klíčová slova

Meteor, Počítačové vidění, Houghova transformace, Optický tok, OpenCV, Qt

Abstract

The goal of this work is to create an application that will be used at the Astronomical Institute of the Academy of sciences of the Czech Republic in Ondřejov, where it will be used for an automatic detection of meteors on the photos of the night sky. The goal of this application is to facilitate the work of Institute employees, because up until now the employees had to look for the meteors in each photo separately. A part of this work is an introduction to the representation of the pictures in computer science. Next part is an introduction to some of the methods of computer vision. The main part is focused on creation of a method for finding meteors in the image using the computer vision techniques. This method is implemented in a user application and it is tested on large astrophotography dataset. The application is available to the Institute workers and currently is being tested by them. The success rate of this new method is provided at the end of this work and the new method is compared with already existing method for finding lines in the image.

Key words

Meteor, Computer vision, Hough transform, Optical Flow, OpenCV, Qt

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Počítačové vidění je disciplína, která se zabývá získáváním informace ze zachyceného obrazu s využitím počítače. S postupným vyvíjením výpočetní techniky a rostoucí dostupností záznamových zařízení, jako jsou videokamery a fotoaparáty s vysokým rozlišením, se tato disciplína stále více rozšiřuje do různých odvětví průmyslu i do každodenního života.

Příkladů aplikací počítačového vidění existuje velké množství. Ty nejznámější jsou například analýza lékařských dat, rozpoznávání tváře, rozpoznávání dopravních značek, sledování osob a pohybujících se objektů, kontrola průmyslové výroby, atd.

Tato práce se zabývá aplikací počítačového vidění pro vyhledání meteoru na snímcích noční oblohy. Zadání této úlohy bylo poskytnuto od Astronomického ústavu AV ČR v Ondřejově, kde probíhá základní výzkum meteorů. Ze zaznamenaných meteorů dopočítávají například jejich dráhu, rychlost a některé jeho fyzikální vlastnosti. Ve výjimečných případech odhadují dokonce místo dopadu jeho zbytků. V současné době vyhodnocují zaznamenané snímky pracovníci „ručně“ nebo pomocí speciálního vyhodnocovacího softwaru. Nejprve však musí mít informaci, na kterých snímcích byl meteor zachycen. Každý snímek tedy prohlížejí, a hledají, zdali na něm byl zachycen meteor. Vyvinutí metody, jež by dokázala meteory najít automaticky, by jim značně ulehčila práci.

První část práce je věnovaná reprezentaci obrazových dat v počítači. Součástí toho je popis a analýza poskytnutých dat od astronomického ústavu. Dále jsou uvedeny jednotlivé metody počítačového vidění, které budou použity pro návrh metody automatického hledání meteorů. Hlavní zaměření práce je navrhnout tuto metodu pomocí popsaných metod počítačového vidění a vytvořit uživatelskou aplikaci, ve které bude tato metoda implementována.

Součástí práce je i popis použitého programovacího jazyku a knihoven funkcí pro počítačové vidění a pro vytvoření uživatelského rozhraní. Protože tyto funkce mají několik volitelných parametrů, bude volba jednotlivých parametrů diskutována. Nakonec je uvedena úspěšnost navržené metody.

2 Obrazová data

Tato kapitola se zabývá reprezentací obrazu v počítači. Bude zde vymezeno několik základních pojmů týkajících se obrazových dat a bude proveden stručný úvod do používaných barevných modelů. Dále se budeme zabývat popisem a analýzou dat poskytnutých od pracovníků z Astronomického ústavu AV ČR v Ondřejově. K datům bylo dodáno i jejich částečné vyhodnocení, což jsou informace o tom, na jakých snímcích a v jaké části oblohy byl zachycen meteor.

2.1 Reprezentace obrazu

Ve zpracování obrazu je zvykem pracovat s konceptem analogového i digitálního obrazu. Obraz můžeme popsat matematickým modelem, který je vhodné brát jako funkci dvou proměnných. Operace s obrazem jsou pak definovány jako operace se spojitou funkcí. Digitální obraz je reprezentován dvourozměrným obdélníkovým polem diskrétních hodnot. Obrazový prostor a intenzitní rozsah jsou kvantizovány do množin diskrétních hodnot. To nám dovoluje obraz uložit do dvourozměrné struktury počítačové paměti. Obvykle se ukládá intenzita v rozsahu 8 bitů (1 byte), což jsou celá čísla od 0 do 255. U barevných obrazů jsou v každém prvku dvourozměrné struktury uloženy 3 hodnoty intenzity pro jednotlivé barvy. [1]

Podle [1] zavedeme definice, které budou následně využívány v celé této práci.

Definice 2.1: Digitální obraz je dvourozměrný obraz **I**[*r*, *c*] reprezentován dvourozměrným polem (maticí) diskrétních hodnot vzorků intenzit s omezenou přesností. Proměnná *r* označuje řádek v obrazu (z anglického *row*) a proměnná *c* označuje sloupec (z anglického *column*). Počet řádků v obraze budeme označovat *R* a počet sloupců *C*.

Poznámka: Jeden bod v digitálním obraze je zvykem nazývat pixel.

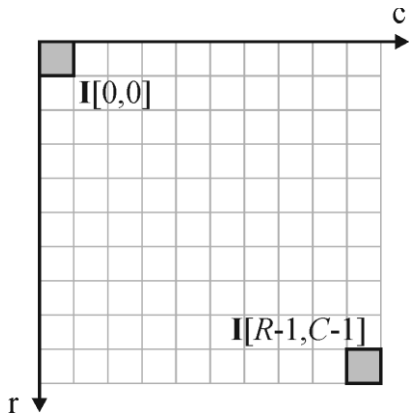
Definice 2.2: Obrazová funkce *f* (*x*, *y*) je matematická reprezentace obrazu pomocí funkce dvou proměnných. Proměnné *x* a *y* jsou reálná čísla definující body v obraze a *f* (*x*, *y*) je reálné číslo definující intenzitu obrazu v bodě (*x*, *y*).

Definice 2.3: Šedotónový obraz $\mathbf{I_g}$ je osmibitový digitální obraz s jednou hodnotou intenzity pro každý pixel. Hodnoty intenzit jsou celá kladná čísla v rozsahu $\langle 0;\ 255 \rangle$.

Definice 2.4: Binární obraz je jednobitový digitální obraz, ve kterém nabývají jednotlivé pixely hodnot 0 nebo 1. V této práci pod označením binární obraz budeme označovat osmibitový obraz, jehož pixely nabývají dvou hodnot, 0 (černá) nebo 255 (bílá) a označíme ho $\mathbf{I_b}$.

Definice 2.5: Barevný obraz $\mathbf{I_c}$ je dvourozměrný obraz, který má pro každý pixel vektor tří intenzitních hodnot pro jednotlivé barvy modelu RGB (tento model je popsán v kapitole 2.2).

Na následujícím obrázku je ukázáno, jaký souřadný systém se používá pro přístup k jednotlivým pixelům v počítači.



Obrázek č. 2.1: Souřadný systém pro digitální obraz používaný v počítači.

Definice 2.6: Čtyř-okolí aktuálního bodu **I**[*r*, *c*] je množina čtyř bodů v těsné blízkosti tohoto bodu tj. body **I**[*r*+1, c], **I**[r-1, c], **I**[r, c+1], **I**[r, c-1].

Definice 2.7: Osmi-okolí aktuálního bodu **I**[*r*, *c*] je množina osmi bodů v těsné blízkosti tohoto bodu tj. body **I**[*r*+1, *c*], **I**[*r*-1, *c*], **I**[*r*, *c*+1], **I**[*r*, *c*-1], **I**[*r*+1, *c*+1], **I**[*r*+1, *c*-1], **I**[*r*-1, *c*+1], **I**[*r*-1, *c*-1].

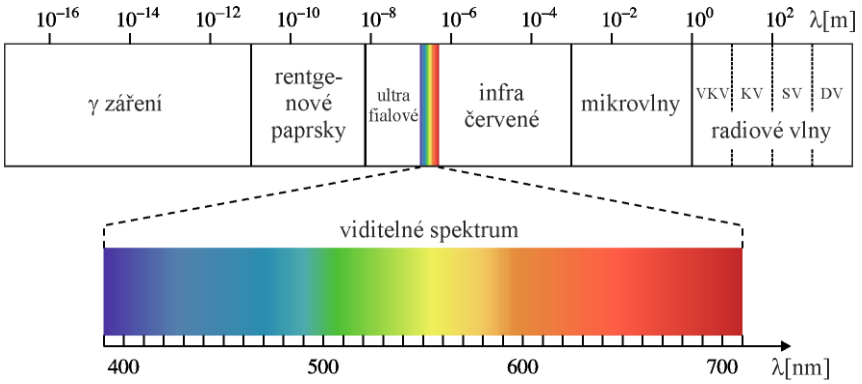
(a) (b)

Obrázek č. 2.2: Okolí aktuálního bodu. (a) Čtyř-okolí aktuálního bodu. (b) Osmi-okolí aktuálního bodu. (aktuální bod označen křížkem)

2.2 Barva v obraze

Barevný obraz byl definován v předchozím odstavci v definici 2.5. Pro barevný obraz $\mathbf{I_c}$ je každý pixel reprezentován třemi hodnotami (červená, zelená, modrá). Jednotlivé prvky obrazu (dvourozměrného pole) budou tedy obsahovat uspořádané trojice. Prvky této trojice jsou v rozsahu 8 bitů, budou tedy nabývat hodnot 0 až 255.

Vnímání barev člověkem přidává další vlastnost k základním fyzikálním vlastnostem objektu tj. vlnovou délku elektromagnetického záření. Pro lidské oko je viditelná pouze malá část tohoto záření. Konkrétně je to záření o vlnové délce λ v rozmezí 380nm až 740nm. Přibližné rozložení elektromagnetického spektra je zobrazeno na následujícím obrázku. [2]

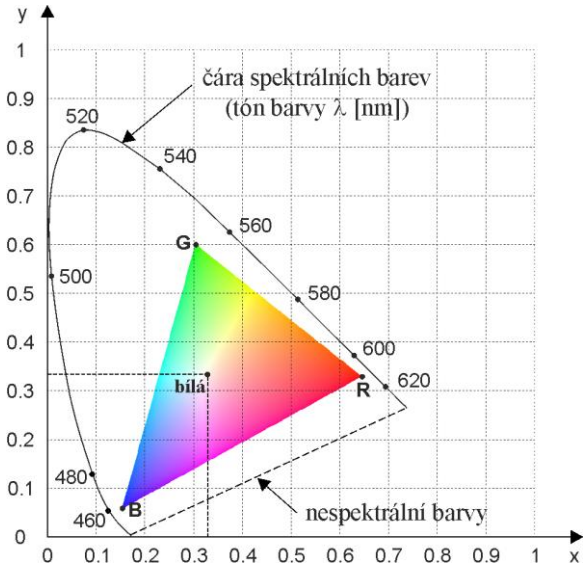


Obrázek č. 2.3: Rozdělení elektromagnetického spektra.

Barva může být vyjádřena kombinací tří základních barev, tj. červená, zelená a modrá. Podle modelu **XYZ color space** definovaným mezinárodní komisí pro osvětlení (CIE – International Commission on Illumination) z roku 1931 jsou pro tyto barvy přiřazeny vlnové délky X=700,0nm, Y=546,1nm, Z=435,8nm. Tento model bývá označován také jako CIE standard. CIE standard je absolutní, tj. definuje jednoznačnou reprezentaci barvy, která není závislá na zařízení. [2]

Barevný model XYZ můžeme zobrazit v rovině. Jednotlivé souřadnice x,y získáme použitím následujících vztahů:

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}, \tag{2.1}$$



Obrázek č. 2.4: CIE diagram. Projekce XYZ modelu do roviny. Trojúhelník zobrazuje barvy, které zobrazuje monitor počítače.

Dále se zabývejme transformací modelu XYZ do jiných barevných modelů.

2.2.1 Model RGB

Modelu RGB existují různé reprezentace. To znamená, že dvě různá RGB zařízení mohou zobrazovat barvy různě. Jedna z transformací z modelu XYZ do RGB modelu je vyjádřena následujícím vztahem: [2]

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3{,}24 & -1{,}54 & -0{,}50 \\ -0{,}98 & 1{,}88 & 0{,}04 \\ 0{,}06 & -0{,}20 & 1{,}06 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

2.2.2 Model YIQ

Další z barevných modelů je model YIQ. Tento model je navržen tak, že složka Y popisuje intenzitu a složky I a Q reprezentují barvu. Tento model nebudeme dále popisovat, pouze využijeme složku Y pro převod barevného obrazu na šedotónový. Více o tomto barevném modelu lze najít například v [1].

Transformační vztah pro složku Y modelu YIQ z modelu RGB je vyjádřen vztahem

$$Y = 0{,}30 \cdot R + 0{,}59 \cdot G + 0{,}11 \cdot B, \tag{2.3}$$

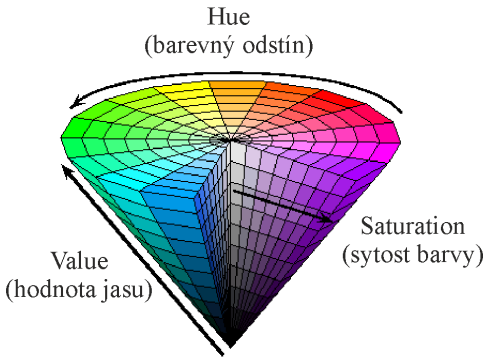
a převod barevného obrazu $\mathbf{I_c}$ na šedotónový obraz $\mathbf{I_g}$ bude tedy vyjádřen jako

$$\mathbf{I_g}[r,c] = 0{,}30 \cdot \mathbf{I_c}[r,c][0] + 0{,}59 \cdot \mathbf{I_c}[r,c][1] + 0{,}11 \cdot \mathbf{I_c}[r,c][2], \tag{2.4}$$

kde $\mathbf{I_c}[r,c][i]$ je *i*-tá složka obrazového bodu na pozici (r,c). Pro RGB obraz je tedy složka 0 intenzita červené barvy, složka 1 intenzita zelené barvy a složka 2 intenzita modré barvy.

2.2.3 Model HSV

HSV model také známý jako HSI model odděluje intenzitu, tón a sytost barvy. Toto oddělení nejvíce odpovídá vnímání barvy člověkem. Složka **H** (Hue) reprezentuje barevný odstín, složka **S** (Saturation) sytost barvy a složka **V** (Value) hodnotu jasu neboli její intenzitu.[2]



Obrázek č. 2.5: HSV model.

Převodem z RGB modelu do HSV modelu se zabývat nebudeme. Algoritmus převodu je popsán například v [1].

2.3 Poskytnutá data

Od Astronomického ústavu AV ČR v Ondřejově byla poskytnuta data pořízená v srpnu roku 2012 v období, kdy se Země na své oběžné dráze setkává s Perseidami[1]. Z tohoto období bylo zaznamenáno šest po sobě jdoucích nocí. Dále byly k dispozici data z jedné noci z roku 2011, při jejichž pořízení nebyla použita elektronická clona. Tato data pochází z institutu v Ondřejově, v Kunžaku a z expedice v Austrálii. Každou noc vzniklo přibližně 800 snímků. K testování bylo tedy dohromady použito přes 5000 snímků.

V současné době je v provozu v Ondřejově automatická bolidová kamera, na jejímž vývoji má hlavní podíl RNDr. Pavel Spurný, CSc. Tato kamera zaznamenává snímky na ploché filmy. Výhodou je výrazně vyšší rozlišení než u CCD snímačů současných digitálních fotoaparátů. Tyto snímky jsou pak digitalizovány speciálním fotogrammetrickým skenerem. Pro vyvíjenou aplikaci jsou však tyto data nevhodná.

Mimo snímků z bolidové kamery jsou v současnosti pořizovány snímky také digitální fotoaparátem Canon EOS 5D Mark II. Rozlišení jednoho snímku je 21 megapixelů, což odpovídá rozlišení 5616 x 3744 pixelů. Jednotlivé snímky jsou snímány s expozicí 30 sekund a citlivostí ISO_3200. K fotoaparátu je navíc připojena elektronická clona. Ta má za následek segmentaci rychle pohybujícího se objektu po obloze (meteoru). Takto pořízené snímky budou použity jako vstupní data pro vyvíjenou aplikaci.

2.4 Analýza snímku

Cílem této práce je najít na snímcích meteor, pokud na nich byl zachycen. Na snímané scéně se ale objevují i jiné objekty. Jsou to například hvězdy, mraky, měsíc, družice a letadla. Některé tyto objekty mohou být meteorům do jisté míry podobné. Mimo tyto objekty musíme na scéně počítat s různě jasnou oblohou, světelným znečištěním a s obzorem.

Z důvodu použití elektronické clony je nastavena vysoká citlivost fotoaparátu. Tato citlivost má za následek větší elektronický šum na snímku. Některé meteory, které

[1] Perseidy jsou kometární meteorický roj, který vznikl z komety Swift-Tuttle 1862 III.

chceme detekovat, jsou velmi slabé a na snímku je poměrně těžké je od tohoto šumu odlišit. Ukázka snímané scény je zobrazena na obrázku č. 2.5a. Na tomto snímku se nacházejí mraky, měsíc a také světelné znečištění způsobené osvětlením blízkých měst. Dále je na tomto snímku zachycen meteor, který není příliš silný.

(a) (b)

Obrázek č. 2.6: Ukázka snímku oblohy. (a) Snímaná scéna. (b) Detail zachyceného meteoru.

3 Zpracování obrazu

Tato kapitola se zabývá různými metodami počítačového vidění. Pro některé z těchto metod zavedeme novou obrazovou funkci $g(x,y)$, která představuje modifikovaný obraz použitím určitého operátoru na původní obrazovou funkci $f(x,y)$. Tyto metody budou dále využity pro návrh metody rozpoznání meteoru v kapitole 4.

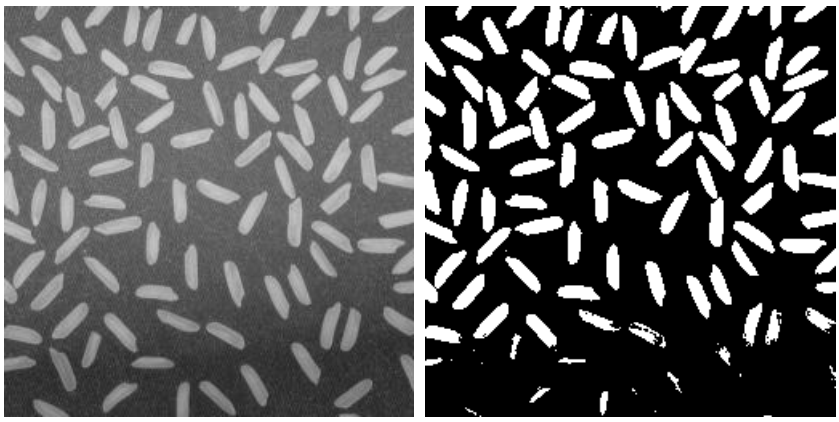
3.1 Prahování

Prahování šedotónového obrazu je nejjednodušší metoda segmentace. Mnoho objektů nebo oblastí v obraze je charakterizováno konstantním odrazem či pohlcením světla od jejich povrchu. Pro oddělení objektů od pozadí můžeme určit jasovou konstantu neboli práh. Prahování je rychlá a široce používaná metoda v různých aplikacích počítačového vidění. [2]

Prahování je transformace vstupního šedotónového obrazu f do výstupního binárního obrazu g podle následující rovnice.

$$g(x,y)=\begin{cases} 1 & \text{pro } f(x,y)\geq T \\ 0 & \text{pro } f(x,y)<T \end{cases}, \tag{3.1}$$

kde T je hodnota prahu. Pokud je nastavena pouze jedna hodnota prahu T, jedná se o prahování globální.[3]

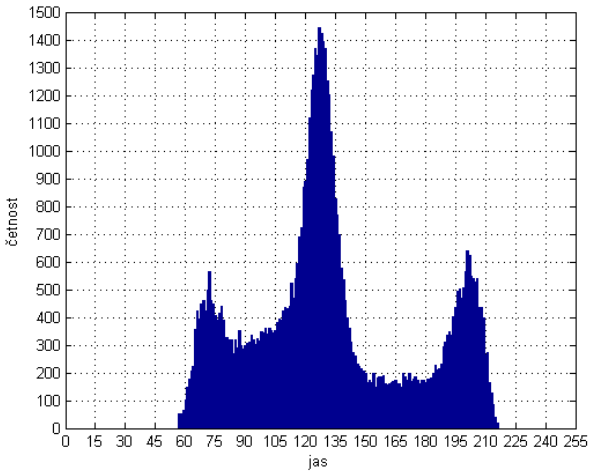


(a) (b)

Obrázek č. 3.1: Prahování. (a) Šedotónový obraz. (b) Prahovaný binární obraz.

3.1.1 Určování hodnoty prahu na základě histogramu

Jedna z metod určování prahu se zakládá na analýze histogramu. Histogram zobrazuje četnosti jednotlivých jasů v šedotónovém obraze. Pro obrázek 3.1a je znázorněný histogram na následujícím obrázku.



Obrázek č. 3.2: Histogram pro obr 3.1a

Metoda volby optimálního prahu OTSU

Metodu OTSU určuje optimální práh mezi dvěma třídami. Tato metoda je založena na minimalizaci vážených rozptylů uvnitř jednotlivých tříd vzniklých na základě volby prahu. Histogram obrazu bereme jako pravděpodobnostní funkci *P*, kde *P*(0),...,*P*(*N*) určují pravděpodobnosti výskytu jednotlivých intenzit zastoupených v šedotónovém obraze. [1]

Rozptyl uvnitř tříd σ_w^2 definujeme jako:

$$\sigma_w^2(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T), \tag{3.2}$$

kde $\sigma_1^2(T)$ je rozptyl pro třídu s hodnotami, které jsou nižší nebo rovny prahu *T* (třída 1). $\sigma_2^2(T)$ je rozptyl pro třídu s hodnotami, které jsou vyšší než práh *T* (třída 2). $q_1(T)$ je pravděpodobnost třídy 1. $q_2(T)$ je pravděpodobnost třídy 2. $\mu_1(T)$ střední hodnota třídy 1. $\mu_2(T)$ střední hodnota třídy 2. Tyto parametry jsou vypočítávány podle:

$$q_1(T) = \sum_{n=1}^{T} P(n) \tag{3.3}$$

$$q_2(T) = \sum_{n=T+1}^{N} P(n) \tag{3.4}$$

$$\mu_1(T) = \sum_{n=1}^{T} i \frac{P(n)}{q_1(T)} \tag{3.5}$$

$$\mu_2(T) = \sum_{n=T+1}^{N} i \frac{P(n)}{q_2(T)} \tag{3.6}$$

$$\sigma_1^2(T) = \sum_{n=1}^{T} [n - \mu_1(T)]^2 \frac{P(n)}{q_1(T)} \tag{3.7}$$

$$\sigma_2^2(T) = \sum_{n=T+1}^{N} [n - \mu_2(T)]^2 \frac{P(n)}{q_2(T)} \tag{3.8}$$

Výsledný práh T_{opt} může být vypočítán postupným dosazováním všech možných hodnot T a zjištěním pro které T je $\sigma_w^2(T)$ minimální. V [1] je popsán, iterativní výpočet, který snižuje výpočetní náročnost tohoto algoritmu.

3.1.2 Adaptivní práh

Některé faktory mohou ovlivnit obraz tak, že nelze jednoznačně rozhodnout o hodnotě prahu k oddělení objektů od pozadí, například nerovnoměrné osvětlení obrazu. Tato situace se dá vyřešit rozdělením obrazu do menších částí a pro každou tuto část vypočítat práh zvlášť. Tato metoda se nazývá adaptivní prahování. [3]

Při adaptivním prahování řešíme úlohu, na jak velké části obraz rozdělit a jak v těchto částech určovat hodnotu prahu. Jedna z možností je pro každou oblast určovat optimální práh metodou OTSU.

Další přístup je určovat práh ze střední hodnoty dané oblasti: [10]

$$T_{ad} = \mu - c\,, \tag{3.9}$$

kde μ je střední hodnota jasu v dané oblasti a c je vhodně zvolená konstanta.

Obrázek č. 3.3: Výsledek adaptivního prahování pro obrázek 3.1a.

3.1.3 Modifikovaný adaptivní práh

Modifikovaný adaptivní práh vychází z adaptivního prahu popsaného v 3.1.2. Metoda podle 3.1.2 prahuje každou část obrazu. Pokud ale rozdělíme obraz na malé oblasti, může se stát, že v některé z těchto oblastí nebude žádná užitečná informace (bude zachycena jen část tmavé oblohy bez hvězd, meteoru, atd.). Tato oblast bude prahována vypočteným prahem T a v binárním obraze vznikne nežádoucí šum. Tomuto efektu můžeme předejít přidáním podmínky pro prahování v dané oblasti, která se odvíjí od rozptylu jasových hodnot.

Pro danou oblast v obraze se nejprve vypočte rozptyl jasových hodnot. Adaptivní práh pro tuto oblast pak bude vypočten podle rovnice

$$T_{ad} = \begin{cases} \mu - c_1, & \text{pro } \sigma \geq c_2 \\ 256, & \text{pro } \sigma < c_2 \end{cases}, \tag{3.10}$$

kde μ je střední hodnota jasu v dané oblasti, c_1 je konstanta prahování, σ je rozptyl jasových hodnot v oblasti a c_2 je konstanta podmínky prahování. Prahování je pak provedeno podle rovnice (3.1) s vypočteným prahem T_{ad}.

3.2 Označování spojených oblastí (labelling)

Při označování spojených oblastí chceme každou samostatnou oblast v binárním obraze označit unikátním znakem. Výsledkem této metody je dvourozměrné pole, které rozměrově odpovídá původnímu obrazu. Jeho hodnoty můžou nabývat libovolných znaků, podle zvolených značek.

Existuje více různých algoritmů pro označení souvislých oblastí. Některé algoritmy jsou stavěny na výpočet v co nejkratším čase, některé jsou určeny pro velké obrazy a pro úsporu paměti. V práci bude využit algoritmus označování oblastí popsaný v [5].

3.3 Detekce hran

Hranami v obraze označujeme místa, kde dochází k nespojitosti v obrazové funkci. Nespojitosti můžou být v jasu, barvě v textuře atd. Detekce hran je jedna z metod segmentace obrazu.

Změna obrazové funkce může být popsána gradientem, který představují body ve směru největšího růstu obrazové funkce. Hrana je vlastnost náležící jednomu pixelu a je počítána z chování obrazové funkce v okolí tohoto pixelu. Jde o vektor dvou složek, významu a směru. Význam hrany je význam gradientu a směr hrany φ je směr gradientu ψ rotovaný o úhel -90°. Směr gradientu udává směr největšího růstu funkce. Význam gradientu |grad $f(x, y)$| a směr gradientu ψ jsou pro spojitou obrazovou funkci vypočítány podle:

$$|\text{grad} \quad f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \tag{3.11}$$

$$\psi = \arg\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right), \tag{3.12}$$

kde arg(x,y) je úhel (v radiánech) z osy **x** do bodu (x,y). [2]

Protože obrazy v počítači nejsou reprezentovány spojitou funkcí, v rovnicích 3.11 a 3.12 aproximujeme derivace diferencemi. První diference diskrétního obratu **I** ve vertikálním směru (pro pevné r) a v horizontálním směru (pro pevné c) jsou dány vztahy:

$$\Delta_r \mathbf{I}[r,c] = \mathbf{I}[r,c] - \mathbf{I}[r-n,c], \tag{3.13}$$

$$\Delta_c \mathbf{I}[r,c] = \mathbf{I}[r,c] - \mathbf{I}[r,c-n], \tag{3.14}$$

kde n je malé celé číslo, nejčastěji rovné 1. [2]

Pro detekci hran se používají gradientní operátory. Ty můžeme rozdělit do tří skupin:

1) Operátory aproximující derivace obrazové funkce použitím diferencí.

2) Operátory založené na průchodu nulou 2. derivace obrazové funkce.
3) Operátory, které se pokouší srovnat obrazovou funkci s parametrickým modelem hrany.

Více o jednotlivých typech gradientních operátorů popisuje [2]. Dále bude popsána nejvíce používaná metoda detekce hran.

3.3.1 Cannyho detekce hran

Cannyho detektor hran je velmi populární a efektivní metoda vhodná pro detekci hran porušených šumem. Optimalita detektoru je závislá na třech kritériích: [2]

- **Kritérium detekce** vyjadřuje fakt, že důležité hrany by neměly být vynechány a neměly by se objevit falešné odpovědi.
- **Kritérium lokalizace** říká, že vzdálenost mezi aktuální a lokalizovanou pozicí hrany by měla být minimální.
- **Kritérium jediné odpovědi** minimalizuje více odpovědí do jedné hrany. To je částečně pokryto prvním kritériem, pokud jsou dvě odpovědi pro jednu hranu, jedna z nich by měla být označená jako falešná. Toto třetí kritérium řeší problém hrany porušené šumem a pracuje proti nehladkým hranovým operátorům.

Algoritmus Cannyho detekce hran [2]

1) Provede konvoluci obrazu f s Gaussiánem o rozsahu σ.
2) Nastaví směr normály hrany podle vzorce
$$n = \frac{\nabla(G * f)}{|\nabla(G * f)|}$$
pro každý bod v obraze.
3) Najde oblast hrany použitím rovnice
$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0 .$$
4) Spočítá význam hrany podle
$$|G_n * f| = |\nabla(G * f)| .$$
5) Provede filtraci hran pro eliminaci falešných odpovědí podle:
   a) Označí všechny hrany s významem větším než T_1 za opravdové a hrany s významem menším než T_0 za falešné.

b) Projde všechny body v obraze s významem v rozsahu [T_0, T_1].
c) Pokud v okolí aktuálního bodu je jiný bod, který už je označený jako hrana, označí aktuální bod také za hranu.
d) Opakuje od bodu b, dokud budou probíhat změny.

3.4 Matematická morfologie

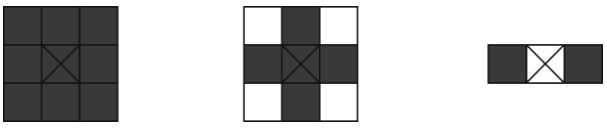
Matematická morfologie je samostatná oblast analýzy obrazu. Zabývá se hlavně prací s tvarem objektů, který může být porušen. Výhodou těchto metod je, že jsou výpočetně nenáročné. V práci je uvedeno pouze několik málo základních metod, které budou dále využity v algoritmech pro detekci meteoru. Tyto metody mohou pracovat i s šedotónovým obrazem. V této práci budou používány pouze pro zpracování binárního obrazu.

Binární obraz může být představen jako množina uspořádaných dvojic. Dvojice odpovídá souřadnicím bodu v obraze. Body patřící objektu v obraze reprezentuje množina X, tyto body jsou pixely s hodnotou 1. Body patřící komplementární množině X^c pozadí s hodnotou pixelu rovnou 0. Počátek má souřadnice (0,0) a v příkladech je označen křížkem. [2]

Definice3.1: Morfologická transformace Ψ je dána relací obrazu (množiny bodů X) s jinou malou množinou bodů B nazývanou strukturní element. [2]

Aplikace morfologické transformace $\Psi(X)$ na obraz X znamená, že strukturní element S je systematicky posouván přes celý obraz. Předpokládáme, že S je umístěn v nějakém bodě obrazu. Bod v obraze odpovídající počátku strukturního elementu je nazván okamžitý pixel. Výsledek relace mezi obrazem X a strukturním elementem S v aktuální pozici je uložen do výstupního obrazu na pozici odpovídající okamžitému bodu. [2]

Na následujícím obrázku jsou zobrazeny typické strukturní elementy.



(a) (b) (c)

Obrázek č. 3.4: Typické strukturní elementy.

3.4.1 Binární dilatace

Morfologická transformace dilatace $\oplus$ kombinuje dvě množiny použitím vektorového součtu.

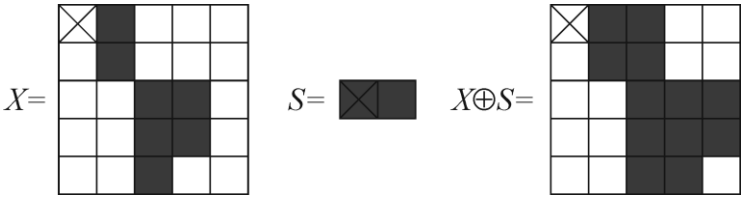
$$X \oplus S = \left\{ p \in \varepsilon^2 : p = x + s, \quad x \in X, s \in S \right\}, \tag{3.18}$$

Příklad dilatace:

$X = \{(1,0),(1,1),(2,2),(2,3),(2,4),(3,2),(3,3)\}$

$S = \{(0,0),(1,0)\}$

$X \oplus S = \{(1,0),(1,1),(2,0),(2,1),(2,2),(2,3),(2,4),(3,2),(3,3),(3,4),(4,2),(4,3)\}$



Obrázek č. 3.5: Příklad binární dilatace.

3.4.2 Binární eroze

Morfologická transformace eroze $\ominus$ kombinuje dvě množiny použitím vektorového rozdílu.

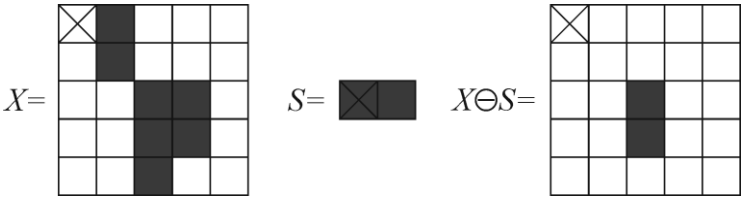
$$X \ominus S = \left\{ p \in \varepsilon^2 : p = x + s \in X, \quad \forall s \in S \right\}, \tag{3.19}$$

Příklad eroze:

$X = \{(1,0),(1,1),(2,2),(2,3),(2,4),(3,2),(3,3)\}$

$S = \{(0,0),(1,0)\}$

$X \ominus S = \{(2,2),(2,3)\}$



Obrázek č. 3.6: Příklad binární eroze.

3.4.3 Otevření

Morfologická transformace otevření je eroze obrazu následovaná dilatací se stejným strukturním elementem. Protože tyto dvě operace nejsou inverzní, výsledek není původní obraz, ale verze obrazu s méně detaily.

3.4.4 Uzavření

Morfologická transformace uzavření je dilatace obrazu následovaná erozí se stejným strukturním elementem. Výsledek je obraz, ve kterém se zaplní úzké díry a zálivy (v závislosti na velikosti strukturního elementu) a objekty, které jsou blízko u sebe, se spojí.

Vzhledem k okrajovému použití metod matematické morfologie se jimi dále v této práci zabývat nebudeme. Více o morfologických transformacích a o jejich vlastnostech můžeme najít například v [2].

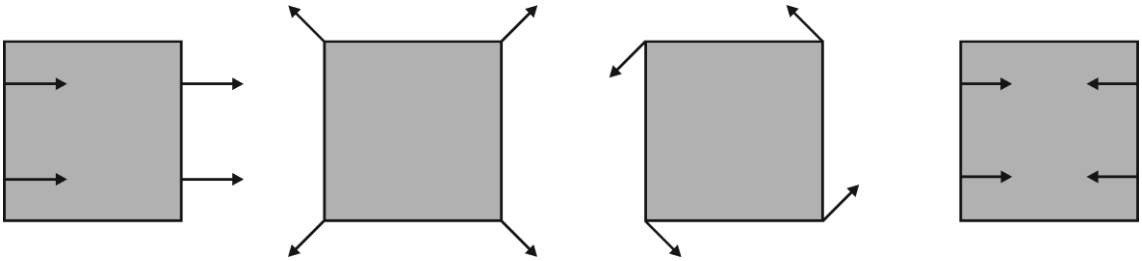
3.5 Optický tok

Výpočet optického toku zachycuje informaci o veškerém pohybu v obraze, respektive jasových změn v čase *t*. Optický tok může být použit ke studiu různých variant pohybu – pohybující se pozorovatel a statické objekty, statický pozorovatel a pohybující se objekty nebo obojí. [2]

Existují metody pro výpočet optického toku v zájmových bodech i v každém bodě digitálního obrazu.

Pohyb, který se vyskytuje v dynamických obrazech, je obvykle kombinace čtyř základních pohybů. Jsou to:

- Translační pohyb v konstantní vzdálenosti od pozorovatele.
- Translační pohyb do hloubky.
- Rotace kolem osy pohledu.
- Rotace kolmá na osu pohledu



Obrázek č. 3.7: Základní druhy pohybu. (a) Translace v konstantní vzdálenost. (b) Translace do hloubky. (c) Rotace v ose pohledu. (d) Rotace kolmá na osu pohledu

Pro výpočet optického toku předpokládejme, že máme posloupnost obrazů popsaných funkcí $f(x, y, t)$, kde proměnná t určuje čas (pořadí) snímku v posloupnosti. Takovou posloupnost můžeme nazvat dynamickým obrazem. Reprezentace dynamického obrazu jako funkce polohy a času může být vyjádřena Taylorovým rozvojem:

$$f(x+dx, y+dy, t+dt) = f(x, y, t) + f_x dx + f_y dy + f_t dt + O(\partial^2), \tag{3.20}$$

kde f_x, f_y, f_t jsou parciální derivace funkce f. Pokud budeme předpokládat, že na následujícím snímku bude poloha (x, y) změněna o vzdálenost (dx, dy) během časového úseku dt, můžeme tuto změnu vypočítat z jasové rovnosti:

$$f(x+dx, y+dy, t+dt) = f(x, y, t). \tag{3.21}$$

Pokud změny dx, dy, dt, budou velmi malé, můžeme zanedbat vyšší členy Taylorova rozvoje v rovnici (3.20) a dosazením do rovnice (3.21) dostaneme:

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt}. \tag{3.22}$$

Cílem výpočtu je určit vektor rychlosti

$$\mathbf{c} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (u, v). \tag{3.23}$$

[2]

V této práci byly vyzkoušeny 2 implementace optického toku. První z nich byla metoda podle [6], která vypočítává optický tok v každém bodě digitálního obrazu (denzitní optický tok). Druhá metoda byla podle [7]. Ta vypočítává optický tok podle zájmových bodů.

3.6 Houghova transformace

Houghova transformace je metoda, která je využívána k nalezení parametrů matematického modelu hledaného objektu za předpokladu známých vstupních dat (obrazových bodů). Tato metoda může být použita například pro detekci přímek nebo kružnic v obraze. Protože meteor viditelný na noční obloze má tvar přímky, bude v této práci použita Houghova transformace právě pro detekci přímek.

3.6.1 Princip Houghovy transformace

Mějme dva body $A = (x_1, y_1)$, $B = (x_2, y_2)$. Všechny přímky procházející bodem A lze popsat rovnicí

$$y_1 = kx_1 + q, \tag{3.24}$$

kde k a q představují parametry přímky. Stejnou rovnici můžeme použít pro vyjádření parametru q v závislosti na souřadnicích bodu A. Všechny přímky, které procházejí bodem A pak popisuje rovnice

$$q = -kx_1 + y_1. \tag{3.25}$$

Přímky procházející bodem B popisuje rovnice

$$q = -kx_2 + y_2. \tag{3.26}$$

To znamená, že každá přímka z prostoru x,y je reprezentována jedním bodem v prostoru k,q. Přímka, která prochází bodem A i B, se tedy v prostoru parametrů zobrazí do stejného bodu. Kterýkoliv bod této přímky se zobrazí rovněž do tohoto bodu.

Každý bod v prostoru parametrů budeme nazývat akumulační buňkou. Na počátku Houghovy transformace je každý element akumulačního prostoru parametrů vynulován. Parametry přímky, která prochází bodem A i B odpovídají souřadnicím maxima v akumulačním prostoru parametrů.[2]

3.6.2 Houghova transformace s polární reprezentací přímky

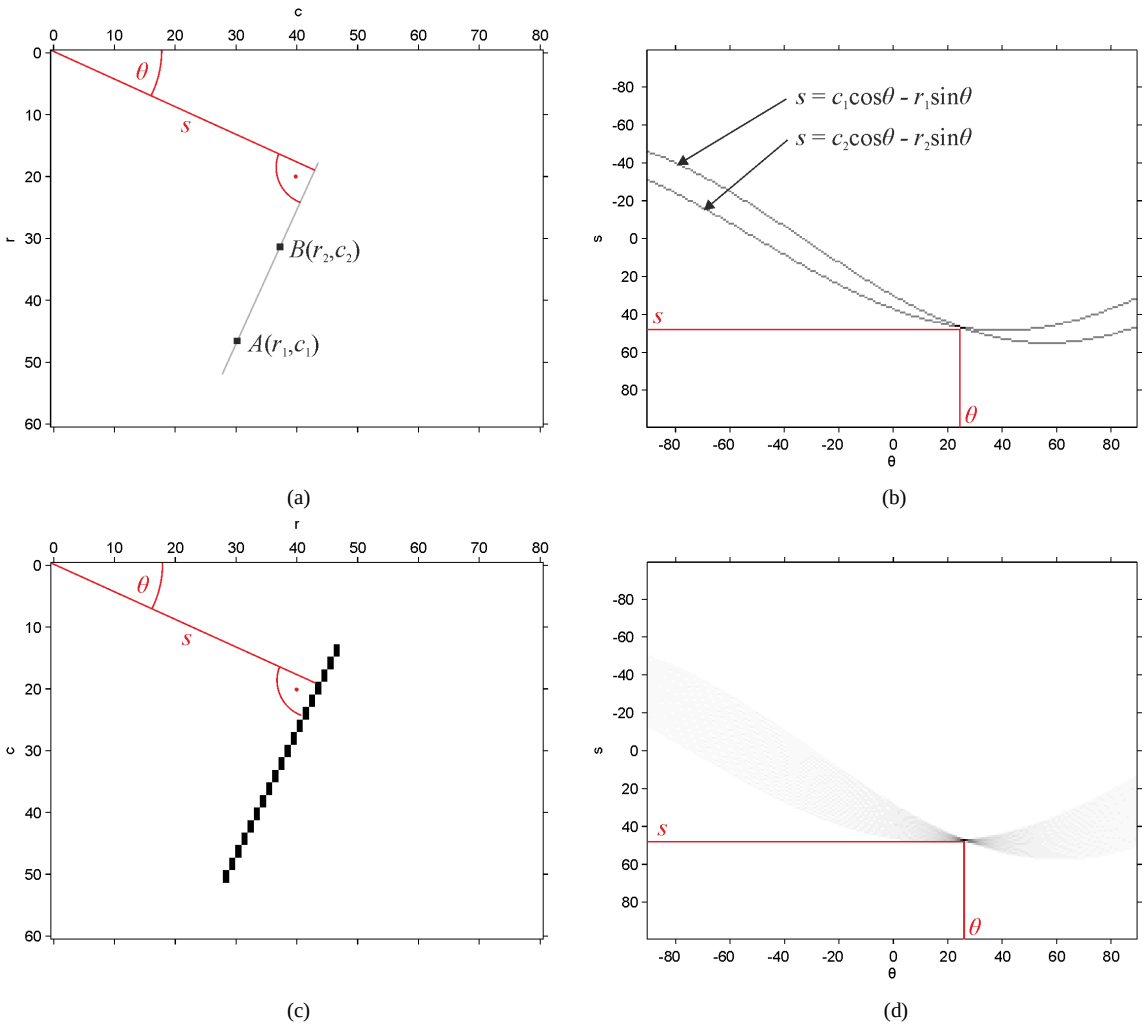
Z důvodu limitního omezení ($k \rightarrow \infty$) u vertikálních přímek se v praxi reprezentace přímky vztahem (3.24) nepoužívá. Tento problém řeší parametrický popis přímky v polárních souřadnicích.

$$s = x \cdot \cos\theta + y \cdot \sin\theta\,, \tag{3.27}$$

kde s je délka kolmice na přímku od počátku a θ je úhel, který tato přímka svírá s osou x. Pro diskrétní obraz a souřadný systém podle obrázku 2.1 přepíšeme rovnici (3.27) do následujícího tvaru:[1]

$$s = c \cdot \cos\theta - r \cdot \sin\theta\,. \tag{3.28}$$

Na následujícím obrázku je zobrazen příklad Houghovy transformace pro 2 body a pro úsečku.



Obrázek č. 3.8: Ukázka Houghovy transformace. (a) Obraz se dvěma body. (b) HT obrazu a. (c) Obraz s úsečkou. (d) HT obrazu c.

3.6.3 Progresivní pravděpodobnostní Houghova transformace

Houghova Transformace, jak byla popsána výše, detekuje standardně pouze přímky. Z důvodu výpočetní náročnosti navíc není tento výpočet HT vhodné použít pro

naše data. Algoritmus, který použijeme pro detekci úsečky, se nazývá Progresivní pravděpodobnostní Houghova transformace (dále jen PPHT). Tato metoda je popsána v [8].

Myšlenka této metody spočívá v náhodném výběru pixelu z binárního obrazu a jeho transformace do akumulátoru. Když akumulační buňka, která odpovídá konkrétní přímce, dosáhne potřebného počtu hlasů, binární obraz je prohledán podél této přímky a zjistí, jestli představuje jednu nebo více konečných čar. Všechny body na této přímce jsou následně odstraněny z binárního obrazu. Tímto způsobem vrátí algoritmus konečně dlouhé čáry.

Algoritmus PPHT [8]

1) Zkontroluje vstupní obraz, pokud je prázdný, pak končí.
2) Aktualizuje akumulátor s jedním náhodně vybraným bodem ze vstupního obrazu.
3) Odebere bod ze vstupního obrazu.
4) Zkontroluje, zda je nejvyšší vrchol v akumulátoru, který byl upraven novým bodem, větší než zvolený práh T_{line}. Pokud ne, tak jde na 1.
5) Zkontroluje body ležící na přímce určené vrcholem v akumulátoru, a najde nejdelší úsek, který je nepřetržitý nebo vykazuje mezeru menší než zvolený práh T_{gap}.
6) Odstraní body tohoto úseku ze vstupního obrazu.
7) Zruší hlasy z akumulátoru všem bodům z úseku, které předtím hlasovaly.
8) V případě, že úsek je delší než minimální délka T_{length}, přidá úsek do výstupního seznamu.
9) Návrat na bod 1.

4 Návrh metody hledání meteoru

Tato kapitola je věnována metodě pro nalezení meteoru v digitálním obrazu. Nejprve bude uveden zavedený postup používaný pro detekci úsečky, který využívá Cannyho detekci hran a bude vysvětleno, proč je tato metoda v neupraveném tvaru nevhodná pro řešení daného problému. Dále budou navrženy postupy předzpracování obrazu, které umožní použití zavedené metody. Nakonec bude navržena nová metoda pro detekci meteoru s využitím adaptivního prahování, optického toku a dvou PPHT. V poslední kapitole této práce budou srovnány úspěšnosti těchto metod.

4.1 Metoda detekce úsečky s použitím Cannyho detektoru a PPHT

Pro hledání přímek v obraze je často spojeno použití Houghovy transformace s Cannyho detektorem hran. Tato metoda detekce čáry je popsána například v [2] nebo v [3] a je vyzkoušena i pro řešenou úlohu detekce meteorů. Nejprve je získán binární obraz pomocí Cannyho detektoru hran a na takto získaný obraz je aplikována PPHT.

Algoritmus metody 4.1:

1) Vstupní obraz $\mathbf{I_g}$ převede na binární obraz $\mathbf{I_b}$ pomocí Cannyho detektoru hran popsaného v 3.3.1.
2) Binární obraz $\mathbf{I_b}$ použije pro vstup PPHT a získané výsledky uloží do seznamu.
3) Výsledek detekce je seznam nalezených čar.

Z důvodu nalezení hrany na horizontu oblohy, není tato metoda pro detekci použitelná. Dále je uvedena metoda, která tento problém odstraní.

4.2 Metody maskování zájmové oblasti

Nyní bude navrženo několik metod, které zlepší výsledek hledání úseček pomocí PPHT. Jejich princip spočívá v maskování oblastí, ve kterých může či nemůže být zachycen meteor. Oblasti, ve kterých se nemůže meteor nalézat, jsou potom při detekci

ignorovány. Tím se značně sníží počet chybných detekcí zejména vlivem detekce horizontu.

4.2.1 Maskování oblohy

V části 2.4 bylo nastíněno, jak vypadá snímaná scéna. Na obrázku 2.6a je vidět, že obloha zabírá jen určitou část snímků. Meteor se může nacházet pouze na obloze, a proto se ji budeme snažit detekovat.

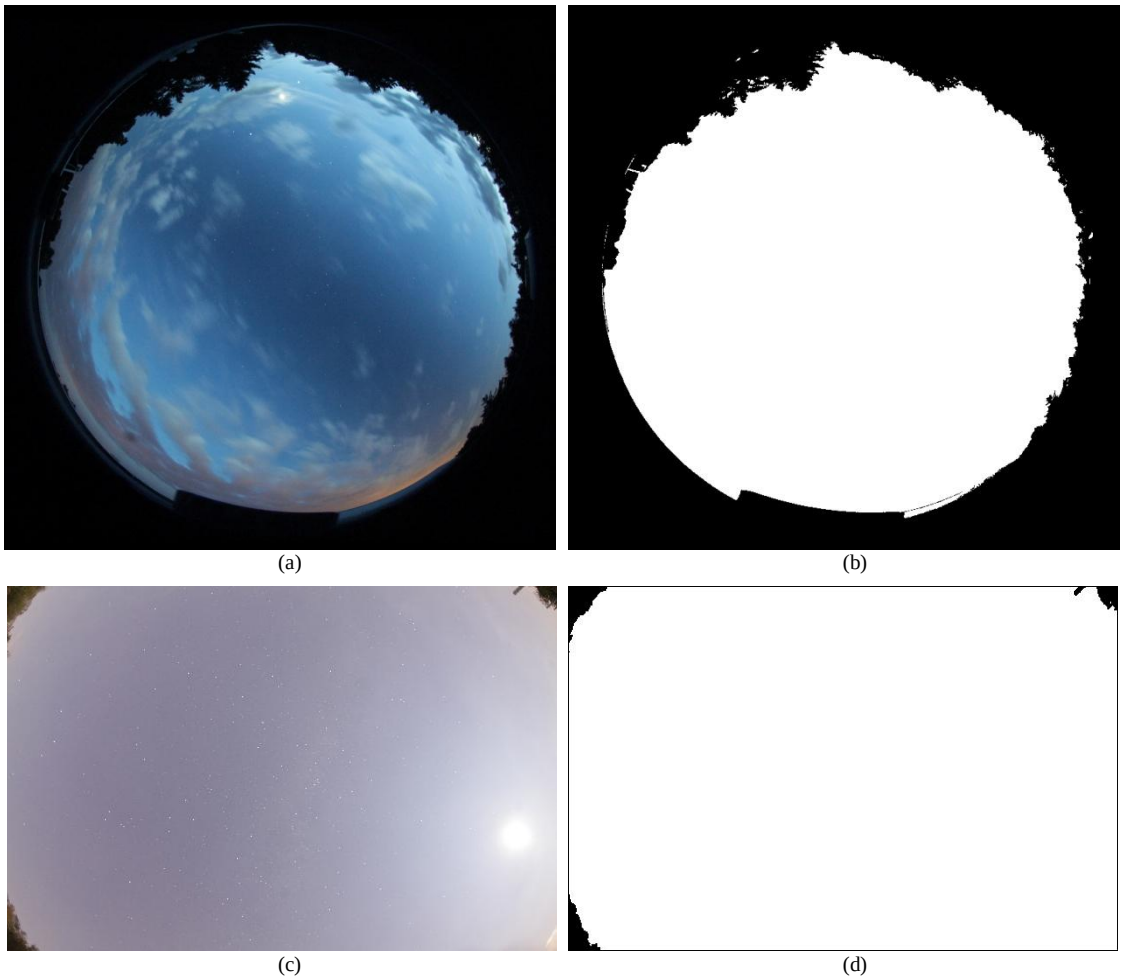
Snímek se skládá z velké části z oblohy a z velké části z obzoru a temných pixelů mimo scénu. Masku oblohy můžeme tedy úspěšně získat aplikací prahování, kde práh volíme automaticky metodou OTSU popsanou v kapitole 3. Takto získaná maska je zobrazena na obrázku 4.1b.

Některé snímky, které byly poskytnuty, neobsahují černé okolí snímané scény. Pro takové snímky selhává metoda volby automatického prahu metodou OTSU. Pro tyto snímky je vypočítávána hodnota prahu na základě střední hodnoty spočítané ze všech bodů obrazu podle:

$$T = E\left[\mathbf{I}_{\mathrm{s}}(r,c)\right] + k\,, \tag{4.1}$$

kde T je výsledný práh, $\mathbf{I}_{\mathbf{g}}(r, c)$ je zkoumaný obraz a k je konstanta přičtená ke střední hodnotě. Tato rovnice je shodná s rovnicí (3.9) pro adaptivní prahování. Narozdíl od adaptivního prahování je ale tento práh použit na celý obraz. Volba konstanty k byla získána experimentálně při provádění testů. Takto získaná maska je zobrazena na obrázku 4.1d.

Volba metody prahování je vybrána automaticky podle počtu temných bodů (body s nižší jasovou hodnotou než 5). Pokud část obrazu tvoří tyto temné body, je zvolena metoda OTSU. Pokud ne práh je vypočítán podle 4.1.



Obrázek č. 4.1: Maskování oblohy. (a) Scéna Ondřejov. (b) Maska scény Ondřejov. (c) Scéna Austrálie. (d) Maska scény Austrálie.

Takto získanou masku můžeme „oříznout“ použitím morfologické operace binární eroze. Volba velikosti strukturního elementu ovlivňuje velikost oříznutí. Pokud erodovanou maskou vynásobíme binární obraz, odstraní se hrany vzniklé na horizontu oblohy.

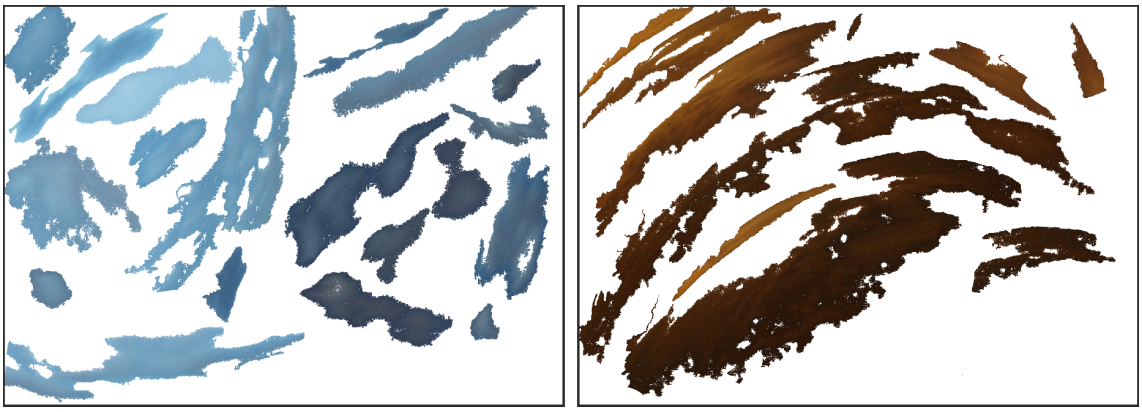
4.2.2 Maskování mraků a světelného znečištění

Mraky a světelné znečištění jsou častým jevem na snímcích noční oblohy. Tyto jevy značně znesnadňují úlohu automatické detekce meteorů. Jejich přítomností vzniknou v binárním obrazu nežádoucí objekty, které mohou být později detekovány jako meteor. Proto se je budeme snažit na snímcích detekovat a následně ignorovat při průběhu detekce.

Pro detekci mraků a světelného znečištění bude využita informace o barvě v obrazu. Většina mraků má barevné složení rozdílné od zachycených meteorů. Mraky při tmavé obloze jsou osvětlené městy a mají barvu do oranžova. Naopak mraky při soumraku nebo při úsvitu mají modrošedou barvu.

Trénování barvy mraků

Mraky rozdělíme podle barvy do dvou skupin, „oranžové“ a „modré“. Pro každou z těchto skupin vytvoříme obraz obsahující reprezentanty těchto mraků. Ty jsou vybrané z různých částí nocí.

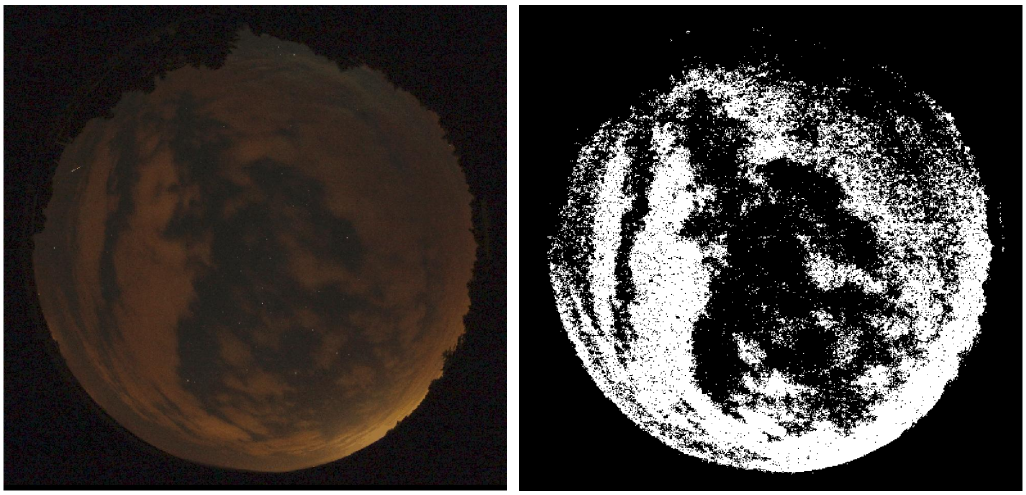


(a) (b)

Obrázek č. 4.2: 2 reprezentace barvy mraků.

Tyto obrazy jsou převedeny do barevného prostoru HSV a následně je vypočten 2D histogram odstínu a sytosti, tedy složek H a S. Tím je potlačen vliv jasu. Pro testovaný soubor je poté vypočtena zpětná projekce histogramů do tohoto souboru. To znamená, že pro každý histogram je spočítána pravděpodobnost, že daný pixel pochází z pravděpodobnostního rozdělení reprezentovaného daným histogramem. Body s vysokou hodnotou pravděpodobnosti jsou poté použity jako maska mraků a jsou při detekci ignorovány.

Tímto postupem se podaří odstranit většina chybných detekcí, způsobených mraky a světelným znečištěním. Bohužel některé mraky mají barevné složení podobné zachyceným meteorům. Jsou to mraky blízké odstínům šedi. Takové mraky nelze tímto způsobem odstranit, byly by totiž ze snímku odstraněny i případné meteory.



Obrázek č. 4.3: Detekce mraků pomocí barev.

4.2.3 Odstranění statických objektů (rozdílový obraz)

Protože meteor je pohybující se objekt, nebude se zřejmě nacházet na dvou po sobě jdoucích snímcích na stejném místě. Naopak některé objekty (zejména na horizontu) svoji polohu s časem nemění. Pokud použijeme metodu pro získání binárního obrazu (například adaptivní prahování) na dva po sobě jdoucí snímky a poté uděláme rozdílový obraz (ten získáme prostým odečtením snímků) z výsledného binárního obrazu zmizí statické objekty.

Formálně, mějme 2 šedotónové obrazy $\mathbf{I_{g1}}$ a $\mathbf{I_{g2}}$, které jsou zaznamenány v krátkém časovém úseku. Na obou obrazech provedeme adaptivní prahování se stejnými parametry a získáme binární obrazy $\mathbf{I_{b1}}$ a $\mathbf{I_{b2}}$. Odečtením obrazu $\mathbf{I_{b2}}$ od $\mathbf{I_{b1}}$ získáme rozdílový obraz $\mathbf{I_{bR}}$, ze kterého jsou odstraněny statické objekty. Případný meteor zachycený na snímku $\mathbf{I_{g1}}$ odstraněn nebude, jelikož snímek $\mathbf{I_{g2}}$ vznikl v pozdějším čase.

Bohužel touto metodou nelze odstranit hvězdy. Ty nejsou vlivem rotace Země statické. Pro odstranění hvězd bude použit výpočet optického toku, který je obsažen v navržené metodě popsané v 4.4.

4.3 Rozšířená metoda detekce úsečky s použitím Cannyho detektoru a PPHT

Metodu popsanou v 4.1 nebylo možné použít kvůli vzniku chybných detekcí způsobených detekcí hrany na horizontu oblohy. Přidáním metody maskování oblohy, která byla popsána v 4.2.1, budou tyto detekce odstraněny. Pro přesnější výsledky je přidána i metoda maskování mraků a světelného znečištění, popsaná v 4.2.2.

Algoritmus metody 4.3:

1) Ze vstupního obrazu $\mathbf{I_c}$ vytvoří masku mraků $\mathbf{M_1}$ metodou 4.2.2.
2) Obraz $\mathbf{I_c}$ převede na šedotónový obraz $\mathbf{I_g}$.
3) Z obrazu obrazu $\mathbf{I_g}$ vytvoří masku $\mathbf{M_2}$ podle metody 4.2.1 a aplikuje na ní operaci binární eroze.
4) Z šedotónového obrazu $\mathbf{I_g}$ vytvoří binární obraz $\mathbf{I_b}$ pomocí Cannyho detektoru hran popsaného v 3.4.
5) Binární obraz $\mathbf{I_b}$ vynásobí maskou $\mathbf{M_2}$ a odečte masku $\mathbf{M_1}$. Výsledek je redukovaný binární obraz $\mathbf{I_{b2}}$.
6) Binární obraz $\mathbf{I_{b2}}$ použijeme pro vstup PPHT a získané výsledky uložíme do seznamu.
7) Výsledek detekce je seznam nalezených čar.

(a) (b) (c)

Obrázek č. 4.4: Postup metody 4.3. (a) Šedotónový obraz. (b) Binární obraz získaný Cannyho detektorem. (c) Výsledek detekce.

4.4 Návrh metody s využitím adaptivního prahování, optického toku a PPHT

Při navrhování této metody byl brán velký důraz na poskytnutá data. Postup použití jednotlivých metod popsaných v kapitole 3 byl navržen tak, aby metoda dávala co nejpřesnější výsledky na poskytnutých datech. V této metodě jsou dále zahrnuty všechny 3 metody popsané v 4.2 pro zlepšení kvality detekce.

Jako první krok je vypočítána maska mraků metodou 4.2.2. Dále je převeden barevný obraz na šedotónový a z něj je vypočítána maska oblohy. Aplikací modifikovaného adaptivního prahování na šedotónový obraz je získán obraz binární. Binární obraz je násoben maskou oblohy, aby se odstranily hrany zaznamenané na obzoru.

Dále jsou ze snímku odstraněny hvězdy pomocí výpočtu optického toku. Myšlenka je taková, že pokud spočítáme optický tok mezi předchozím snímkem a následujícím snímkem (ob snímek, prostřední je snímek s potenciálním meteorem), získáme informaci o tom, kde se pohybovaly hvězdy. Tento pohyb můžeme následně odečíst od binárního obrazu s meteorem a hvězdy tak z obrazu eliminovat.

Po adaptivním prahování zůstane v obrazu zbytkový nežádoucí šum, který vznikl v segmentech, kde byla splněna prahovací podmínka, ale zároveň v ní nebyl zaznamenán pohyb optickým tokem. Tyto oblasti budou velice malé (jasné hvězdy by byly nalezeny výpočtem optického toku). Tyto malé oblasti v obraze jsou detekovány a odstraněny.

V takto vzniklém binárním obraze jsou dále hledány úsečky pomocí PPHT. PPHT je použita dvakrát po sobě. První detekce je nastavena na krátké čáry bez mezer a provádí se z důvodu nalezení krátkých segmentů meteoru. Druhá detekce je nastavena na delší čáry s mezerami a je použita pro detekci celých meteorů, odstraní ale falešné detekce vzniklé první transformací.

4.4.1 Binární obraz pomocí adaptivního prahování

Získání binárního obrazu $\mathbf{I_b}$ z šedotónového obrazu $\mathbf{I_g}$ je zde řešeno modifikovaným adaptivním prahováním popsaným v 3.1.3. Z binárního obrazu je poté odstraněno světelné znečištění a statické objekty. Toto odstranění jsem popsal v 4.2. Volba jednotlivých parametrů je diskutována v kapitole 6.

4.4.2 Detekce hvězd s využitím optického toku

Další fází je odstranění hvězd z binárního obrazu. Mějme 3 po sobě jdoucí snímky $\mathbf{I_g}(t\text{-}1)$, $\mathbf{I_g}(t)$ a $\mathbf{I_g}(t\text{+}1)$ a předpokládejme, že na snímku $\mathbf{I_g}(t)$ je zachycen meteor. Pro získání informace o pohybu hvězd vypočteme optický tok mezi snímky $\mathbf{I_s}(t\text{-}1)$ a $\mathbf{I_g}(t\text{+}1)$. Získanou informaci o pohybu významových bodů (což budou nejčastěji hvězdy) vykreslíme do nového obrazu $\mathbf{I_{OF}}$ (binární obraz). Takto získaný obraz odečteme od binárního obrazu získaného v 4.4.1

$$\mathbf{I_{b-OF}} = \mathbf{I_b} - \mathbf{I_{OF}} \,. \tag{4.2}$$

4.4.3 Odstraňování malých oblastí

V obraze $\mathbf{I_{b\text{-}OF}}$ zůstane nežádoucí šum vzniklý adaptivním prahováním. Segmenty zachyceného meteoru mají určitou velikost. Oblasti, které jsou menší než nejmenší ze segmentů zachycených meteorů, můžeme prohlásit za šum a všechny tyto oblasti z obrazu eliminovat.

Eliminace malých oblastí je realizována pomocí označování oblastí (labelling [2]). Označíme všechny oblasti v obraze a pro každou oblast potom určíme velikost na základě počtu bodů se stejnou značkou. Oblasti menší než daná mez poté obarvíme načerno.

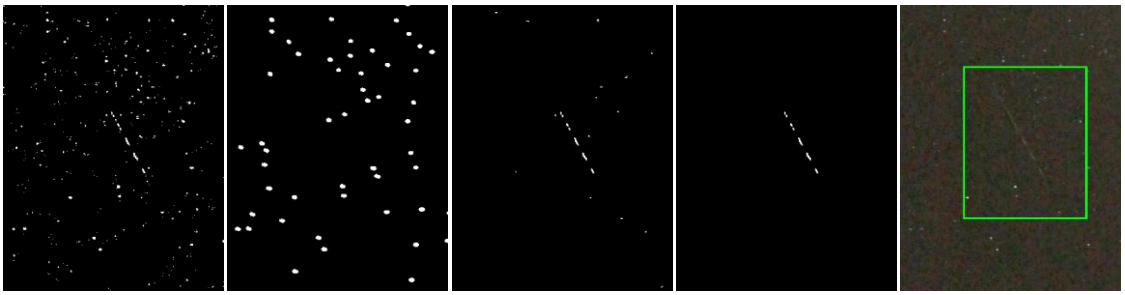
4.4.4 Detekce meteorů

Po odstranění šumu (malých oblastí) z obrazu $\mathbf{I_{b\text{-}OF}}$ je tento obraz použit pro detekci úseček pomocí algoritmu PPHT. Jak už bylo zmíněno dříve, PPHT bude použita dvakrát. První PPHT má za úkol nalézt malé segmenty meteoru vzniklé použitím elektronické clony fotoaparátu (viz. 2.3). Tato první PPHT odstraní z obrazu většinu nežádoucích bodů. Druhá PPHT je nastavena na hledání delších úseků s mezerami tak, aby nalezla celý meteor. Použitím dvou PPHT je značně snížen počet falešných detekcí.

Algoritmus metody 4.4:

1) Ze vstupního snímku $\mathbf{I_c}(t)$ v čase t vytvoří masku mraků $\mathbf{M_1}$.
2) Převede snímky $\mathbf{I_c}(t\text{-}1)$, $\mathbf{I_c}(t)$, $\mathbf{I_c}(t\text{+}1)$ na šedotónové obrazy $\mathbf{I_g}(t\text{-}1)$, $\mathbf{I_g}(t)$, $\mathbf{I_g}(t\text{+}1)$.

3) Z šedotónového snímku $\mathbf{I_g}(t)$ vytvoří masku oblohy $\mathbf{M_2}$.
4) Provede modifikované adaptivní prahování snímku $\mathbf{I_g}(t)$. Výsledkem je obraz $\mathbf{I_b}(t)$.
5) Odstraní z obrazu $\mathbf{I_b}(t)$ mraky a horizont. $\mathbf{I_{b2}}(t) = \mathbf{I_b}(t) \cdot \mathbf{M_2} - \mathbf{M_1}$.
6) Vytvoří obraz pohybu hvězd $\mathbf{I_{OF}}$ pomocí optického toku mezi snímky $\mathbf{I_g}(t-1)$ a $\mathbf{I_g}(t+1)$.
7) Odečte obraz $\mathbf{I_{OF}}$ od obrazu $\mathbf{I_{b2}}(t)$. Ve vzniklém obraze $\mathbf{I_{b2\text{-}OF}}$ odstraní malé oblasti.
8) Provede první PPHT pro detekci segmentů meteoru.
9) Provede druhou PPHT pro detekci celých meteorů.
10) Výsledek je seznam detekovaných oblastí.



(a) (b) (c) (d) (e)

Obrázek č. 4.5: Postup metody 4.4 (stejný snímek jako u metody 4.3). (a) Binární obraz získaný modifikovaným adaptivním prahováním. (b) Pohyb hvězd získaný z optického toku. (c) Rozdílový obraz získaný odečtením obrazu b od a a odstranění malých oblastí. (d) Výsledek první PPHT. (e) Výsledek druhé PPHT.

4.5 Filtrace výsledků (heuristická metoda)

Metody popsané výše nalézají úspěšně přímkové útvary na noční obloze. Bohužel na noční obloze nemají charakter rovné čáry pouze meteory ale například také letadla. Následující myšlenkou může být provedena filtrace získaných výsledků.

Myšlenka: Meteor se nebude nacházet 3 snímky po sobě na stejných souřadnicích nebo v jejich blízkém okolí.

Pravděpodobnost, že tento jev nastane, je velmi malá, v trénovacích datech nebyl ani jeden případ tohoto jevu. Získané výsledky tedy můžeme znovu projít a zkoumat, zdali se tento jev někde zaznamenal. Pokud ano, tak ze všech třech snímků smažeme informaci o nálezu na těchto souřadnicích. Většina těchto jevů budou právě detekovaná letadla.

Odvážnější tvrzení je, že dva po sobě jdoucí snímky nebude meteor zaznamenán na stejných souřadnicích. Touto filtrací je možné, že ztratíme informaci o nalezeném meteoru. Z toho důvodu bude tato „agresivní“ filtrace použita pouze experimentálně.

5 Návrh aplikace

V této kapitole je popsána vyvíjená aplikace pro automatickou detekci meteorů. Nejprve bude stručně popsán zvolený programovací jazyk a programové knihovny, které budou použity. Dále bude uveden návrh uživatelského rozhraní a popsány funkce aplikace. Nakonec se budeme zabývat paralelizací jednotlivých metod pro vícejádrový procesor.

5.1 Programovací jazyk C++

Jazyk C++ je vytvořen na pevných základech jazyka C. Oproti klasickému ANSI C přináší podporu objektově orientovaného programování. Zachovává všechny vlastnosti jazyka C.[9]

Jazyk C++ byl vybrán pro jeho vlastnosti, nezávislost na platformě a možností použít některé rozšiřující knihovny. Značnou výhodou je možnost použití knihovny pro počítačové vidění OpenCV. Program napsaný v jazyce C++ není problém přeložit například pro Windows, Linux nebo MacOS. Navíc oproti klasickému jazyku C podporuje objektově orientované programování.

5.2 Knihovna OpenCV

OpenCV (Open Source Computer Vision Library) je otevřená softwarová knihovna pro počítačové vidění a strojové učení. OpenCV byla vytvořena, aby zajistila společný základ pro aplikace počítačového vidění a urychlila používání strojového vidění v komerčních produktech.[10]

Knihovna má více než 2500 optimalizovaných algoritmů, které zahrnují komplexní sadu klasických i nově vyvíjených metod počítačového vidění a algoritmů strojového učení. Tyto algoritmy mohou být použity například k detekci a rozpoznání tváře, k identifikaci objektů, ke sledování pohybu kamery nebo pohybujících se objektů a k řadě dalších úloh. [10]

Z popsaných vlastností knihovny OpenCV je zřejmé, že její využití je ideální i pro vyvíjenou aplikaci pro automatickou detekci meteorů.

5.2.1 Třída pro reprezentaci obrázku v OpenCV

Obrázek je v OpenCV reprezentován jako matice číselných hodnot. Pro její uložení slouží objektová třída *cv::Mat*. Každý element této matice představuje jeden obrazový bod. Pro šedotónový obraz nabývají elementy hodnot odpovídající jednomu bytu, 0 odpovídá černé barvě a 255 odpovídá bílé barvě. Pro barevný obraz obsahuje každý element matice trojici hodnot reprezentující 3 základní barvy z modelu RGB {červená, zelená, modrá}.[4]

OpenCV dovoluje použít třídu *cv::Mat* i pro matice reálných čísel (datového typu float). Toho bude využito pro ukládání hodnot při různých metodách zpracování obrazu. Většina těchto metod je implementována pro matice různých datových typů, zbytek metod vyžaduje matici určitého datového typu.[4]

Vlastnosti třídy *cv::Mat* a návody pro práci s ní a s načteným obrázkem jsou uvedeny například v [4]. Kompletní dokumentaci k třídě *cv::Mat* lze najít v [10].

5.2.2 Funkce implementované v OpenCV

Většina metod pro zpracování obrazu popsaných v kapitole 3 je již v knihovně OpenCV implementována. Dále je uvedena tabulku názvů funkcí knihovny OpenCV, které odpovídají popsaným metodám v kapitole 3. Tyto funkce jsou použity ve vyvíjené aplikace zejména v implementaci metod popsaných v kapitole 4.

Převod barevného režimu obrazu	*cv::cvtColor*
Prahování	*cv::threshold*
Adaptivní prahování	*cv::adaptiveThreshold*
Optický tok (metoda [7])	*cv::calcOpticalFlowPyrLK*
Optický tok (metoda [6])	*cv::calcOpticalFlowFarneback*
Algoritmus PPHT	*cv::houghLinesP*
Morfologické operace	*cv::morphologyEx*
Hledání kontur	*cv::findContours*
Cannyho detekce hran	*cv::Canny*

Tabulka č. 5.1: Použité metody implementované v knihovně OpenCV.

Příklady použití těchto funkcí jsou uvedeny v [4]. Dokumentace a syntaxe těchto funkcí je uvedena v [10].

5.3 Vytvořené objektové třídy a funkce

Při implementaci metody pro automatickou detekci meteorů byly z velké části využity funkce, které jsou uvedeny v tabulce 5.1. Pro uložení postupu metody byla vytvořena objektová třída *MetDetector*. Dále byly naprogramovány další funkce, které nejsou součástí knihovny OpenCV a které implementují metody maskování z kapitoly 5.2 a funkce zpracování obrazu, které nejsou součástí OpenCV. Pro zobrazení výsledků detekce byla vytvořena objektová třída *MetView*.

Objektová třída MetDetector

Tato objektová třída slouží pro načtení sekvence tří snímků $\mathbf{I_c}(t\text{-}1)$, $\mathbf{I_c}(t)$, $\mathbf{I_c}(t+1)$ a provedení detekce na snímku $\mathbf{I_c}(t)$. Ukládá všechny provedené kroky detekce do nových matic. Tento postup je možné v aplikace procházet a ladit jednotlivé parametry.

Objektová třída MetView

Slouží pro načtení obrázku a údajů o oblastech, ve kterých byl nalezen meteor. Tyto nálezy jsou do obrázku zakresleny.

Označování oblastí (labeling)

Provádí označení souvislých oblastí. To je řešeno použitím funkce z knihovny OpenCV *cv::findContours*, která postupuje podle algoritmu popsaného v [5]. Nalezené kontury jsou dále vykreslovány (vyplněné) a označené příslušným číslem od 1 do N, kde N je počet samostatných oblastí. Návratová hodnota funkce je počet nalezených oblastí.

Odstraňování malých oblastí

Tato funkce odstraní z binárního obrazu „malé“ oblasti. K tomu využívá označování oblastí popsané výše. Nejprve jsou označeny souvislé oblasti. Pro každou oblast je potom vypočítán počet bodů, a pokud je tento počet menší než zvolený práh, oblast bude odstraněna.

Hlavičky a parametry uvedených funkcí a objektových tříd a další naprogramované funkce jsou uvedeny v příloze A.

5.4 Knihovny a vývojové prostředí Qt

Qt je kompletní integrované vývojové prostředí (IDE) pro jazyk C++ původně vyvinuté norskou softwarovou společností Trolltech, kterou koupila v roce 2008 společnost Nokia. Je poskytované jak pod LPGL open source licencí, tak i pod komerční licencí. Skládá se ze dvou samostatných celků. První je multiplatformní vývojové prostředí Qt Creator a druhý je sada knihoven objektových tříd vývojových nástrojů Qt libraries. Použití Qt Softwaru pro vytváření C++ aplikací má řadu výhod:

- Qt je open source, dává přístup do zdrojových kódů různých Qt komponent.
- Je multiplatformní, to znamená, že v Qt můžeme vytvářet aplikace pro různé operační systémy.
- Obsahuje kompletní multiplatformní GUI knihovnu (pro grafické uživatelské rozhraní), založenou na objektově orientovaném programování a událostmi řízeným modelem. [4]

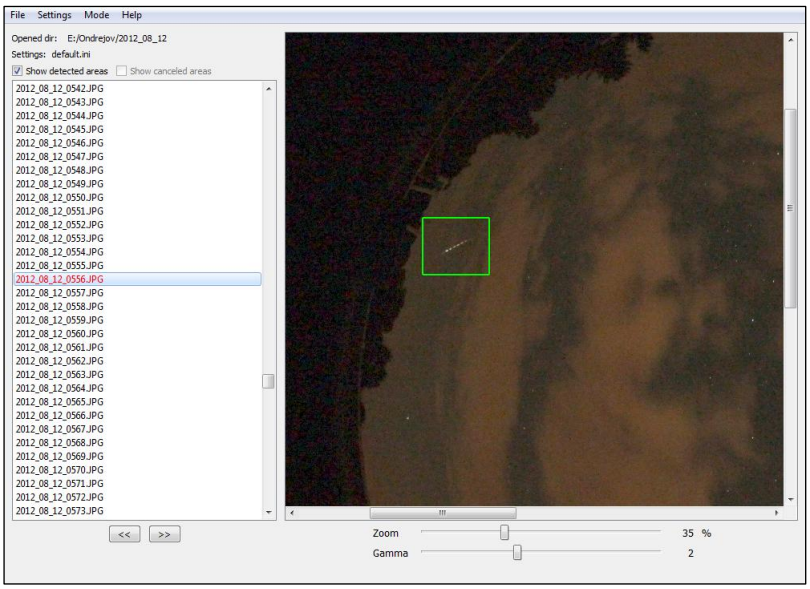
5.5 Uživatelské rozhraní

Uživatelské rozhrani (GUI – Graphics User Interface) je vytvořeno ve vývojovém prostředí Qt Creator s využitím knihoven Qt Libraries. Tyto knihovny obsahují velké množství komponent pro snadné vytvoření grafické aplikace.

Aplikace se skládá ze dvou hlavních částí. Je to dávkový (aplikační) mód a ladící mód. Mezi těmito dvěma módy lze v aplikaci přepínat.

5.5.1 Dávkový mód

Tento aplikační mód slouží k běžnému použití vyvinuté aplikace. Umožňuje načíst dávku snímků (např. celou nafocenou noc nebo její část) a na nich spustit detekci meteorů. Výsledek je ukládán do souboru a lze jej kdykoliv načíst a prohlížet výsledky. Snímky je také možné v aplikaci pouze prohlížet a používat ji tedy jako standardní prohlížeč obrázků.



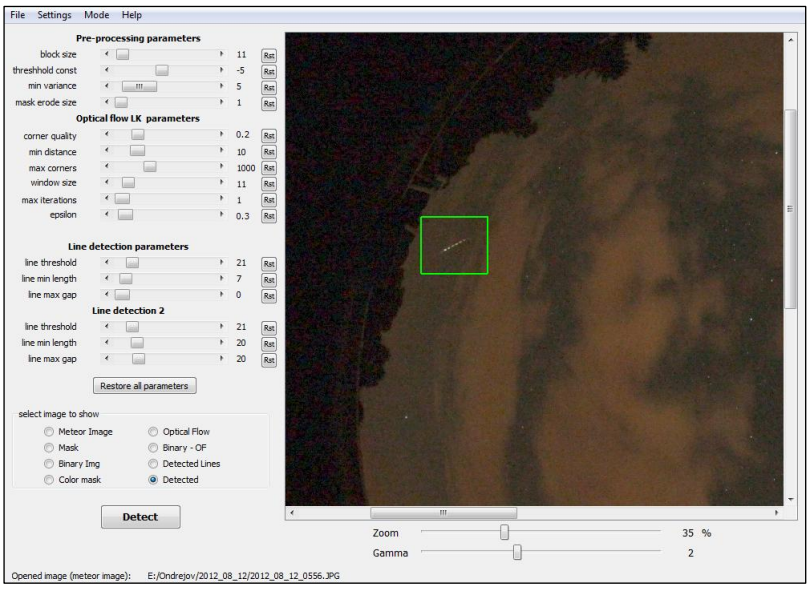
Obrázek č. 5.1: Náhled aplikace v dávkovém módu.

Součástí dávkového módu je možnost ručně měnit získané výsledky. Nalezenou oblast je možné zrušit a naopak lze oblast přidat tam, kde nebyla nalezena. Tato vlastnost může být dále využita pro vytvoření trénovací množiny meteorů.

5.5.2 Ladící mód

V tomto módu můžeme načíst pouze jeden snímek. Automaticky (pokud jsou soubory pojmenovány stylem **název**_*xxxx*.**JPG**, kde *xxxx* značí číslo vyfoceného snímku) jsou načteny předchozí a následující snímek a je vytvořena instance třídy *MetDetector*.

Dále tento mód umožňuje „ladit“ parametry jednotlivých funkcí a sledovat průběh metody detekce.



Obrázek č. 5.2: Náhled aplikace v ladícím módu.

5.6 Výstup aplikace

Aplikace ukládá výsledky detekce do souboru, jehož název a umístění volí uživatel při spuštění detekce. Do výstupního souboru jsou ukládány pouze souřadnice nalezených meteorů.

Struktura výstupního souboru může vypadat například takto:

```
[%General]                              // hlavní sekce
directory=G:/Ondrejov/2012_08_12        // umístění souborů na disku
detectedFiles=1                         // počet snímků s nálezem
file_1=2012_08_12_0401.JPG              // seznam snímků s nálezy

[FILELIST]                              // sekce souborů
Files=5                                 // počet zkoumaných souborů
file_0=2012_08_12_0399.JPG              //
file_1=2012_08_12_0400.JPG              //
file_2=2012_08_12_0401.JPG              // názvy souborů
file_3=2012_08_12_0402.JPG              //
file_4=2012_08_12_0403.JPG              //

[2012_08_12_0401.JPG]                   // sekce určitého souboru (s nálezem)
areas=1                                 // počet nálezů
0_x1=2454                               // souřadnice x prvního bodu obdélníku
0_x2=2797                               // souřadnice x druhého bodu obdélníku
0_y1=1425                               // souřadnice y prvního bodu obdélníku
0_y2=1705                               // souřadnice y druhého bodu obdélníku
```

5.7 Paralelizace funkcí

Při průběhu detekce je využito pouze jedno procesorové jádro. V současnosti jsou už téměř všechny procesory v osobních počítačích a dokonce i v mobilních telefonech více jádrové. Provedením paralelizace můžeme výpočet jednotlivých metod rozdělit na více jader procesoru a detekci tak značně urychlit.

Ve funkcích použitých pro detekci se zpracovává celý obraz najednou. To ale není podmínka pro správnou funkčnost těchto funkcí. Obraz tedy rozdělíme na více částí podle počtu jader procesoru a každou část zpracuje jedno jádro, po dokončení výpočtu jsou opět všechny části spojeny do jednoho obrazu. Aby nedošlo ke ztrátě informace v místech rozdělení obrazu, rozdělení bude provedeno s nenulovým přesahem.

Protože programování paralelních aplikací je značně složité, byla pro něj využita funkce *QtConcurrent::run*, která je implementována v knihovnách Qt. Tato funkce spustí požadovanou funkci v novém vlákně. Syntaxe a příklad funkce *QtConcurrent::run* je uvedena v [11].

6 Experimenty

V této kapitole budou provedeny experimenty s laděním parametrů jednotlivých funkcí. Nejprve budou stanoveny doporučené rozsahy parametrů z analýzy trénovací množiny. Dále budou nastaveny konfigurace parametrů, se kterými budou provedeny experimenty na testovacích datech.

6.1 Trénovací a testovací množina

Nastavování parametrů jednotlivých funkcí bude provedeno na předem vybrané trénovací množině. Z akademického institutu v Ondřejově bylo poskytnuto několik různých záznamů nocí. Jako trénovací množina jsou použity snímky ze záznamu jedné noci, na kterých byl zachycen meteor. Z důvodu objemu dat bude vyhodnocovaná testovací množina složena z dvou zaznamenaných nocí. Data z první noci jsou dále označena jako *sada1* a data z druhé noci jako *sada 2*.

6.2 Stanovení doporučených parametrů pro detekci

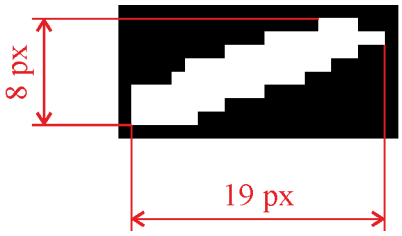
Na základě analýzy zachycených meteorů z trénovací množiny a následných experimentů budou stanoveny doporučené parametry pro jednotlivé funkce. Tyto parametry budou otestovány experimentem na testovacích datech.

Na následujícím obrázku je zobrazena nejčastější podoba zachyceného meteoru.

(a) (b) (c)

Obrázek č. 6.1: Typický tvar meteoru. (a) Originální snímek. (b) Vyšší jas. (c) Binární snímek (jednotlivé segmenty meteoru).

Meteor na obrázku 6.1 je složený ze 4 patrnějších segmentů. Dále je prozkoumán jeden z těchto segmentů.



Obrázek č. 6.2: Segment meteoru z obrázku 6.1

Délku segmentu na obrázku 6.2 určíme z Pythagorovy věty.

$$l = \sqrt{19^2 + 8^2} = 20{,}6 \cong 21. \tag{6.1}$$

6.2.1 Parametry adaptivního prahování

Pro modifikované adaptivní prahování musíme určit velikost oblastí, na které bude rozdělen vstupní obraz a na nichž bude provedeno prahování. Dále musíme určit prahovou konstantu a podmínku prahování. To je prováděno podle rovnice (3.1), kde práh je vypočítán pomocí vztahu (3.10).

Minimální velikost oblasti pro prahování by měla být minimálně rovna velikosti jednoho segmentu. Vlivem rozdělování obrazu může tato oblast obsahovat pouze část segmentu. Z toho důvodu je do funkce pro modifikované adaptivní prahování přidána hrana (rámeček) této oblasti. Tento rámeček by měl být stejně velký jako daná oblast, aby nedošlo k ořezu segmentu při prahování vlivem rozdělování obrazu.

Konstanta prahování a podmínka prahování se odvíjí od jasu a šumu na snímcích a jejich hodnota je zvolena experimentálně.

Po provedení experimentů byly zvoleny následující rozsahy pro jednotlivé hodnoty.

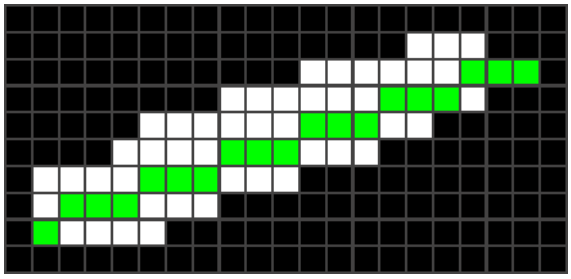
1) *blokSize* = **9 ~ 13** (velikost bloku bez rámečku – s rámečkem je tato hodnota trojnásobná 27 ~ 39)
2) *thresholdConst* = **-10 ~ -4** (prahová konstanta)
3) *varConst* = **5 ~ 10** (minimální rozptyl jasových hodnot)

6.2.2 Parametry PPHT

Pro progresivní pravděpodobnostní Houghovo transformaci volíme 6 parametrů (PPHT se provádí dvakrát, pro každou volíme 3 parametry).

Parametry první PPHT

Tyto parametry jsou voleny tak, aby byly nalezeny jednotlivé segmenty meteoru. Parametr *threshold* určuje, kolik hlasů musí dostat přímka, aby byla brána v úvahu (kolik bodů ji musí tvořit). Na obrázku 6.3 je naznačeno, jaké body ze segmentu mohou tvořit přímku. V tomto segmentu je konkrétně těchto bodů 19. Protože HT je málo citlivá na nepřesnosti přímek, bude pro tuto přímku ve skutečnosti o něco více hlasů.



Obrázek č. 6.3: Ukázka bodů segmentu, které tvoří přímku

Délka tohoto segmentu byla vypočítána vztahem (6.1). V experimentech bylo zjištěno, že pro správnou detekci segmentů musí být tato hodnota přibližně o polovinu nižší než vypočítaná hodnota.

Z analýzy dat a z předpokladů uvedených výše a z provedených experimentů byly stanoveny doporučené hodnoty pro první PPHT.

1) *threshold* = **16 ~ 24** (počet bodů segmentu tvořící přímku)
2) *minLength* = **6 ~ 15** (min délka přímky tvořící jeden segment)
3) *maxGap* = **0** (v segmentu nejsou povoleny mezery)

Parametry druhé PPHT

Na základě meteorů z trénovací množiny předpokládejme, že meteor bude složen nejméně ze tří segmentů. Mezera mezi jednotlivými segmenty je maximálně stejně

dlouhá jako segment. Dále připustím možnost, že jeden segment může vlivem prahování chybět (mezera může být větší).

Parametry druhé PPHT stanovým na následující hodnoty.

1) *threshold* = **18 ~ 30**
2) *minLength* = **25 ~ 40**
3) *maxGap* = **18 ~ 22**

6.2.3 Parametry Optického toku

Pro výpočet optického toku podle [7] s využitím funkce *cv::calcOpticalFlowPyrLK* byly na základě experimentů zvoleny tyto rozsahy.

1) *cornerQuality* = **0,1 ~ 0,3** (kvalita zjištěného rohu – významového bodu VB)
2) *minDistance* = **8 ~ 15** (minimální vzdálenost mezi VB)
3) *maxCorners* = **1000 ~ 3000** (maximální počet VB)
4) *windowSize* = **8 ~ 12** (velikost okna)
5) *maxIterations* = **1** (počet iterací)

Další parametry funkce *cv::calcOpticalFlowPyrLK* jsou nastaveny na výchozí hodnoty podle příkladu použití v [10].

6.3 Konfigurace parametrů

Nyní bude uvedeno několik konfigurací parametrů, které budou testovány na testovacích datech.

6.3.1 Přísná konfigurace

Nalezne spolehlivě meteory obvyklého typu zobrazeného na obrázku 6.1. Protože velká většina zachycených meteorů má tento tvar, poskytuje tato konfigurace dobré výsledky.

Parametry AT		Parametry PPHT 1	
blockSize	13	threshold	20
thresholdConst	-6	minLength	7
varConst	5	maxGap	0
Parametry OF		**Parametry PPHT 2**	
cornerQuality	0,2	threshold	21
minDistance	10	minLength	35
maxCorners	1000	maxGap	21
windowSize	11		
maxIterations	1		

Tabulka č. 6.1: Přísná konfigurace parametrů.

6.3.2 Citlivá konfigurace

Nalezne spolehlivě meteory obvyklého typu zobrazeného na obrázku 6.1. Navíc oproti přísné konfiguraci nalezne i některé slabší a kratší meteory, které nemají klasický segmentový tvar. Vzhledem k citlivějším parametrům očekáváme větší množství falešných detekcí.

Parametry AT		Parametry PPHT 1	
blockSize	11	threshold	21
thresholdConst	-5	minLength	7
varConst	5	maxGap	0
Parametry OF		**Parametry PPHT 2**	
cornerQuality	0,2	threshold	21
minDistance	10	minLength	20
maxCorners	1000	maxGap	20
windowSize	11		
maxIterations	1		

Tabulka č. 6.2: Citlivá konfigurace parametrů.

7 Získané výsledky

V této kapitole jsou uvedeny výsledky testování vyvinuté metody, která byla popsána v 4.4. Ta je testována s parametry uvedenými v kapitole 6. Dále jsou tyto výsledky porovnány s metodou popsanou v 4.3.

Pro vyhodnocení testů je použito vyčíslení přesnosti a pokrytí (precision and recall). Úspěšnost metody vyjádříme jako pokrytí v procentech. Tyto hodnoty určíme z provedených testů a z apriorní informace o zaznamenaných meteorech. Pro výpočet jsou využity následující vztahy:

$$přesnost = \frac{počet\ detekovaných\ meteorů}{počet\ všech\ detekovaných\ objektů} \tag{7.1}$$

$$pokrytí = \frac{počet\ detekovaných\ meteorů}{počet\ všech\ zaznamenaných\ meteorů} \tag{7.2}$$

$$úspěšnost\ [\%] = pokrytí \cdot 100 \tag{7.3}$$

7.1 Výsledky metody 4.3

NEFILTROVANÉ VÝSLEDKY		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	7	8
celkový počet detekovaných objektů	522	1106
přesnost detekce	0,01	0,01
úspěšnost detekce	43,75%	57,14%

BEZPEČNÁ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	7	8
celkový počet detekovaných objektů	113	290
přesnost detekce	0,06	0,03
úspěšnost detekce	43,75%	57,14%

AGRESIVNÍ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	7	8
celkový počet detekovaných objektů	73	170
přesnost detekce	0,10	0,05
úspěšnost detekce	43,75%	57,14%

Tabulka č. 7.1: Výsledky metody 4.3.

7.2 Výsledky metody 4.4

7.2.1 Přísná konfigurace

NEFILTROVANÉ VÝSLEDKY		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	9	8
celkový počet detekovaných objektů	264	98
přesnost detekce	0,03	0,08
úspěšnost detekce	56,25%	57,14%

BEZPEČNÁ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	9	8
celkový počet detekovaných objektů	84	64
přesnost detekce	0,11	0,13
úspěšnost detekce	56,25%	57,14%

AGRESIVNÍ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	9	8
celkový počet detekovaných objektů	56	27
přesnost detekce	0,16	0,30
úspěšnost detekce	56,25%	57,14%

Tabulka č. 7.2: Výsledky metody 4.4 – přísná konfigurace.

7.2.2 Citlivá konfigurace

NEFILTROVANÉ VÝSLEDKY		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	11	12
celkový počet detekovaných objektů	601	154
přesnost detekce	0,02	0,08
úspěšnost detekce	68,75%	85,71%

BEZPEČNÁ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	11	12
celkový počet detekovaných objektů	202	94
přesnost detekce	0,05	0,13
úspěšnost detekce	68,75%	85,71%

AGRESIVNÍ FILTRACE		
	Sada 1	**Sada 2**
počet zachycených meteorů	16	14
počet detekovaných meteorů	11	12
celkový počet detekovaných objektů	132	47
přesnost detekce	0,08	0,26
úspěšnost detekce	68,75%	85,71%

Tabulka č. 7.3: Výsledky metody 4.4 – citlivá konfigurace.

7.3 Srovnání výsledků

		Metoda 4.3		Metoda 4.4			
		Sada 1	Sada 2	přísná konfigurace		citlivá konfigurace	
				Sada 1	Sada 2	Sada 1	Sada 2
přesnost	**nefiltrováno**	0,01	0,01	0,03	0,08	0,02	0,08
	bezpečná filtrace	0,06	0,03	0,11	0,13	0,05	0,13
	agresivní filtrace	0,10	0,05	0,16	0,30	0,08	0,26
úspěšnost	**nefiltrováno**	43,75%	57,14%	56,25%	57,14%	68,75%	85,71%
	bezpečná filtrace	43,75%	57,14%	56,25%	57,14%	68,75%	85,71%
	agresivní filtrace	43,75%	57,14%	56,25%	57,14%	68,75%	85,71%

Tabulka č. 7.4: Srovnání výsledků.

Z provedených testů nejlépe vyšla citlivá konfigurace metody 4.4, která měla sice o něco nižší přesnost než přísná konfigurace této metody, ale zato mnohem vyšší úspěšnost detekce. Přesnost s využitím filtrace dosáhla přibližně jedné čtvrtiny, což znamená, že každý čtvrtý nález byl opravdu meteor. Úspěšnost na testovacích datech v sadě 2 dosáhla 85,71%.

Použití obou filtrací nemělo vliv na úspěšnost detekce. Meteory, které se nepodařilo detekovat, jsou velmi slabé nebo krátké a jsou špatně rozeznatelné i při ručním prohlížení. Těchto meteorů je více v sadě 1, proto je zde úspěšnost detekce nižší. Zřetelnější meteory se podařilo nalézt všechny.

Detekované objekty, které nebyly meteory, jsou nejčastěji mraky, které nebyly odstraněny maskováním. Jejich velká většina je odstraněna při filtraci výsledků detekce. Objekty, které zůstanou i po filtraci jsou nejčastěji záblesky družic pro satelitní komunikační síť Iridium Satellite LLC. Tyto záblesky mají na snímku velmi podobný charakter jako meteor hořící v atmosféře.

8 Závěr

V předkládané diplomové práci jsme se zabývali metodami počítačového vidění a zpracováním digitalizovaného obrazu. Úvodem do reprezentace digitálního obrazu v počítači a do obrazových dat, která byla použita pro testování, se zabývá kapitola 2. Metody počítačového vidění, jež jsou použity pro návrh metody automatické detekce meteorů, byly popsány v kapitole 3.

V kapitole 4 byla navržena metoda pro automatické vyhledávání meteorů na snímcích noční oblohy. Dále byla programově implementována v uživatelské aplikaci a následně testována na poskytnutých datech. Součástí testování aplikace bylo naladění vstupních parametrů jednotlivých metod pro co nejlepší výsledek detekce. Volba těchto parametrů byla diskutována v kapitole 6. Při testech došlo k potvrzení předpokladu, že pro vyšší citlivost metody, přibývá falešných detekcí.

V tabulce č. 7.4 jsou uvedeny dosažené výsledky při použití zavedené metody detekce přímek s využitím Cannyho detektoru hran a dále výsledky získané nově navrženou metodou se dvěma různými konfiguracemi. V testech byla ověřena funkčnost nově navržené metody, která poskytla lepší výsledky než zavedená metoda pro hledání přímek v obraze.

Vytvořená aplikace byla poskytnuta astronomickému ústavu, kde je nyní testována a může sloužit k orientačnímu hledání meteorů. Dokáže úspěšně najít většinu zaznamenaných meteorů v rozumném čase. Další práce na této aplikaci by byla například na optimalizaci využití procesorového času a úspory operační paměti počítače. Takové úpravy jsou však už nad rámec této práce.

Při vytváření diplomové práce jsem se seznámil s metodami strojového vidění. Dále jsem se seznámil s knihovnami OpenCV a Qt, které byly využity pro vytvoření uživatelské aplikace. V té je implementovaná metoda pro automatickou detekci meteorů. Tuto aplikaci jsem otestoval na poskytnutých datech a vyhodnotil její úspěšnost. Všechny cíle této práce se tedy podařilo splnit.

9 Použitá literatura

[1] L. Shapiro, G. Stockman, *Computer Vision*. New Yersey: Prentice Hall, 2001. ISBN 0-130-30796-3.

[2] M. Šonka, V. Hlaváč, R. Boyle, *Image Processing, Analysing and Machine Vision*. 3. vyd. Toronto: Thomson Learning, 2008. ISBN 0-495-08252-X.

[3] R. González, R. Woods, *Digital image processing*. 2.vyd., New Yersey: Prentice Hall, 2002. ISBN 0-201-18075-8.

[4] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Birmingham: Packt Publishing Ltd, 2011. ISBN 1-849-51325-2.

[5] S. Suzuki, K. Abe, Topological Structural Analysis of Digitized Binary Images by Border Following, *Computer Vision, Graphics, and Image Processing*, April 1985, vol. 30, s. 32-46. ISSN 0734-189X.

[6] G. Farnebäck, Two-Frame Motion Estimation Based on Polynomial Expansion. In *Image Analysis, 13th Scandinavian Conference, SCIA 2003, Halmstad, Sweden, June 29 - July 2, 2003*, Berlin: Springer, 2003, s. 363 – 370. (Lecture Notes in Computer Science; vol. 2749, ISBN 978-3-540-40601-3)

[7] J. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*. Intel Corporation Microprocessor Research Labs, 2000.

[8] C. Galamhos, J. Matas, J. Kitter, Progressive probabilistic Hough transform for line detection. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, Fort Collins, 1999, vol. 1, s. 560. ISSN 1063-6919.

[9] H. Schildt, *Nauč se sám C++*, Praha: SoftPress s.r.o., 2001. ISBN 80-86497-13-5.

[10] *OpenCV 2.4.6.0 documentation* [online]. Poslední aktualizace 1. 6. 2013. < http://docs.opencv.org/ >

[11] *Qt Project* [online]. < http://qt-project.org/doc/ >

Přílohy

Příloha A: Objektové třídy a funkce vytvořené při vyvíjení aplikace.

Objektová třída MetDetector

Atributy:

cv::Mat *imgPrevious*	snímek $I_c(t-1)$
cv::Mat *imgMeteor*	snímek $I_c(t)$
cv::Mat *imgNext*	snímek $I_c(t+1)$
cv::Mat *imgMask*	maska horizontu
cv::Mat *imgBin*	prahovaný snímek $I_b(t)$
cv::Mat *imgBin2*	prahovaný snímek $I_b(t+1)$
cv::Mat *imgOpticalFlow*	binární obraz optického toku I_{OF}
cv::Mat *imgBin_OF*	rozdílový obraz $I_b(t)$ - $I_b(t+1)$ - I_{OF}
cv::Mat *imgLines*	výsledek první detekce pomocí PPHT
cv::Mat *imgDetected*	celkový výsledek detekce pomocí PPHT
cv::vector<cv::Vec4i> *areas*	uložené souřadnice nálezů
OF_LK_parameters *opt_flow_lk_param*	parametry pro optický tok podle [7]
OF_FB_parameters *opt_flow_fb_param*	parametry pro optický tok podle [6]
PPHT_parameters *ppht1_param*	parametry první PPHT
PPHT_parameters *ppht2_param*	parametry druhé PPHT
AT_parameters *at_param*	parametry pro modifikovaný adaptivní práh
CANNY_parameters *canny_param*	parametry pro cannyho detekci hran
int *maskErodeSize*	hodnota pro velikost ořezu masky horizontu (eroze)
int *ofFBthreshold*	hodnota prahu pro získání binárního obrazu z amplitudy optického toku vypočítaného podle [6]

Metody:

void *preProcessingAT*(***int*** *coreNumber*)
Provede modifikované adaptivní prahování snímku $I_c(t)$. Výsledek je snímek $I_b(t)$.
coreNumber určije na kolik jader bude metoda paralelizována.

void *preProcessingCanny*()
Provede Cannyho detekci hran na snímku $I_c(t)$. Výsledek je snímek $I_b(t)$.

void *calcOF_FB*(***int*** *coreNumber*)
Vypočte optický tok podle metody popsané v [6].

void *calcOF_PK*(***int*** *coreNumber*)
Vypočte optický tok podle metody popsané v [7].

void *postProcessing*(***int*** *coreNumber*, ***bool*** *colorFilter*)
Odstraní malé oblasti, a odečte informaci o pohybu hvězd.
colorFilter = 1 – Maskuje mraky na základě barevné informace. (Metoda 5.2.2)

void *OF_FB_Threshold*()
Získání binárního obrazu z optického toku podle metody [6].

int *lineDetection*(***bool*** *testLines*, ***bool*** *testNeigh*)
Provede detekci úseček použitím dvou PPHT.

Třída dále obsahuje metody *set* a *get* pro získání hodnoty a nastavení hodnoty jednotlivých parametrů.

Objektová třída MetView

Atributy:

***cv::Mat** imgMeteor*	načtený snímek
***cv::Mat** imgDetected*	načtený snímek s vyznačenými nálezy
***cv::vector<cv::Vec4i>** areas*	souřadnice nálezů

Metody:

***void** addArea(**int** x1, **int** x2, **int** y1, **int** y2)*
Přidá souřadnice nálezu do *areas*.

Načte obrázek do *imgMeteor*.

***void** paintAreas()*
Vykreslí nálezy do *imgDetected*.

Metody set a get pro uložení a získání snímků:

***void** setImgMeteor(**cv::Mat** *img)*
***cv::Mat** getImgMeteor()*
***cv::Mat** getImgDetected()*

Metody maskování a zpracování obrazu

Maskování mraků
syntaxe:

```
void colorFiltr(cv::Mat src, cv::OutputArray dst);
```

Vytvoří masku mraků.

src – vstupní obraz
dst – výstupní obraz - maska

Maksování oblohy
syntaxe:

```
void getMask(cv::Mat src, cv::Mat* mask);
```

Vytvoří masku oblohy.

src – vstupní obraz
dst – výstupní obraz - maska

Modifikované adaptivní prahování
syntaxe:

```
void myAdaptiveTreshold(cv::Mat src, cv::OutputArray dst, int
blockSize, int edge, int varConst, int threshDist, cv::Mat
mask);
```

Provádí adaptivní prahování podle metody popsané v kapitole 3.1.3.

src – vstupní obraz
dst – výstupní obraz
blockSize – velikost prahované oblasti
edge – velikost přesahu (rámečku) prahované oblasti
varConst – podmínka prahování

threshDist – prahová konstanta
mask – maska horizontu

Označování oblastí (labelling)
syntaxe:

```
int bwlabel(cv::Mat src, cv::OutputArray dst);
```

Provádí označení souvislých oblastí. To je řešeno použitím funkce *cv::findContours*, která postupuje podle algoritmu popsaného v [5]. Nalezené kontury jsou dále vykreslovány (vyplněné) a označené příslušným číslem od 1 do N, kde N je počet samostatných oblastí. Návratová hodnota funkce je počet nalezených oblastí.

src – vstupní obraz
dst – výstupní obraz s označenými oblastmi

Odstraňování malých oblastí
syntaxe:

```
int removeObject(cv::Mat src, cv::OutputArray dst, int minSize,
int maxSize);
```

Tato funkce odstraní z binárního obrazu „malé“ oblasti. Využívá funkci *bwlabel*. Nejprve jsou označeny souvislé oblasti. Pro každou oblast je potom vypočítán počet bodů, a pokud je tento počet menší než zvolený práh, oblast bude odstraněna.

src – vstupní obraz
dst – výstupní obraz s označenými oblastmi
minSize – minimální velikost oblasti
maxSize – maximální velikost oblasti (není využita)

Příloha B: Seznam zaznamenaných meteorů ve vyhodnocených datech

SADA 1 - 12/13 VIII 2012			SADA 2 - 13/14 VIII 2012	
SNÍMEK	**METEOR**		**SNÍMEK**	**METEOR**
2012_08_12_0052	krátký, slabší, horizont		2012_08_13_0088	zřetelný
2012_08_12_0096	slabší		2012_08_13_0203	krátký, slabší
2012_08_12_0134	krátký jasný, horizont		2012_08_13_0222	zřetelný
2012_08_12_0138	nezřetelný		2012_08_13_0389	zřetelný
2012_08_12_0219	kratší, slabší, horizont		2012_08_13_0418	krátký, slabší
2012_08_12_0224	slabší		2012_08_13_0476	krátký, slabší
2012_08_12_0254	zřetelný		2012_08_13_0548	zřetelný
2012_08_12_0292	slabý, horizont		2012_08_13_0589	slabší
2012_08_12_0329	jasný		2012_08_13_0612	zřetelný
2012_08_12_0345	krátký , slabý		2012_08_13_0628	slabý
2012_08_12_0348	slabý, horizont		2012_08_13_0637	nepatrný
2012_08_12_0401	zřetelný		2012_08_13_0660	jasný
2012_08_12_0556	jasný		2012_08_13_0762	krátký, slabý
2012_08_12_0692	slabý		2012_08_13_0799	nepatrný
2012_08_12_0701	zřetelný			
2012_08_12_0556	zřetelný			

Příloha C:

Aplikace Meteor Detector - Uživatelská příručka

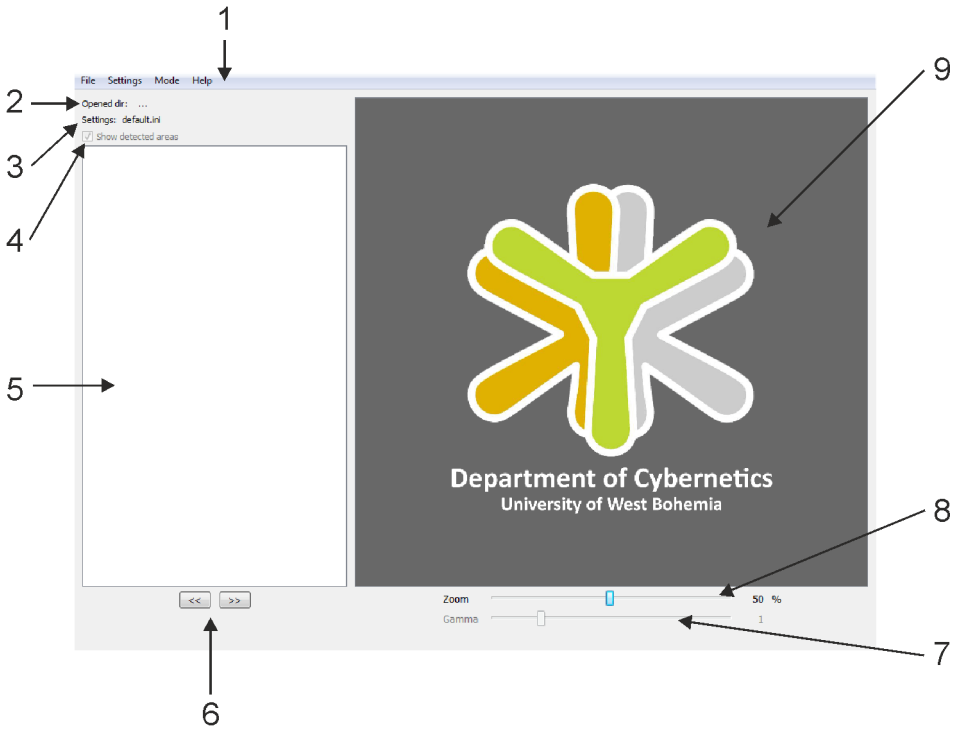
Aplikace Meteor detektor vznikla na katedře kybernetiky jako diplomová práce na závěr studia fakulty aplikovaných věd Západočeské univerzity v Plzni. Aplikace slouží k automatické detekci meteorů na snímcích noční oblohy. Dále je v aplikaci možné ladit parametry jednotlivých funkcí a provádět tak různé experimenty. Výsledky detekce jsou ukládány do souborů a je možné je později prohlížet.

1 Instalace

Instalace aplikace se spustí otevřením instalačního souboru. Instalační balíčky jsou vytvořeny pro MS Windows XP 32bit, MS Windows 7 32bit a MS Windows 7 64bit. Správná funkce aplikace na jiných verzích systému MS Windows není zaručena. Po otevření instalačního balíčku je spuštěn průvodce, který Vás provede celou instalací.

2 Po spuštění

Po spuštění aplikace se otevře grafické okno s logem katedry kybernetiky. Jednotlivé prvky uživatelského rozhraní jsou zobrazeny na následujícím obrázku a jsou popsány níže.



1. Menu
2. Otevřený adresář
3. Načtené parametry

4. Volba pro zobrazení detekovaných nálezů
5. Seznam pro listování načtených snímků
6. Přepínání mezi následujícím a předchozím snímkem s nálezem
7. Korekce gamy
8. Změna velikosti
9. Oblast pro zobrazování snímků

3 Menu

File

Záložka pro práci se soubory.

Open images

Načtení vstupních souborů. Po stisknutí se otevře dialog pro vybrání požadovaných souborů. Při detekci se první a poslední soubor nedetekují. Pro detekci jednoho snímku je totiž zapotřebí posloupnosti tří snímků, z nichž na prostředním probíhá detekce.

Start detection

Spustí detekci pro načtené soubory. Po skončení detekce je provedena filtrace a zobrazí se dialogové okno s dotazem, zdali se mají výsledky zobrazit.

OpenResults

Otevře výsedky dříve provedené detekce.

Results info

Zobrazí informace o výsledcích. (počet souborů, pro které byla detekce provedena, počet souborů, ve kterých byl nález a počet všech nálezů)

Results filtration

Spustí filtraci výsledků

Type 1

Bezpečná filtrace. Pokud jsou nálezy na třech snímcích po sobě ve stejném okolí, odstraní je.

Type 2

Agresivní filtrace. Pokud jsou nálezy na dvou snímcích po sobě ve stejném okolí, odstraní je.

Exit

Ukončí aplikaci

Settings

Záložka pro nastavování parametrů jednotlivých funkcí a jejich uložení.

Pre-proccessing method

Nastavení metody pro získání binárního obrazu. Možné volby jsou adaptivní prahování nebo cannyho detekce hran (experimentální).

Optical flow method

Nastavení metody optického toku, kterou je získávána informace o pohybu hvězd. Možné volby jsou *Lucas Kanade* nebo *Farneback*(experimentální). Lucas Kanade počítá optický tok pouze v detekovaných zájmových bodech – rychlejší výpočet. Farneback počítá denzitní optický tok celého obrazu – pomalejší výpočet.

Line detector

Nastavení detektoru.

Use color mask

Použije barevnou masku pro nalezení mraků. Tyto oblasti jsou pak při detekci úseček ignorovány.

Line test, Neighbourhood test

Experimentální klasifikace nalezených oblastí. *Line test* zkoumá tvar segmentu po první detekci úsečky. *Neighbourhood test* zkoumá tvar konvexního obalu objektů v okolí nálezu. Tyto metody můžou způsobit ztrátu detekovaného meteoru, proto jsou standardně vypnuty. Slouží k budoucímu vývoji klasifikace nálezu.

Load settings, Save settings

Načtení a uložení nastavených parametrů funkcí.

Set factory settings

Načtení výchozích parametrů. Možné volby jsou *Strict, Sensitive, Insensitive*. *Strict* jsou přísné parametry pro detekci spíše jasnějších meteorů a minimum falešných detekcí. *Sensitive* jsou citlivé parametry. Naleznou většinu meteorů. *Insensitive* jsou méně citlivé parametry. Jsou vhodné pro data pořízená bez elektronické clony.

Auto detection

Pokud je tato možnost zaškrtnuta, provádí se detekce v ladícím módu ihned po změně některého z parametrů. Pokud zaškrtnutá není, detekce je provedena až po stisknutí tlačítka *Detect*.

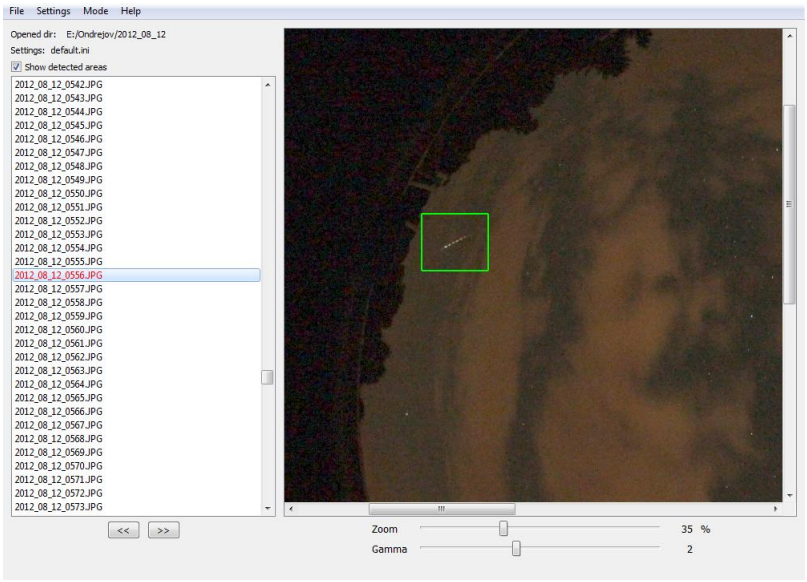
Mode

Slouží pro přepínání mezi dávkovým módem a ladícím módem.

4 Dávkový mód

Dávkový mód je standardní uživatelský mód. Slouží pro načtení dávky souborů (snímků noční oblohy) *Menu → File → Open images*. Seznam načtených snímků je zobrazen v levé části aplikace. Aktuální snímek je zobrazen v pravé části aplikace. Pro načtené snímky je možné spustit detekci *Menu → File → Start detection*.

Po provedení detekce se výsledky automaticky filtrují bezpečnou filtrací a následně jsou načteny. V souborech, které jsou v seznamu označeny červeně, byl nalezen objekt (meteor nebo falešná detekce).



Ruční úpravy

Při prohlížení výsledků je v aplikaci možnost odstranit chybný nález nebo naopak přidat nález tam, kde nebyl zaznamenán. Mazání nebo přidání nálezu se provádí přes kontextové menu vyvolané kliknutím pravého tlačítka myši do snímku.

Po přidání nového nálezu je ještě nutné ho uložit. To je opět řešeno přes kontextové menu.

5 Ladící mód

V tomto módu je možné provádět experimenty s laděním parametrů. Tyto parametry je následně možné uložit *Menu → Settings → Save settings* a následně je použít v dávkovém módu aplikace.

U jednotlivých parametrů jsou tlačítka *Rst*, která nastaví daný parametr na hodnotu, kterou měl při spuštění programu, nebo při načtení parametrů ze souboru.

Pre-processing parameters:

Parametry pro získání binárního obrazu ze vstupního obrazu.

block size

Velikost bloku pro provedení adaptivního prahování.
Doporučení hodnota **9 ~ 13.**

threshold const

Konstanta odečtená od střední hodnoty jasu bloku.
Doporučená hodnota **-10 ~ -4.**

min variance

Minimální rozptyl jasu bloku.
Doporučená hodnota **5 ~ 10**

Optical flow parameters:

Parametry pro výpočet optického toku. Výsledek je informace o pohybu hvězd.

corner quality

Koeficient kvality pro detekci zájmových bodů.
Doporučená hodnota **0,1 ~ 0,3**

min distance

Minimální vzdálenost dvou zájmových bodů.
Doporučená hodnota **8 ~ 15**

max corners

Maximální počet zájmových bodů
Doporučená hodnota **1000 ~ 3000**

window size

Velikost okna
Doporučená hodnota **8 ~ 12**

max iterations

Počet iterací algoritmu
Doporučená hodnota **1**

line detection parameters:

Parametry pro první detekci úseček.

line threshold

Počet bodů tvořící přímku.
Doporučená hodnota **16 ~ 24.**

line min length

Minimální délka úsečky.
Doporučená hodnota **6 ~ 15.**

line max gap

Maximální mezera v detekované úsečce
Doporučená hodnota **0.**

line detection 2:

Parametry pro druhou detekci úseček.

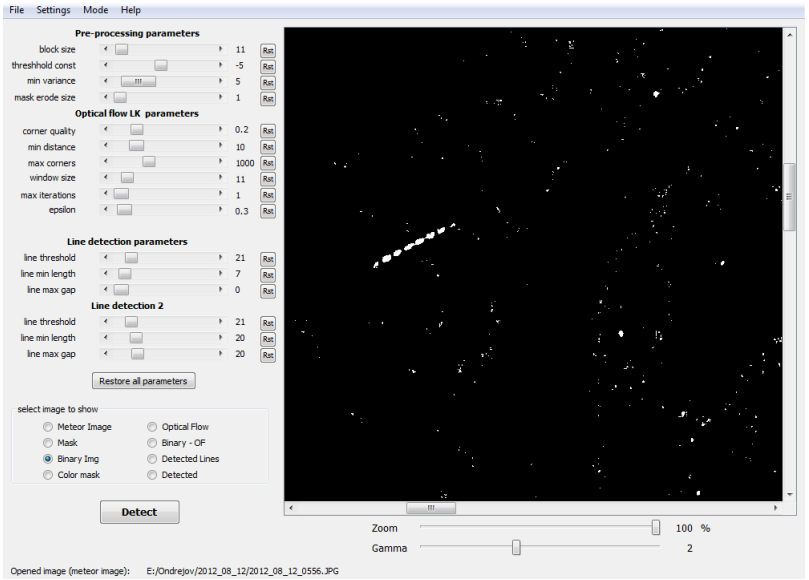
line threshold

Doporučená hodnota **18 ~ 30.**

line min length
Doporučená hodnota **25 ~ 40.**
line max gap
Doporučená hodnota **18 ~ 22.**

Dále je v ladícím módu možnost přepínání průběhu detekce:

Meteor Image
Vstupní snímek.
Mask
Maska oblohy.
Binary Image
Binární obraz získaný adaptivním prahováním.
Color Mask
Barevná maska pro detekci mraků.
Optical Flow
Optický tok (pohyb hvězd).
Binary – OF
Redukovaný binární obraz. Jsou od něho odečteny hvězdy a maska mraků. Dále jsou z něj odstraněny malé objekty (menší než segment meteoru).
Detected Lines
Detekované segmenty meteoru. První detekce úseček.
Detected
Výsledek detekce.



Další volitelné parametry jiných konfigurací detekce nejsou uvedeny. V aplikaci jsou obsaženy pro experimentální použití. Jsou to parametry Cannyho detektoru a optického toku metodou Farneback.

6 Výsledky

Výsledky jsou ukládány do speciálního souboru s příponou md. Umístění výsledků je možné zvolit při spuštění detekce. Mimo zvoleného názvu se vytvoří ještě druhý soubor, ve kterém je provedena bezpečná filtrace. Tento soubor je načítán po skončení detekce.

Struktura výstupního souboru:

```
[%General]                          // hlavní sekce
directory=G:/Ondrejov/2012_08_12    // umístění souborů na disku
detectedFiles=1                     // počet snímků s nálezem
file_1=2012_08_12_0401.JPG          // seznam snímků s nálezy

[FILELIST]                          // sekce souborů
Files=5                             // počet zkoumaných souborů
file_0=2012_08_12_0399.JPG          //
file_1=2012_08_12_0400.JPG          //
file_2=2012_08_12_0401.JPG          // názvy souborů
file_3=2012_08_12_0402.JPG          //
file_4=2012_08_12_0403.JPG          //

[2012_08_12_0401.JPG]               // sekce určitého souboru (s nálezem)
areas=1                             // počet nálezů
0_x1=2454                           // souřadnice x prvního bodu obdélníku
0_x2=2797                           // souřadnice x druhého bodu obdélníku
0_y1=1425                           // souřadnice y prvního bodu obdélníku
0_y2=1705                           // souřadnice y druhého bodu obdélníku
```