Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Detekcia textových dopravných značiek

Bakalárska práca

2019 Michal Zrubec

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Detekcia textových dopravných značiek

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Zuzana Černeková, PhD.

Bratislava, 2019 Michal Zrubec

53712399

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Zrubec
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Detekcia textových dopravných značiek
Detection of informational traffic signs

Anotácia: Z videa zosnímaného mobilom umiestneným na palubnej doske automobilu, vedieť detegovať textové dopravné značky.

Cieľ: Z videa zosnímaného mobilom umiestneným na palubnej doske automobilu, vedieť detegovať textové dopravné značky.

Vedúci: RNDr. Zuzana Černeková, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 27.09.2018

Dátum schválenia: 03.10.2018 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie: Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedenej literatúry.

V Bratislave dňa: .....

.....
Michal Zrubec

Poďakovanie: Ďakujem svojej školiteľke RNDr. Zuzane Černekovej, PhD. za poskytnutý čas, materiál, cenné rady a pomoc pri riešení tejto práce.

Abstrakt

Cieľom tejto bakalárskej práce je detekcia textových dopravných značiek na videu. V prvej kapitole je popísaná história a rozdelenie dopravných značiek. V druhej kapitole sú podrobnejšie popísané farebné modely a spracovanie obrazu. V tretej kapitole je popísaný návrh riešenia problému a návrh aplikácie. V nadväznosti na návrh je popísaná samotná implementácia konkrétnej aplikácie na detekciu textových dopravných značiek a v poslednej kapitole analýza výsledkov.

Kľúčové slová: spracovanie obrazu, video, dopravné značky, text

Abstract

The aim of this bachelor thesis is detection of text traffic signs on video. The first chapter describes the history and distribution of traffic signs. The second chapter describes color models and image processing in more detail. In the third chapter there is a description of the problem solution and the application design. Following the design, the actual implementation of a specific application for the detection of text traffic signs is described. In the last chapter the analysis of results is described.

Keywords: image processing, video, traffic signs, text traffic signs

Obsah

Úvod **1**

1 Prehľad problematiky **2**
1.1 Dopravné značky . . . 2
1.1.1 História dopravných značiek . . . 2
1.1.2 Rozdelenie dopravných značiek . . . 2
1.1.3 Rozdielnosť dopravných značiek v rôznych štátoch . . . 3
1.2 Podobné práce . . . 4
1.2.1 Rozpoznávanie dopravných značiek . . . 4
1.2.2 Detekcia a rozpoznávanie dopravných značiek . . . 4
1.2.3 Rozpoznávanie dopravných značiek a ich použitie v mapových aplikáciách . . . 5

2 Teoretická časť **7**
2.1 Farebné modely a priestory . . . 7
2.1.1 CIE L*a*b* model . . . 7
2.1.2 Lineárny model RGB . . . 8
2.1.3 HSV a HSL model . . . 9
2.2 Snímanie obrazu . . . 9
2.2.1 Rozlíšenie vstupného obrazu . . . 10
2.2.2 Problémy pri získavaní obrazu . . . 10
2.3 Predspracovanie obrazu . . . 11
2.3.1 Vyhladzovanie obrazu (filtrácia) . . . 11
2.3.2 Morfologické operácie . . . 12
2.4 Segmentácia . . . 13

3 Návrh riešenia problematiky **15**
3.1 Detekcia kandidáta na textovú dopravnú značku . . . 15
3.2 Detekcia textu na kandidátovi . . . 18
3.3 Užívateľské prostredie . . . 19

4 Riešenie problematiky a implementácia **20**
4.1 Použité nástroje 20
4.2 Získavanie obrazu 20
4.3 Postup pri implementácii 21
4.3.1 Spracovanie snímky 22
4.3.2 Používateľské prostredie 26

5 Vyhodnotenie a výsledky **29**
5.1 Čas vykonávania metód 30
5.2 Hľadanie kandidáta 30
5.3 Nájdenie textu na kandidátovi 32
5.4 Celková úspešnosť detekcie textových dopravných značiek 32

Záver **34**

Zoznam obrázkov

1.1 Dopravné značky vo svete 3

2.1 Cie L*a*b* model 8
2.2 Lineárny RGB model 8
2.3 HSV a HSL 9
2.4 Obraz zosnímaný kamerou 10
2.5 Rozlíšenia 10
2.6 Dilatácia a Erózia 12
2.7 Morfologická rekonštrukcia 13

3.1 Návrh postupu k nájdeniu kandidáta 15
3.2 Predspracovanie obrázku 16
3.3 Nájdenie kandidáta 17
3.4 Návrh postupu k nájdeniu textu na kandidátovi 18
3.5 Návrh postupu k nájdeniu textu 19

4.1 Rôzne počasie počas natáčania 21
4.2 Predspracovanie obrázku pri implementácii 23
4.3 Príprava na hľadanie kandidáta 24
4.4 Hľadanie bieleho textu - problém 25
4.5 Používateľské rozhranie 26
4.6 Dialógové okno pre zvolenie videa 27
4.7 Nájdená textová dopravná značka 27
4.8 Kandidát, ktorý nie je vyhodnotený ako textová dopravná značka . . . 28
4.9 Posledný nájdený kandidát a značka 28

5.1 Chyba - nájdenie nesprávneho kandidáta 31

Zoznam tabuliek

4.1 Tabuľka prahovacích hodnôt farieb v HSV modeli. 23
4.2 Tabuľka prahovacích hodnôt čiernej farby v HSV modeli. 25
4.3 Tabuľka prahovacích hodnôt bielej farby na šedotónovom obrázku. . . . 25

5.1 Meranie času vykonávania operácií. 30
5.2 Vyhodnotenie hľadania kandidátov na textovú dopravnú značku. 30
5.3 Vyhodnotenie detegovania textu z kandidátov. 32
5.4 Vyhodnotenie detegovania textových dopravných značiek. 32

Úvod

Za posledné desaťročie môžeme pozorovať prudký nárast počtu automobilov na cestách. Pre zabezpečenie bezpečnej a plynulej premávky je dôležité, aby dopravné značky, riadiace premávku, boli dostatočne viditeľné a jednoznačné. Medzi prvky, ktoré môžu zvýšiť bezpečnosť premávky, patria systémy detekcie dopravných značiek. V súčasnosti sú mnohé dopravné značky zobrazované vodičovi aj prostredníctvom navigácie vo vozidle. Takéto zobrazovanie značiek však nemusí byť úplne presné, nakoľko značky nie sú priamo odčítavané z kamery na zariadení.

Cieľom tejto bakalárskej práce je detegovať textové dopravné značky. Tento typ značiek má najčastejšie informatívny charakter a často sa stáva, že vodič z rôznych dôvodov nestihne text na značke prečítať. Naším cieľom je vytvoriť aplikáciu, ktorá na videu dokáže správne detegovať textové dopravné značky.

V prvej kapitole je predstavená história dopravných značiek a ich delenie. Ďalej sú popísané predchádzajúce práce so zameraním na detekciu dopravných značiek. V druhej kapitole sú predstavené teoretické poznatky o farebných modeloch a spracovaní obrazu. Tieto poznatky sú využité pri tvorení návrhu aplikácie v tretej kapitole, kde sú postupne navrhnuté riešenia rôznych problémov. V štvrtej kapitole je popísaná implemenácia samotného riešenia tejto problematiky a je opísaná aplikácia. V piatej kapitole je testovaná detekcia textových dopravných značiek a na základe získaných výsledkov je vyhodnotená jej úspešnosť.

1. Prehľad problematiky

1.1 Dopravné značky

Dopravné značky sú jednoduché piktogramy, ktoré slúžia na riadenie a reguláciu cestnej premávky. Ich význam stanovujú Pravidlá cestnej premávky [1]. Sú dôležitou časťou dopravnej infraštruktúry. Každá značka má svoj tvar, farbu a nesie informáciu pre účastníkov cestnej premávky. Dopravné značky môžu upozorňovať účastníkov cestnej premávky na hroziace nebezpečenstvo, ukladať príkazy, zákazy alebo obmedzenia. Taktiež môžu doplniť, spresniť alebo obmedziť význam inej značky. Z tohto dôvodu musia byť jednoznačné a výrazné.

1.1.1 História dopravných značiek

Koncom devätnásteho storočia po vzniku prvého automobilu, ktorý bol poháňaný spaľovacím motorom, nastal nárast automobilizmu. Tento rast počtu automobilov na cestných komunikáciách podmienil vznik prvej zmluvy o úprave cestnej premávky, ktorou bola Parížska konvencia. Bola podpísaná na druhom Medzinárodnom kongrese v Paríži a bol to dohovor o jazde motorovými vozidlami. O rok neskôr boli navrhnuté prvé dopravné značky, ktoré boli väčšinou čiernobiele a obsahovali obrázky a text. V roku 1949 vznikol Ženevský protokol (Svetový dohovor o cestnej a automobilovej doprave), ktorý sa skladal z Dohovoru o cestnej premávke a Protokolu o dopravných značkách a signáloch. Neskôr v roku 1968 sa vo Viedni konala Viedenská konvencia, na ktorej bol upravený Ženevský protokol. Túto zmluvu podpísala väčšina európskych krajín, medzi ktorými bolo aj Československo [2].

1.1.2 Rozdelenie dopravných značiek

Dopravné značky sa delia na zvislé (sú postavené vedľa vozovky) a vodorovné (sú nakreslené na ceste). Zvislé dopravné značky sa z hľadiska významu delia na [1]:

- Výstražné značky,
- Značky upravujúce prednosť v jazde a dodatkové tabuľky s tvarom križovatky,

- Zákazové značky,
- Príkazové značky,
- Informatívne prevádzkové značky,
- Informatívne smerové značky,
- Informatívne iné značky a dodatkové tabuľky.

1.1.3 Rozdielnosť dopravných značiek v rôznych štátoch

Napriek tomu, že väčšina európskych a viacero iných krajín podpísalo Viedenskú konvenciu, ktorá určuje ako majú značky vyzerať, sa nájdu rozdiely medzi dopravnými značkami v rôznych štátoch. V niektorých krajinách sa používajú značky s textom a v iných zase značky so symbolmi. Na obrázku 1.1 môžeme vidieť rozdiely medzi dopravnými značkami používanými v rozdielnych krajinách.

Obr. 1.1: Rozdiely v dopravných značkách v Holandsku, Veľkej Británii, USA a Austrálii [3].

1.2 Podobné práce

1.2.1 Rozpoznávanie dopravných značiek

Cieľ práce

Rozpoznávaním dopravných značiek sa venoval v bakalárskej práci Miloš Fabian [4]. V práci opísal problematiku detekcie a rozpoznávania červených zákazových dopravných značiek. Cieľom práce bolo implementovať jednu metódu, zaoberajúcu sa rozpoznávaním dopravných značiek a s tým súvisiace metódy výberu vhodných príznakov používané v skúmanej oblasti ako sú farba, geometrické charakteristiky a iné.

Riešenie

Ako programovací jazyk použil C++ a na pomoc so spracovaním obrazu využil knižnicu OpenCV. Prvým krokom pri riešení bolo získať oblasť, kde by sa mohla nachádzať dopravná značka. Tento problém riešil tak, že najskôr obrázok konvertoval z BGR farebného modelu do RGB, kvôli tomu aby sa obraz vedel správne vykresliť v okne aplikácie. Potom nasledovala konverzia z RGB do HSV a na tento obraz použil OpenCV funkciu cv::inRange, výsledkom ktorej je binárny obraz, ktorý potom spojil s HSV obrazom. Obraz ešte vyhladil Gaussovským filtrom. Ďalším krokom bola segmentácia podľa tvaru, kde použil Houghovu transformáciu na hľadanie kruhov, výsledkom čoho bolo získanie oblasti obrázka, kde sa pravdepodobne nachádza kruhová značka. Rozpoznávanie dopravných značiek vyriešil porovnávaním príznakov a použil algoritmy SIFT (Scale-Invariant Feature Transform) a SURF (Speeded Up Robust Feature), ktoré najskôr detegujú kľúčové body a potom vypočítajú ich deskriptory, ktoré porovnáva s deskriptormi získanými zo vzorových obrázkov.

Testovanie a výsledky

Testovaním prišiel nato, že pri rozpoznávaní kandidátov bola metóda SURF menej spoľahlivejšia ako metóda SIFT. Algoritmu SIFT sa podarilo značky detegovať až s 89,09% úspešnosťou, zatiaľ čo algoritmu SURF len s 52,73% úspešnosťou.

1.2.2 Detekcia a rozpoznávanie dopravných značiek

Cieľ práce

Detekcii a rozpoznávaniu dopravných značiek sa venovala v bakalárskej práci aj Anikó Szabóová [5], ktorá testovala detegovanie modrých dopravných značiek pomocou algoritmov SIFT, SURF, MSER, Harrisov rohový detektor, FAST a BRISK.

Riešenie

Prácu implementovala v MATLAB-e. Pri predspracovaní obrazu skúšala obraz konvertovať do RGB, HSV a CIE L*a*b* farebného modelu. Na vyhladzovanie šumu použila Gaussov a mediánový filter a potom nasledovala segmentácia podľa farby. Po prahovaní snímku upravila pomocou morfologických operácii otvorenia a uzatvorenia, ktorými odstránila príliš malé oblasti. Po získaní regiónov, kde by sa značka mohla nachádzať, nasleduje segmentácia podľa tvaru. Pomocou Houghovej transformácie zisťovala kruhovitosť danej oblasti. Na rozpoznávanie dopravných značiek použila algoritmy SIFT, SURF, MSER, Harrisov rohový detektor, FAST a BRISK, pomocou ktorých najskôr našla kľúčové body a potom vypočítala ich deskriptory, ktoré potom porovnávala s deskriptormi vypočítanymi zo vzorových obrázkov. Okrem týchto metód použila aj metódu porovnávania pixelov šablóny a získanej oblasti.

Testovanie a výsledky

Pri testovaní sa jej podarilo získať najlepšie výsledky pomocou farebného modelu HSV a použitím Gaussového filtra. Farebný model RGB je veľmi citlivý na osvetlenie a mediánový filter je náročný na spracovanie a spomaľuje beh programu. Spomedzi algoritmov na rozpoznávanie najlepšie obstáli deskriptory SURF a SIFT. Za nimi skončili algoritmus MSER a Harrisov rohový detektor a najhoršie skončili FAST a BRISK. Algoritmus SIFT klasifikoval značky so 78% úspešnosťou, SURF s 56% úspešnosťou. Porovnávaním pixelov získala až 80% úspešnosť. Kombináciou dvoch techník sa jej podarilo detegovať dopravné značky s 93% úspešnosťou.

1.2.3 Rozpoznávanie dopravných značiek a ich použitie v mapových aplikáciách

Cieľ práce

Štefan Toth vo svojej práci Rozpoznávanie dopravných značiek a ich použitie v mapových aplikáciách [6] využil na klasifikáciu kandidátov umelé neurónové siete.

Riešenie

Na implementáciu si zvolil programovací jazyk C#. Pri predspracovaní obrázka použil na vyhladenie Gaussov filter. Na detekciu oblastí si zvolil segmentáciu na základe farby, ktorú vykonával na obraze skonvertovaného do Cie L*a*b* farebného modelu. Z tohto obrazu potom vyprahoval červenú, modrú a žltú farbu. Následne použil morfologickú operáciu otvorenia, ktorá odstránila šum a príliš malé oblasti. Na klasifikáciu kandidátov si vytvoril niekoľko neurónových sietí, do ktorých sa značky radili podľa

tvaru a farby. Na sledovanie dopravnej značky implementoval Lucas-Kanadeho algoritmus, ktorý funguje na princípe porovnávania dvoch po sebe idúcich snímok. Potom nasledovalo určenie polohy danej značky kde získal zemepisnú šírku, dĺžku a smer.

Testovanie a výsledky

Úspešnosť detekcie bola 70%. Z týchto 70% je úspešnosť neurónových sietí okolo 85%. Výsledkom práce je .NET knižnica, ktorá umožňuje sledovať a rozpoznávať viac ako 75 rôznych druhov značiek.

2. Teoretická časť

2.1 Farebné modely a priestory

Farebný model je abstraktný matematický model, ktorý opisuje ako môžeme vyjadriť farbu pomocou n-tíc čísel. Väčšinou sú to tri alebo štyri hodnoty alebo farebné komponenty. Ak je farebný model asociovaný s presným predpisom, ako sú jednotlivé komponenty interpretované, hovoríme o farebnom priestore. Pri detekcii dopravných značiek bývajú v prácach využité rôzne farebné priestory. Použitie každého farebného modelu prinesie určité výhody, ale aj nevýhody. Medzi najviac používané patria napríklad RGB, HSV, HSI, CIE L*a*b* modely. Farebné modely a priestory sa delia na [7]:

- Homogénne priestory farieb,
- Hardvérovo orientované modely,
- Modely farieb v televízii a videotechnike,
- Používateľsky orientované modely.

2.1.1 CIE L*a*b* model

Je to trojdimenzionálny farebný priestor, ktorý sa skladá z jasovej zložky L* a z dvoch farebných zložiek a* a b*. Hodnoty L* jasovej zložky sa pohybujú od 0 (čierna) po 100 (biela). Farebné zložky tvoria dve osi [7]. Záporné hodnoty osi a* prislúchajú zelenej farbe a kladné hodnoty osi a* prislúchajú červenej farbe. Záporné hodnoty osi b* prislúchajú modrej a kladné hodnoty osi b* prislúchajú žltej farbe. Tento model použili vo svojej práci Lopez a Fuentes pri detekcii dopravných značiek [8].

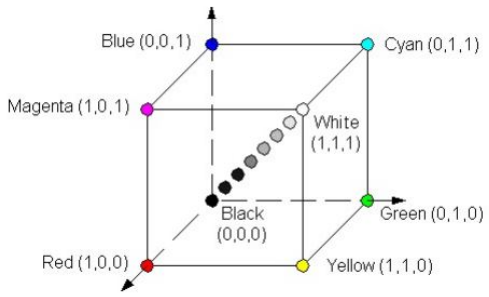
Obr. 2.1: Farebný model Cie L*a*b* [9].

2.1.2 Lineárny model RGB

Patrí medzi hardvérovo orientované modely. Tento model obsahuje tri primárne farby:

- R - červená (red),
- G - zelená (green),
- B - modrá (blue).

Ostatné farby vzniknú aditívnym skladaním týchto farieb. Tieto farby vyjadrujeme pomocou váhového súčtu jednotlivých zložiek. Skladaním týchto primárnych farieb vznikajú sekundárne farby - v tomto modeli sú to modrozelená (cyan), fialová (magenta) a žltá (yellow) [7]. Model RGB je reprezentovaný jednotkovou kockou (obr. 2.2), umiestnenou na začiatku súradnicovej sústavy Euklidovského priestoru. Jednotlivé osi reprezentujú množstvo príslušnej farebnej zložky vo výslednej farbe. Intenzita základných farieb sa vyjadruje číslom z intervalu <0,1>. V množine základných farieb z vrcholov sa nachádza dokopy osem farieb - primárne, sekundárne, biela (Vrchol[1,1,1]) a čierna (Vrchol[0,0,0]). Na diagonále medzi bielou a čiernou farbou sa nachádzajú odtiene šedej (gray).



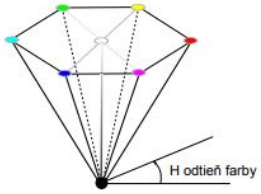
Obr. 2.2: Farebný model RGB [7].

2.1.3 HSV a HSL model

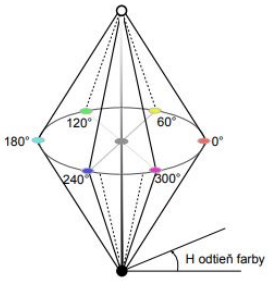
HSV a HSL patria medzi používateľsky orientované modely. Medzi hlavné parametre patria:

- H - odtieň (hue),
- S - sýtosť (saturation),
- V a L - jasová hodnota, svetlosť (value, lightness).

Odtieň farby určuje dominantnú spektrálnu farbu, sýtosť určuje prímes ďalších farieb a jasová hodnota určuje množstvo bieleho bezfarebného svetla. Na priestorové zobrazenie HSV modelu používame väčšinou tvar obráteného pravidelného šesťbokého ihlanu (obr. 2.3a), ktorý vznikol deformáciou RGB kocky, kde v bode [0,0,0] je zobrazená čierna farba. Na zobrazenie modelu HSL používame obojstranný kužeľ (obr. 2.3b). Jedným z nedostatkov HSV modelu je, že zmena farebného odtieňa nie je plynulá - H sa pohybuje po šesťuholníku [7]. V prácach pri detekcii dopravných značiek je tento model najpoužívanejší.



(a) Farebný model HSV [7].



(b) Farebný model HSL [7].

Obr. 2.3

2.2 Snímanie obrazu

Obraz získavame pomocou kamery umiestnenej na palubnej doske automobilu. Je otočená v smere jazdy, takže sú snímané dopravné značky umiestnené pred automobilom nad vozovkou a dopravné značky umiestené vedľa vozovky (obr. 2.4).

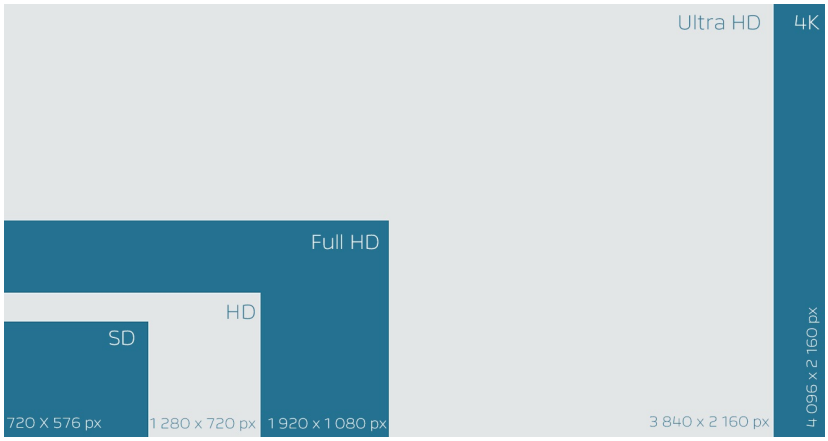
Obr. 2.4: Obraz zosnímaný kamerou na palubnej doske automobilu.

2.2.1 Rozlíšenie vstupného obrazu

Jednou z najdôležitejších vlastností videa je rozlíšenie. Uvádza počet obrazových bodov - pixelov, z ktorých je tvorené výsledné video. Medzi napoužívanejšie rozlíšenia patria:

- Full HD - 1920 × 1080 pixelov,
- Ultra HD - 3840 × 2160 pixelov,
- 4K - 4096 × 2160 pixelov.

Veľkosť rozlíšenia môže vplývať na správnu detekciu dopravnej značky.



Obr. 2.5: Ukážka rozdielov medzi rozlíšeniami [10].

2.2.2 Problémy pri získavaní obrazu

Pri získavaní obrazu sa môžeme stretnúť s mnohými faktormi, ktoré negatívne vplývajú na správne fungovanie detekcie. Jedným z týchto faktorov je počasie. Príkladom tohto zlého vplyvu je zasnežená značka, znížená viditeľnosť pri hmle, atď. Medzi ďalšie faktory môže patriť napríklad poškodenie značky.

2.3 Predspracovanie obrazu

Slúži na pripravenie obrazu na ďalšie spracovanie. Pomocou spracovania obrazu môžeme opraviť pixel, ktorý je skreslený šumom, na základe jeho susedných pixelov. Avšak nedokážeme zvýšiť množstvo informácie spracovaného obrazu [11]. Rozlišujeme viacero metód predspracovania obrazu [12]:

- Bodové jasové transformácie,
- Geometrické transformácie,
- Lokálne predspracovanie (filtrácia, ostrenie a detekcia hrán),
- Obnovenie obrazu pri známej degradácii,
- Matematická morfológia.

Na odstránenie šumu pri detekcii dopravných značiek je používané väčšinou predspracovanie pomocou lokálnych operátorov. Tieto operátory pracujú tak, že prechádzajú celý obraz pixel po pixely a na základe susedných pixelov upravia hodnotu aktuálneho pixelu.

2.3.1 Vyhladzovanie obrazu (filtrácia)

Vyhladzovanie používame na odstránenie šumu a ostrých hrán z obrazu (vysokofrekvenčné zložky vo Fourierovskom spektre). Sprievodným javom pri vyhladzovaní býva rozmazanie predtým ostrých hrán [11].

Spriemerovanie

Spriemerovanie je najjednoduchší spôsob vyhladzovania šumu v obraze. Pre každý bod obrazu sa hodnota jasu nahradí aritmetickým priemerom jasov jeho susedov (väčšinou sú to body zo štvorcového nepárneho okolia - 3x3, 5x5). Je to vlastne špeciálny prípad konvolúcie [12]. Konvolučná maska má tvar:

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.1}$$

Gaussovský filter

Najpoužívanejšie filtrovanie pri detekcii dopravných značiek v existujúcich prácach je pomocou Gaussovej masky - Gaussovský filter. Masku získame tak, že v konvolučnej maske posilníme význam stredového bodu a jeho 4 najbližších susedov. Vďaka tomu

dosahuje lepšie a presnejšie výsledky ako pri spriemerovaní. Kvôli tomu vlastne vzniká rozmazávanie hrán. [12]

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.2}$$

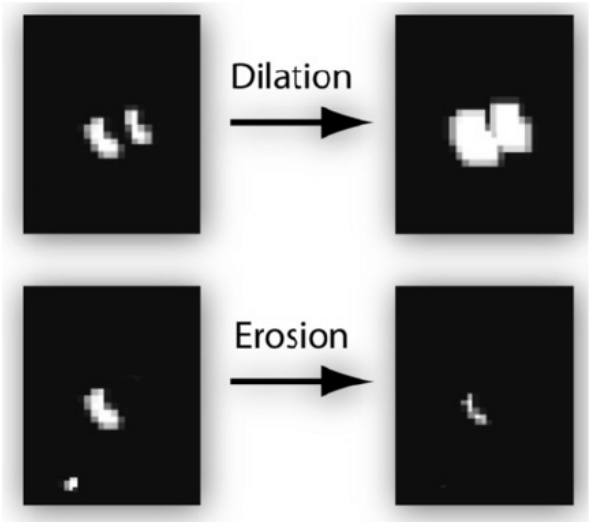
Mediánový filter

Mediánový filter usporiada vzostupne jasové úrovne z okolitých pixelov a stredná hodnota z tejto postuponosti bude nová hodnota pixelu. Nevýhodou použitia mediánového filtru je, že poruší tenké čiary a ostré hrany v obraze. [12]

2.3.2 Morfologické operácie

Medzi dve základné morfologické operácie patria:

- Dilatácia - pomocou nej dokážeme zväčšiť plochu bielej binárnej oblasti. Pomocou matematických výpočtov dokáže vyrátať okolie jedného bodu a tým pádom vyplní diery.
- Erózia - je to opak dilatácie. Zmenšujeme daný objekt a odstraňujeme menšie objekty.



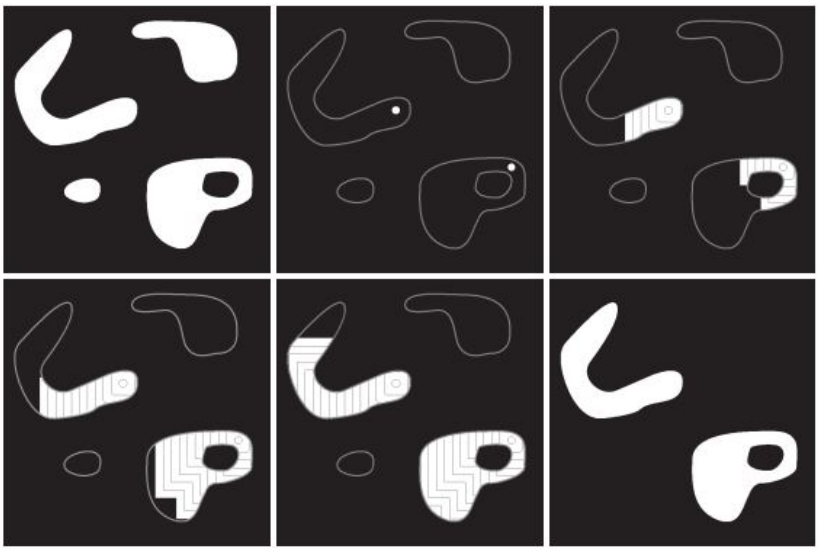
Obr. 2.6: Dilatácia a Erózia [13].

Kombináciou týchto dvoch základných operácií vznikajú ďalšie morfologické operácie:

- Otvorenie - použijeme najskôr dilatáciu a potom eróziu,
- Zatvorenie - použijeme najskôr eróziu a potom dilatáciu.

Morfologickú operáciu otvorenie použil v práci Štefan Toth [6]. Ďalšie užitočné morfologické operácie:

- Morfologická rekonštrukcia - vstupom sú dva maskovacie obrazy A a B a štrukturálny element E. Obraz A je ďalej spracovávaný pomocou inej morfologickej operácie (napr. dilatácia alebo erózia) s daným štrukturálnym elementom E a potom sa vytvorí prienik s druhým vstupným obrazom B. Postup sa opakuje až kým nedosiahneme idempotenciu - nemenný stav v maskovacom obraze B. [7]



Obr. 2.7: Morfologická rekonštrukcia [14].

2.4 Segmentácia

Z definície segmentácie vyplýva, že povôdný obraz rozdelíme na disjunktné homogénne časti. Každá táto časť má nejakú vlastnú podmienku homogenity. Medzi dva základné typy segmentácií patria:

- Segmentácia podľa tvaru,
- Segmentácia podľa farby.

Keďže dopravné značky sú špecifické svojím tvarom (trojuholník, štvorec, atď.) ale aj svojou farbou (modrá, červená, zelená, atď.), pri segmetnácii obrazu a následnom detegovaní dopravných značiek sa používajú oba typy. Segmentácia má dve úrovne [15]:

- Kompletná - segmentované objekty sú už priamo hľadané objekty,
- Čiastočná - segmentované objekty treba ešte ďalej spracovávať.

Metódy segmentácie rozdeľujeme na [15]:

- Globálne - napr. prahovanie,
- Určovanie hraníc - napr. detekcia hrán,
- Určovanie oblastí - napr. metódy založené na narastaní oblastí.

Segmentácia podľa farby

Prahovanie je najpoužívanejšia, najrýchlejšia a najmenej výpočtovo náročná metóda segmentácie. Najčastejšie sa používa globálny prah, ktorý závisí len na hodnote jasu obrazu. Zastúpenie jednotlivých jasových úrovní v obraze sa väčšinou vyjadruje v histograme.

Segmentácia podľa tvaru

Medzi napoužívanejšie metódy pri hľadaní rôznych tvarov značky patrí Cannyho detektor [16] a Houghova transformácia [17]. Väčšinou sa tieto metódy navzájom kombinujú.

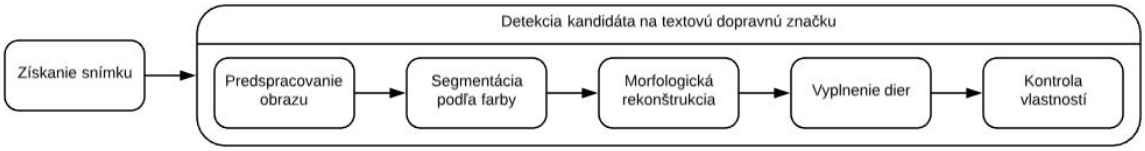
3. Návrh riešenia problematiky

Cieľom našej práce je detegovať textové dopravné značky na videu. Postup riešenia tejto problematiky si môžeme logicky rozdeliť na dva veľké celky:

- Hľadanie kandidáta na textovú dopravnú značku,
- Hľadanie textu na kandidátovi.

3.1 Detekcia kandidáta na textovú dopravnú značku

V tejto časti budeme pracovať s jednotlivými snímkami získanými z videa, na ktorých budeme vykonávať rôzne operácie. Kvôli časovej náročnosti algoritmu bude dôležité čo najviac znížiť počet nájdených oblastí tak, aby tam zostali všetky textové dopravné značky. Postup môžeme vidieť na obrázku 3.1.



Obr. 3.1: Návrh postupu k nájdeniu kandidáta na textovú dopravnú značku.

Predspracovanie obrazu

Prvým krokom pri práci so snímkou bude konverzia obrazu do HSV farebného modelu. Potom na snímku aplikujeme mediánový filter, pomocou ktorého odstránime z obrázku nežiadúci šum.

Obr. 3.2: Znázornenie krokov pri predspracovaní obrázku.

Segmentácia

Po vyhladení obrázka nasleduje segmetnácia podľa farby. Tento krok bude veľmi dôležitý z dôvodu, že potrebujeme čo najpresnejšie určiť, kde by sa značka s danou farbou mohla nachádzať. Keďže primárne farby dopravných značiek, ktoré obsahujú text sú zelená, modrá a hnedá, budeme sa sústrediť práve na tieto tri farby. Výsledkom segmentácie bude binárny obrázok, kde pixely s hľadanými farbami budú mať hodnotu 1 a ostatné pixely hodnotu 0.

Morfologické operácie

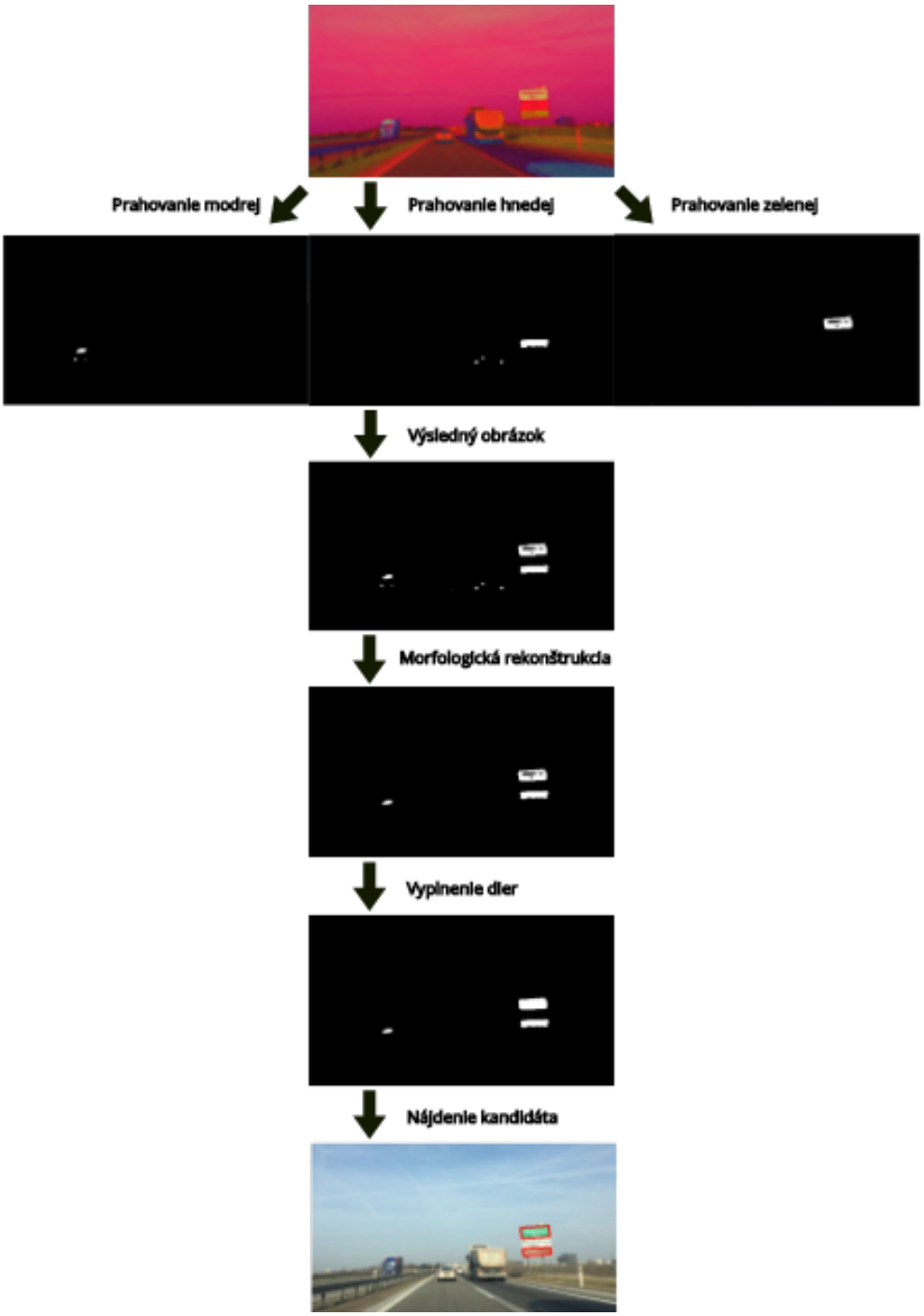
Následne na obrázok aplikujeme morfologickú rekonštrukciu eróziou, pomocou ktorej z obrázku odstránime malé plochy, ktoré zostanú po segmentovaní.

Vyplnenie dier

Po vykonaní morfologickej rekonštruckie získame všetky oblasti, ktoré môžu byť dopravná značka. Avšak vnútri v týchto oblastiach sa budú stále nachádzať oblasti, ktoré nie sú označené ako hľadaná oblasť kvôli farbe, ktorú sme vyradili pri segmentácii a preto ich vyplníme. Bude to napríklad biely text alebo iná biela dopravná značka, ktorá sa nachádza v nájdenej modrej, zelenej alebo hnedej oblasti.

Kontrola vlastností kandidáta

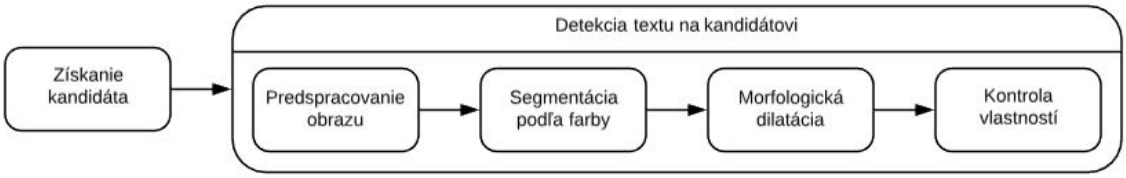
Ďalej sa budeme sústrediť na vlastnosti značiek, ktoré chceme detegovať. Vlastnosti, ktoré budeme kontrolovať sú výška a šírka. Týmto krokom vyradíme nežiadúce objekty, o ktorých vieme určite povedať, že nebudú dopravná značka.



Obr. 3.3: Znázornenie krokov k nájdeniu kandidáta

3.2 Detekcia textu na kandidátovi

V tejto časti budeme pracovať už len so získanou oblasťou - kandidátom, získaným z pôvodného a neupraveného obrázku. Keďže detekcia bude bežať na videu, musíme brať ohľad aj na časovú náročnosť algoritmu, takže sa musíme pokúsiť nájsť optimálne riešenie, čo sa týka úspešnosti, aj rýchlosti detekcie. Pri hľadaní textu sa musíme zamyslieť, na základe akých vlastností ho budeme hľadať. V našom riešení sa využijeme dĺžku slov a nadväznosť písmen v slovách. Na obrázku 3.4 môžeme vidieť postupnosť krokov pri hľadaní textu.



Obr. 3.4: Návrh postupu k nájdeniu textu na kandidátovi.

Predspracovanie obrazu

Z dôvodu, že text na dopravných značkách je čierny alebo biely, obrázok prevedieme do šedotónového obrazu.

Segmentácia

Segmentáciu rozdelíme na dve časti. Na dopravných značkách modrej, zelenej a hnedej farby budeme hľadať text bielej farby. Pre prípad, že sa v nájdenej značke nachádza iná biela značka, budeme hľadať aj čierny text.

Morfologické operácie

Následne na obrázok aplikujeme morfologickú dilatáciu, pomocou ktorej rozšírime oblasti jednotlivých písmenok, výsledkom čoho bude vytvorenie spoločnej oblasti pre všetky písmenká slova.

Kontrola vlastností textu

V tomto kroku budeme kontrolovať výšku a šírku nájdenej oblasti. Vo väčšine prípadov platí, že šírka textu by mala byť väčšia ako jeho výška.

Obr. 3.5: Znázornenie krokov k nájdeniu textu na kandidátovi.

3.3 Užívateľské prostredie

Vytvorenie správneho a jednoduchého užívateľského prostredia bude dôležité nielen pre testovanie detekcie textovej dopravnej značky, ale aj pri hľadaní kandidáta na textovú dopravnú značku.

4. Riešenie problematiky a implementácia

4.1 Použité nástroje

Python

Pre implementáciu algoritmu na detekciu textových dopravných značiek sme si vybrali programovací jazyk Python [18]. Podporuje funkcionálne, štruktúrované aj objektovo orientované programovanie. Je to dynamicky typový jazyk, ktorý podporuje veľké množstvo vysokoúrovňových dátových typov. Tento programovací jazyk sme si vybrali aj preto, že podporuje OpenCV knižnicu, ktorá nám značne zjednoduší prácu s obrazom. V implemetácií použijeme verziu 3.7.2 [19]. Ako programovacie prostredie použijeme Python IDE PyCharm [20].

OpenCV

Je to open-source knižnica určená predovšetkým na prácu s obrazom [21]. Zameriava sa hlavne na počítačové videnie a spracovanie obrazu v reálnom čase. Túto knižnicu je možné použiť v mnohých jazykoch vrátane Pythonu. V implementácií budeme pracovať s verziou 4.0.1 [22].

4.2 Získavanie obrazu

Spôsob získavania obrazu je opísaný v časti Snímanie obrazu 2.2. Na natáčanie videí sme použili zariadenia s vysokým aj nižším rozlíšením. Zariadenia použité na snímanie obrazu a ich rozlíšenia sú:

- Apple Iphone 6S - video 4K (30 fps) / FullHD (60 fps),
- Samsung Galaxy S9 - video 4K (60 fps),
- Canon PowerShot A580 - 640 x 480 (20 fps).

Získané videá sú uložené vo formáte MP4 a MOV. Videá boli natáčané za prevažne dobrých sveteľných podmienok počas dňa, za rôzneho počasia, keď bolo slnečno, šero a počas dažďa.

Obr. 4.1: Rôzne počasie počas natáčania.

4.3 Postup pri implementácii

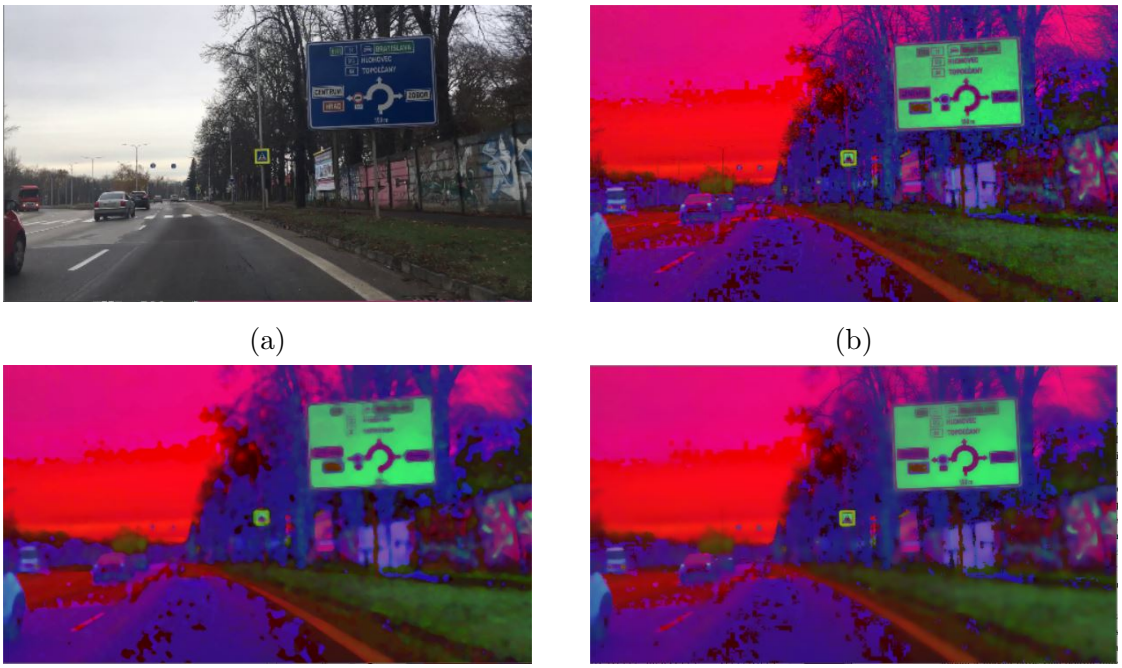
V samotnom riešení problematiky pri implementácii sme postupovali hlavne podľa návrhu riešenia problematiky, opísanom v kapitole 3. Pri niektorých krokoch v návrhu sme testovaním prišli nato, že dané riešenie nie je úplne optimálne a tak sme implementovalý iné. Najväčší problém nám robila hlavne časová náročnosť algoritmu, keďže prebieha algoritmus beží na videu a teda je to detekcia v reálnom čase. Pre väčšiu prehľadnosť sme si implementáciu programu rozdelili na dve triedy. Prvou triedou je *GUI* - používateľské rozhranie, ktorá obsahuje hlavne metódy slúžiace na prácu s Tkinterom. Druhou triedou je *VideoFrame*, ktorej úlohou je práca so samotným videom a jeho snímkami.

4.3.1 Spracovanie snímky

V triede *VideoFrame* sú implementované metódy, ktoré sa zaoberajú získaním snímky z videa a prácou s jednotlivými snímkami. Metóda *processFrame* najskôr získa snímku z videa a potom postupne volá ďalšie metódy, ktoré ju ďalej spracovávajú.

Hľadanie kandidáta

Prvá volaná metóda - *preprocessing* slúži na vykonanie troch operácii. Z dôvodu, že vstupné videá sú natáčané vo veľmi vysokom rozlíšení a rýchlosť vykonávania nášho algoritmu na takomto videu by bola veľmi pomalá, prvou operáciou je downsampling. Na vykonanie tejto operácie sme použili opencv metódu *cv.resize* a napevno sme si určili šírku 480 pixelov a výšku 270 pixelov. Druhá operácia, ktorú vykonávame v tejto metóde je konverzia farebného modelu. Z dôvodu, že knižnica OpenCV ukladá snímky vo formáte n-dimenzionálneho poľa, ktoré majú prednastavené poradie farebných kanálov BGR - modrá, zelená a červená, vykonáme konverziu do HSV farebného modelu pre pracovný obrázok a konverziu do RGB farebného modelu pre výstupný obrázok. Na konverziu použijeme funkciu *cv.cvtColor*. Posledná operácia, ktorú vykonáme v tejto metóde je aplikovanie vyhladzovacieho filtra na pracovný obrázok. Na aplikovanie mediánového filtra v implementácii použijeme funkciu *cv.medianBlur*. Okrem mediánového filtra sme testovali aj bilaterálny filter, pri ktorom sme dosiahli mierne lepšiu úspešnosť ako pri mediánovom, ale tento filter mal väčšiu časovú náročnosť.



(a) (b)

(c) (d)

Obr. 4.2: Predspracovanie obrázku pri implementácii. 4.2a - Pôvodný obrázok v RGB modeli. 4.2b - Pôvodný obrázok v HSV modeli. 4.2c - Mediánový filter aplikovaný na obrázok v HSV. 4.2d - Bilaterálny filter aplikovaný na obrázok v HSV.

Druhou volanou metódou je *processColors*, v ktorej vykonávame ďalšie tri operácie - prahovanie, morfologickú rekonštrukciu a vyplnenie dier. Na prahovanie modrej, zelenej a hnedej farby sme použili funkciu *cv.inRange*, ktorá nám umožňuje nastaviť interval hodnôt pre každý farebný kanál. Keďže pracovný obrázok je v HSV modeli, tak nastavujeme tri intervaly pre odtieň (hue), sýtost (saturation) a jasovú hodnota (value). Na tieto hodnoty majú vplyv rôzne faktory, ako napríklad počasie. Použité hodnoty sme získali experimentovaním a môžeme ich vidieť v tabuľke 4.1.

Tabuľka 4.1: Tabuľka prahovacích hodnôt farieb v HSV modeli.

Farba	**Odtieň - H**	**Sýtosť - S**	**Jasová hodnota - V**
Modrá	<100, 117>	<110, 250>	<45, 180>
Zelená	<60, 98>	<95, 255>	<50, 230>
Hnedá	<8, 17>	<60, 175>	<130, 245>

Výsledkom tejto funkcie je binárny obrázok pre každú farbu. Následne obrázky spojíme do jedného. Potom na vyprahovaný obrázok aplikujeme morfologickú rekonštrukciu obrazu eróziou pomocou funkcie *skimage.reconstruction* z knižnice Scilit-image. Scilit-image (skimage) je open-source knižnica určená na spracovanie obrazu pre programovací jazyk Python [23]. Obsahuje rôzne algoritmy na segmentáciu, filtrovanie,

morfológiu a ďalšie. Posledný krok pri úprave spracovávaného obrázku je vyplenie dier. Tu sme použili metódu *fillHolles*, pomocou ktorej vyplníme vnútro nájdenej oblasti a tým vznikne jedna spoločná biela oblasť. Výsledný obrázok po úpravách môžeme vidieť na obrázku 4.3.

(a) (b)

Obr. 4.3: 4.3a - Pôvodný obrázok v RGB modeli. 4.3b - Upravený binárny obrázok pripravený na hľadanie kandidáta.

Nasledujúcim krokom je samotné hľadanie kandidáta. Táto časť je implementovaná v metóde *findCandidate*, v ktorej sme na nájdenie kontúr na obrázku použili funkciu *cv.findContours*. Pre zníženie počtu kandidátov sme vytvorili okolo kontúry opísaný obdĺžnik pomocou *cv.minAreaRect*, na ktorom kontrolujeme správnosť rozmerov ako sú výška, šírka a uhol sklonu. V ďalšej kontrole porovnávame obsah kontúry s obsahom opísaného obdĺžnika, ktoré sme získali pomocou funkcie *cv.contourArea*. Keďže vieme, že kontúra by mala mať tvar obdĺžnika, tieto dva obsahy by mali byť takmer rovnaké. Vďaka tomuto postupu sme získali oblasť kandidáta, na ktorom budeme v ďalších krokoch hľadať text.

Hľadanie textu

Na hľadanie textu používame funkciu *findText*, ktorá pracuje s pôvodným vstupným obrázkom získaným z videa. Z tohto obrázku následne vyrežeme a vyrovnáme oblasť kandidáta získaného v predchádzajúcej časti (opísaný obdĺžnik) pomocou funkcie *cropRect* a ďalej budeme pracovať iba s touto oblasťou. Pri hľadaní textu je potrebné si uvedomiť, že text na modrých, zelených a hnedých značkách je bielej farby, ale na nájdenej značke sa môže vyskytnúť iná biela značka, ktorá obsahuje čierny text. Preto sme museli implementovať pre každú farbu textu vlastné predspracovanie. Pri detekcii čierneho textu sme získali najlepšie výsledky použitím HSV modelu. Predspracovanie pri hľadaní čierneho textu je implementované vo funkcii *processBlack*. Prvým krokom v predspracovaní je konverzia do HSV pomocou funkcie *cv.cvtColor*. Ďalej vykonáme prahovanie s funkciou *cv.inRange* a nakoniec aplikujeme morfologickú dilatáciu pomocou

funkcie *cv.dilate*, kde ako kernel použijeme maticu

$$kernel = \left[\left[0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\right]\right]. \tag{4.1}$$

Hodnoty prahovania pre čiernu farbu sú zobrazené v tabuľke 4.2.

Tabuľka 4.2: Tabuľka prahovacích hodnôt čiernej farby v HSV modeli.

Farba	**Odtieň - H**	**Sýtosť - S**	**Jasová hodnota - V**
Čierna	všetky	všetky	<0, 80>

Pri detekcii bieleho textu sme získali najlepšie výsledky použitím šedotónového obrazu. Pri použití šedotónového obrazu sme narazili na problém pri slnečnom počasí. Slnko sa odráža od značky, čo spôsobuje, že je biela farba jasnejšia a nie je možné správne vyprahovať oblasť písmen. Tento problém sme vyriešili pridaním prahovania, ktoré hľadá iba veľmi svetlé pixely (obr. 4.4). Hodnoty pri prahovaní sú zobrazené v tabuľke 4.3. Posledný krok pri predspracovaní obrázku je aplikovanie morfologickej dilatácie pomocou *cv.dilate*, kde ako parameter použijeme rovnaký kernel ako pri hľadaní čierneho textu.

(a) (b) (c)

(d) (e) (f)

Obr. 4.4: Motivácia pre implementáciu druhého prahovania pre bielu farbu. 4.4d a 4.4a - Pôvodný obrázok. 4.4e a 4.4b - Prahovanie pre tmavšiu bielu. 4.4f a 4.4c - Pridané prahovanie pre svetlú bielu.

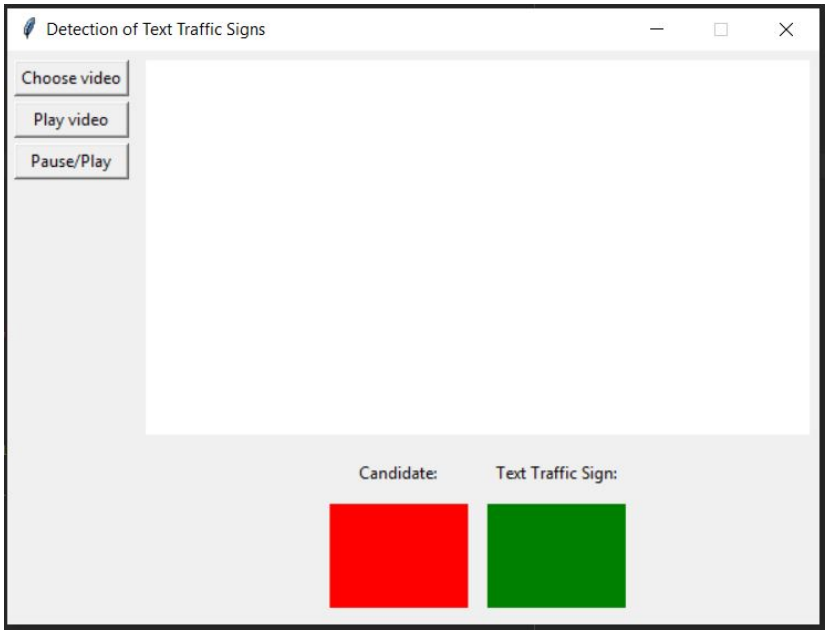
Tabuľka 4.3: Tabuľka prahovacích hodnôt bielej farby na šedotónovom obrázku.

Farba	**Minimálna hodnota pixelu**	**Maximálna hodnota**
Tmavšia biela	85	255
Svetlá biela	180	255

Ďalej postupujeme podobne ako pri hľadaní kandidáta. Pomocou *findContours* získame z obrázku zoznam kontúr, ktorými opíšeme pomocou *cv.minAreaRect* obdĺžnik. Potom nasleduje kontrola vlastností obdĺžnika, pomocou ktorých zisťujeme či je daná oblasť text. Medzi tieto vlastnosti patrí napríklad, že šírka odbĺžnika musí byť väčšia ako jeho výška, uhol sklonu tohto obdĺžnika mal byť približne 0. Avšak medzi útvary, ktoré spĺňajú tieto podmienky patria aj iné objekty (napr. šípka). Preto budeme okrem týchto vlastností kontrolovať aj pomer obsahu kontúry k obsahu opísaného obdĺžnika. Ak nejaká oblasť na kandidátovi spĺňa všetky tieto podmienky, je kandidát vyhodnotený ako textová dopravná značka.

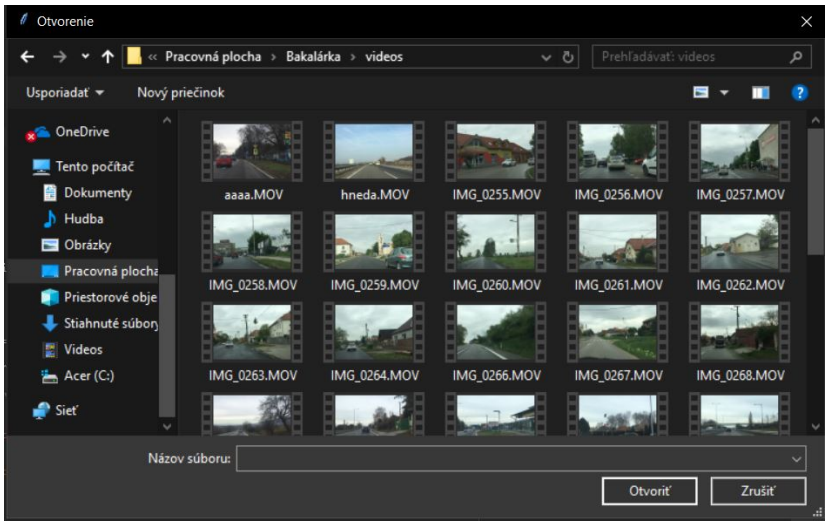
4.3.2 Používateľské prostredie

Na implementáciu používateľského prostredia sme využili Tkinter [24]. Tkinter je knižnica pre programovací jazyk Python, ktorá slúži na tvorbu GUI (Graphic User Interface) a dokážeme ním vytvoriť grafické okno. Na tomto okne vieme vytvárať rôzne prvky ako tlačidlá, textové boxy a iné. V našom prostredí využijeme iba tlačidlá a canvas (plátno). Canvas slúži na zobrazovanie štruktúrovanej grafiky, zobrazovanie grafov, kreslenie útvarov a mnoho ďalších. Po spustení aplikácie sa zobrazí okno, ktoré obsahuje tri tlačidlá, tri canvasy a dva labely, ktoré slúžia na popis ku canvasom, ako môžeme vidieť na obrázku 4.5.



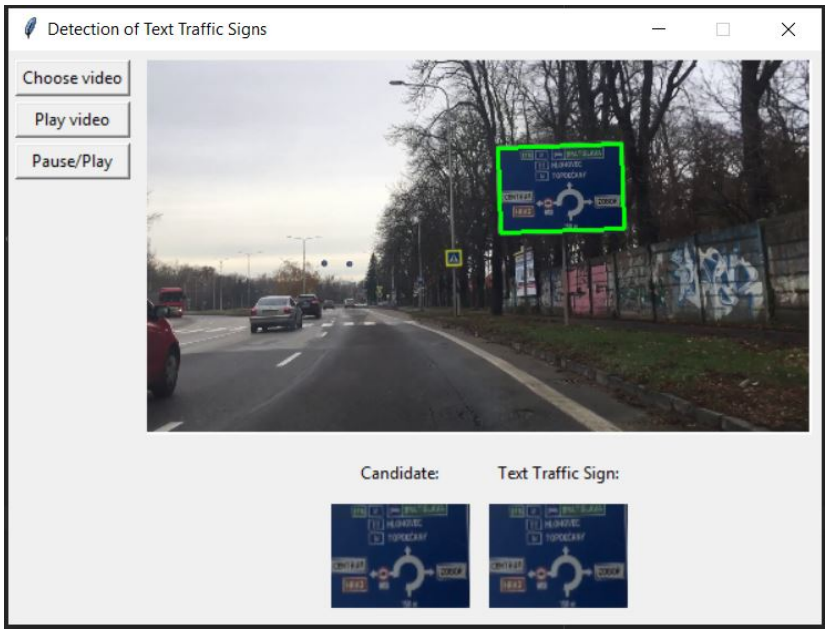
Obr. 4.5: Používateľské rozhranie.

Po kliknutí na tlačidlo *"Choose video"* sa zobrazí dialógové okno, pomocou ktorého si môžeme zvoliť video, ktoré budeme spracovávať (Obrázok 4.6).

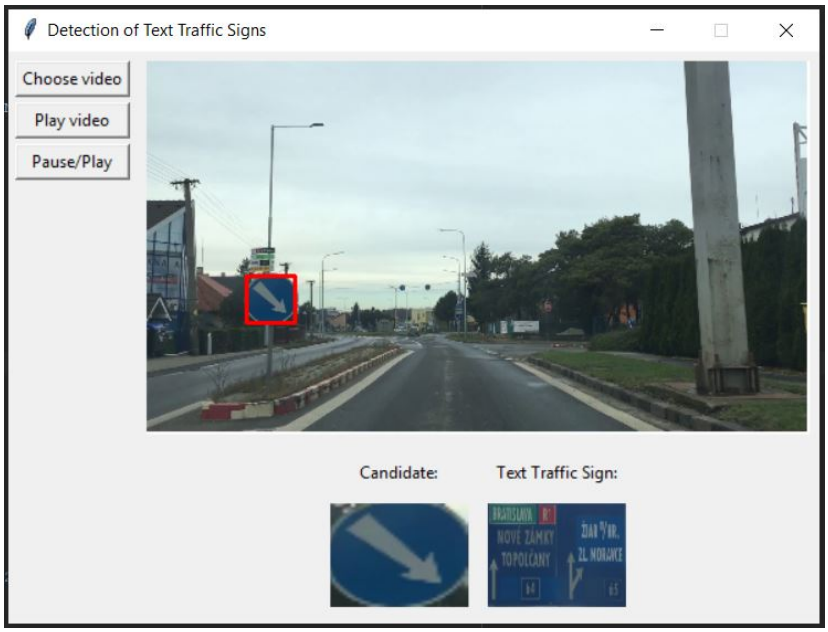


Obr. 4.6: Dialógové okno pre zvolenie videa.

Po zvolení videa a kliknutí na tlačidlo *"Play video"* sa spustí prehrávanie a detekcia textových dopravných značiek na videu. Na malom červenom canvase s popisom *"Candidate:"* sa zobrazujú kandidáti na textovú dopravnú značku a na veľkom canvase bude kandidát označený červeným rámčekom. Ak je kandidát vyhodnotený ako textová dopravná značka, tak sa zobrazí aj v druhom malom zelenom canvase s popisom *"Text Traffic Sign:"* a na veľkom canvase bude označený zeleným rámčekom (Obrázok 4.7). V prípade ak kandidát nie je vyhodnotený ako textová dopravná značka sa zobrazí iba v canvase *"Candidate"* (Obrázok 4.8).

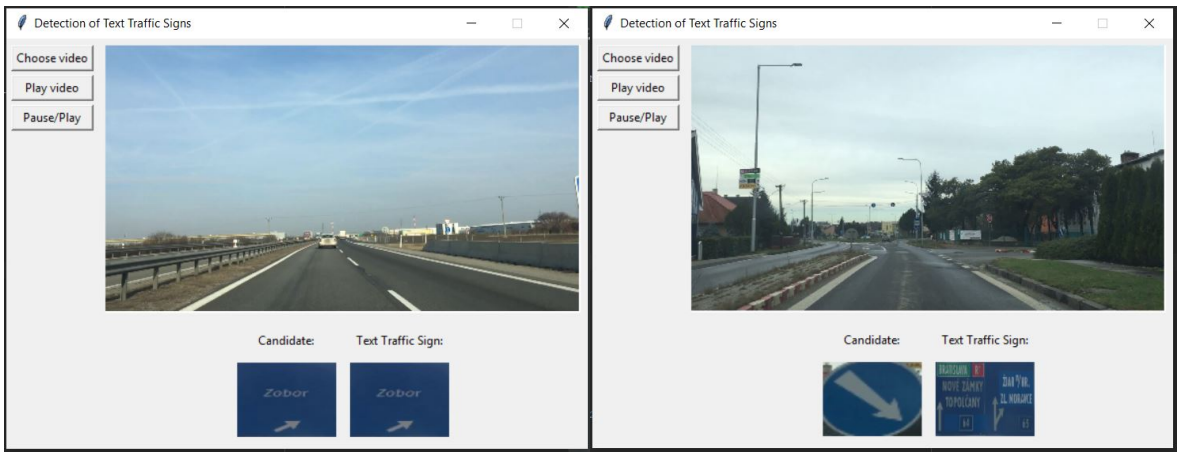


Obr. 4.7: Nájdená textová dopravná značka.



Obr. 4.8: Kandidát, ktorý nie je vyhodnotený ako textová dopravná značka.

Užívateľ môže kedykoľvek zastaviť a pokračovať v detekcii kliknutím na tlačidlo *"Pause/Play"*. Na canvase *"Candidate"* sa bude nachádzať posledný nájdený kandidát a na canvase *"Text Traffic Sign"* sa bude nachádzať posledná nájdená textová dopravná značka. (Obrázok 4.9)



(a) (b)

Obr. 4.9: Posledný nájdený kandidát a textová dopravná značka.

V triede *GUI* sú implementované metódy *drawCanvases* a *drawButtonsMenu*, ktoré slúžia na vykreslovanie canvasov s labelmi a tlačidiel a ďalšie metódy, ktoré zabezpečujú správne fungovanie tlačidiel. Okrem týchto metód je tu implementovaná aj metóda *update*, ktorá slúži na obnovu obrazu, ktorý sa zobrazuje na canvasoch.

5. Vyhodnotenie a výsledky

Testovanie prebiehalo na rôznych videách z cestnej premávky, ktoré boli nasnímané za rôzneho počasia v mestách, aj mimo miest. Samotné testovanie sme si rozdelili na viac častí:

- čas vykonávania metód,
- testovanie úspešnosti nájdenia kandidáta,
- testovanie úspešnosti nájdenia textu na kandidátovi,
- celková úspešnosť detekcie textových dopravných značiek.

Vo výsledných tabuľkách sme úspešnosť vypočítali ako výsledok vzťahu: $\text{úspešnosť} = \frac{\text{správne nájdené značky}}{\text{skutočný počet značiek}}$ a v tabuľkách sú použité skratky:

- TP - True positive - nájdená značka je značka,
- FN - False negative - značka nie je nájdená,
- FP - False positive - nájdená značka nie je značka.

5.1 Čas vykonávania metód

Testovanie sme vykonávali na notebooku s procesorom Intel i5 s frekvenciou 2.40 GHZ.

Tabuľka 5.1: Meranie času vykonávania operácií.

Metóda	Operácia	Čas (milisek.)
preprocessing	downsampling	3
	konverzia do HSV	1
	konverzia do RGB	1
	mediánový filter	4
processColors	prahovanie	2
	morfologická rekonštrukcia	53
	vyplnenie dier	2
findCandidate	hľadanie kandidáta	1
findText	vyrezanie opísaného obdĺžnika	6
	predspracovanie obrázku	3
	hľadanie textu	5

V tabuľke 5.1 je zobrazený čas behu jednotlivých operácií v milisekundách. Tieto hodnoty boli namerané počas behu algoritmu pri nájdení jednej textovej dopravnej značky. S narastajúcim počtom kandidátov na obrázku narastá aj čas behu operácii v metódach *findCandidate* a *findText*. Najdlhšie bežala operácia morfologická rekonštrukcia.

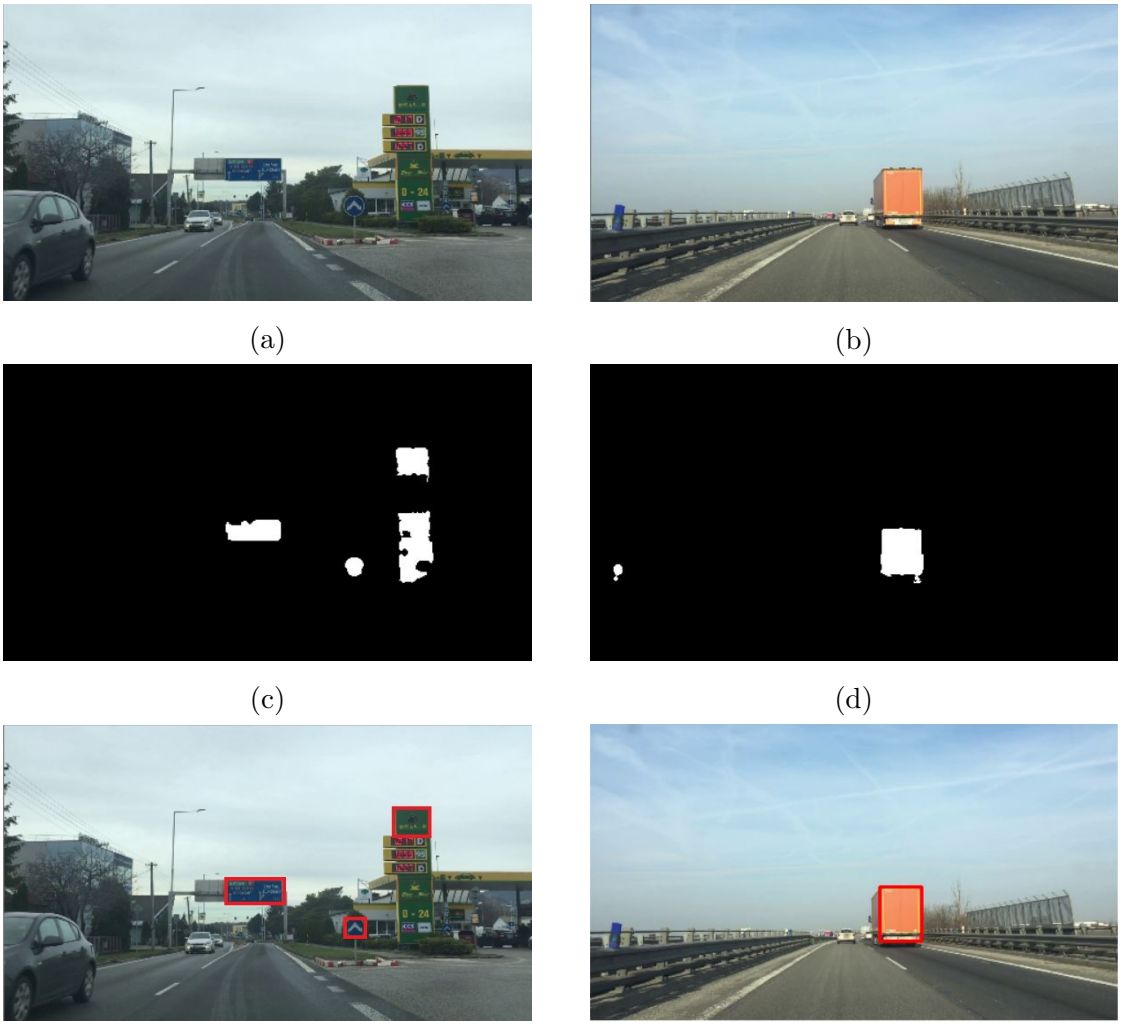
5.2 Hľadanie kandidáta

Tabuľka 5.2: Vyhodnotenie hľadania kandidátov na textovú dopravnú značku.

Farba	Skutočný p.	Správne ***TP***	Nenájdené ***FN***	Nesprávne ***FP***	Úspešnosť (%)
Modrá	106	101	5	7	95,3%
Zelená	44	44	0	4	100,0%
Hnedá	19	15	4	5	78,9%
Celkovo	**169**	**160**	**9**	**16**	**94,7%**

V tabuľke 5.2 sú uvedené hodnoty, ktoré boli získané testovaním. V stĺpci *Skutočný počet* je uvedený celkový počet oblastí, ktoré mali byť vyhodnotené ako kandidát. V stĺpci *Správne* sú uvedené počty oblastí, ktoré aplikácia vyhodnotila ako kandidáti. Počet oblastí, ktoré mali byť aplikáciou vyhodnotené ako kandidáti, ale neboli sú uvedené

v stĺpci *Nenájdené*. V stĺpci *Správne* je zobrazený pomer správne nájdených kandidátov k reálnemu počtu kandidátov na videu v percentách. V stĺpci *Nesprávne* je ukázaný počet nesprávne vyhodnotených oblastí, ktoré nemali byť vyhodnotené ako kandidát. Táto situácia môže nastať, ak daný objekt na obrázku má podobné vlastnosti ako dopravná značka - tvar, farba. Patrí sem napríklad strecha domu, časť billboardu, zadná časť prívesu kamiónov a ďalšie. Táto situácia je zobrazená na obrázku 5.1, kde kamión a stojan sú vyhodnotené ako kandidáti na textovú dopravnú značku.



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

Obr. 5.1: Nájdenie nesprávneho kandidáta. 5.1a a 5.1b - Obrázok v RGB. 5.1c a 5.1d - Binárny obrázok po úpravách. 5.1e a 5.1f - Vyhodnotený obrázok.

5.3 Nájdenie textu na kandidátovi

Tabuľka 5.3: Vyhodnotenie detegovania textu z kandidátov.

Farba	**Kandidáti**	**Skut. p.**	**Správne *TP***	**Nesprávne *FP***	**Nenájdené *FN***	**Úspešnosť (%)**
Modrá	108	55	51	4	4	92,7%
Zelená	48	44	42	3	2	95,5%
Hnedá	20	15	14	1	1	93,3%
Celkovo	**176**	**114**	**107**	**8**	**7**	**93,9%**

Ďalšou testovanou časťou je vyhodnotenie kandidáta za textovú dopravnú značku (tab. 5.3). V stĺpci *kandidáti* sú uvedení všetci kandidáti (správne aj nesprávne), ktorých sme získali pri testovaní v časti 5.2. Počet reálnych textových dopravných značiek z týchto kandidátov vyjadruje stĺpec *Skutočný počet*. V ďalších stĺpcoch máme uvedený počet správne vyhodnotených - kandidát je textová dopravná značka, nesprávne vyhodnotených - kandidát je vyhodnotený ako textová dopravná značka, ale nemal by byť a počet nenájdených - kandidát by mal byť textová dopravná značka, ale nie je. Celkovú úspešnosť v stĺpci *Úspešnosť* sme získali ako pomer správne vyhodnotených k počtu všetkých dopravných značiek v percentách.

5.4 Celková úspešnosť detekcie textových dopravných značiek

Tabuľka 5.4: Vyhodnotenie detegovania textových dopravných značiek.

Farba	**Skutočný p.**	**Správne *TP***	**Nesprávne *FP***	**Nenájdené *FN***	**Úspešnosť (%)**
Modrá	60	51	4	9	85,0%
Zelená	44	42	3	2	95,5%
Hnedá	19	14	1	5	73,7%
Celkovo	**123**	**107**	**8**	**16**	**87,0%**

Celkové vyhodnotenie detekcie textových dopravných značiek je uvedené v tabuľke 5.4. V stĺpci *Skutočný počet* sa nachádza reálny počet všetkých textových dopravných značiek na testovacích videách. V stĺpci *Správne* sú uvedené hodnoty, ktoré sme dosiahli

pri testovaní našej aplikácie. Stĺpec *Nesprávne* vyjadruje počet nesprávne vyhodnotených objektov, ktoré nie sú textová dopravná značka. V stĺpci *Nenájdené* sú počty nenájdených textových dopravných značiek. Výsledný stĺpec *Úspešnosť* sme získali ako pomer správne detegovaných ku všetkým textovým dopravným značkám.

Celková úspešnosť detekcie je 87,0%. Najväčšiu percentuálnu úspešnosť sme dosiahli pri zelených dopravných značkách - 95,5%, a naopak najmenšiu pri hnedých - 73,7%. Pri modrých značkách je úspešnosť 85,0%.

Výhodou zelených značiek je fakt, že všetky tieto značky sú veľkých rozmerov. Príčinou nízkej úspešnosti pri hnedej farbe môže byť nedostatok hnedých značiek v testovacej množine alebo problémy určenia správnych hodnôt pri prahovaní, ktoré sa prejavili pri horšom počasí. Z tohto dôvodu sme získali pri hľadaní veľa nesprávnych kandidátov.

Záver

Cieľom tejto bakalárskej práce bolo implementovať aplikáciu, ktorá dokáže na videu detegovať textové dopravné značky, čo sa nám aj podarilo. V prvej časti sme načrtli úvod do problematiky, v ktorom sme ukázali históriu dopravných značiek, ich rozdelenie a rozdiely v rôznych krajinách. Potom sme predstavili podobné práce a existujúce riešenia, vďaka ktorým sme získali informácie o rôznych problémoch, ktoré mali autori pri riešení.

V druhej časti sme predstavili rôzne operácie spracovania obrazu. Postupne sme opísali najpoužívanejšie farebné modely, rozlíšenia obrazu a problémy pri jeho získavaní, rôzne metódy predspracovania obrazu a segmentácie.

V ďalšom kroku sme navrhli vlastné riešenie problematiky, v ktorom sme postupovali krok po kroku a prácu sme rozdelili na dva veľké celky.

Na základe tohto návrhu sme postupovali v implementácii, kde sme opísali použité nástroje, použité metódy s hodnotami a užívateľské prostredie.

Po implementácii sme otestovali čas a úspešnosť metód a krokov. Tieto hodnoty sme vyjadrili v tabuľkách.

Počas tvorby aplikácie nám napadli rôzne vylepšenia, ktoré by mohli navýšiť úspešnosť detegovania, ale aj rozšíriť funkcionalitu aplikácie.

Jedným z týchto vylepšení by mohlo byť vytvorenie verzie tejto aplikácie pre platformu Android. Toho vylepšienie by zvýšilo komfort pre užívateľa, ktorý by nemusel získané videá nahrávať do počítača, ale detekcia by bežala naživo v telefóne. Navyše pri implementácii by sme použiť knižnicu OpenCV, pretože existuje verzia kompatibilná s Androidom.

Medzi vylepšenia, ktoré by rozšírili funkcionalitu aplikácie patrí napríklad odčítanie textu z dopravnej značky a vypísanie alebo prečítanie tohto textu. Toto rozšírenie by sa dalo použiť pri šoférovaní, keď vodič nie vždy dokáže prečítať všetky informácie na dopravnej značke.

Literatúra

[1] Vyhláška MV SR č. 9/2009 Z. z. o cestnej premávke a o zmene a doplnení niektorých zákonov v znení neskorších predpisov. `www.epi.sk/zz/2009-9`. [21.01.2019].

[2] Martin Klebaško. Vplyv opatrení dopravnej polície na bezpečnosť cestnej premávky. Diplomová práca, 2004. Akadémia Policajného zboru v Bratislave.

[3] Bicycle Dutch. Road signs for cycling in the Netherlands. `bicycledutch.wordpress.com/2012/06/04/road-signs-for-cycling-in-the-netherlands/`. [21.01.2019].

[4] Miloš Fabian. Rozpoznávanie dopravných značiek. Bakalárska práca, 2012. Univerzita Komenského v Bratislave, Fakulta Matematiky, Fytiky a Informatiky.

[5] Anikó Szabóová. Detekcia a rozpoznávanie dopravných značiek. Bakalárska práca, 2017. Univerzita Komenského v Bratislave, Fakulta Matematiky, Fytiky a Informatiky.

[6] Štefan Toth. Rozpoznávanie dopravných značiek a ich použitie v mapových aplikáciách. Bakalárska práca, 2011. Žilinská univerzita v Žiline.

[7] Elena Šikudová, Zuzana Černeková, Wanda Benešová, Zuzana Haladová, and Júlia Kučerová. *Počítačové videnie Detekcia a rozpoznávanie objektov*. Vydavateľstvo Wikina, 2011.

[8] Luis David Lopez and Olac Fuentes. Color-Based Road Sign Detection and Tracking. In *ICIAR*, 2007.

[9] A. Ibraheem Noor, M. Hasan Mokhtar, Z. Khan Rafiqul, and K. Mishra Pramod. Understanding Color Models: A Review. *ARPN Journal of Science and Technology*, 2(3), April 2012. [21.01.2019].

[10] Števo Porubský. Čo je 4K a čo Ultra HD? *Denník N*, Február 2016. [21.01.2019].

[11] Rafael C. Gonzalez and Richard E. Woodws. *Digital Image Processing*. Prentice Hall, 2002. Second Edition.

[12] Zoltán Tomori and Matej Nikorovič. *Počítačové videnie v praxi*. Ústav experimentálnej fyziky SAV, 2016.

[13] Khairi Reda, Victor Mateevitsi, and Catherine Offord. A human-computer collaborative workflow for the acquisition and analysis of terrestrial insect movement in behavioral field studies. *EURASIP Journal on Image and Video Processing*, 2013:48, 12 2013. doi: 10.1186/1687-5281-2013-48.

[14] Rafael C. Gonzalez, Richard E. Woodws, and Steven L. Eddins. *Morphological Reconstruction From Digital Image Processing Using MATLAB,*. MATLAB Digest, 2010. Academic Edition.

[15] Šonka, Hlaváč, and Boyle. *Image processing, Analysis and Machine Vision*. International Thomson Publishing Inc., 1999. Second Edition.

[16] Canny edge detection. `https://docs.opencv.org/4.0.1/da/d22/tutorial_py_canny.html`. [21.01.2019].

[17] Hough line transform. `docs.opencv.org/4.0.1/d9/db0/tutorial_hough_lines.html`. [21.01.2019].

[18] Python. `www.python.org/about/`, . [21.01.2019].

[19] Python 3.7.2. `docs.python.org/3.7/`, . [21.01.2019].

[20] Pycharm. `www.jetbrains.com/pycharm/?var=landing&utm_source=quora&utm_medium=cpc&utm_campaign=pycharm`. [21.01.2019].

[21] Opencv. `opencv.org/about.html`, . [21.01.2019].

[22] Opencv 4.0.1. `docs.opencv.org/4.0.1/`, . [21.01.2019].

[23] scilit-image. `scikit-image.org/docs/0.15.x/`. [06.05.2019].

[24] Tkinter. `docs.python.org/3.7/library/tkinter.html`. [06.05.2019].