Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Automatická anotace digitálních snímků

Plzeň 2013 Milan Medvec

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. května 2013

Milan Medvec

Abstract

Automatic Annotation of Digital Images

The goal of this master's thesis is to design and implement algorithms for automatic annotation of digital images. The software is written in C/C++ programming language and uses the Qt cross-platform library for manipulation with multimedia. The work is divided into four parts. The first part deals with the theory needed for good understanding of the solved problem. The second part specifically formulates the goal of this thesis. In the third part we describe the application design and discuss its implementation features. The fourth part of the work concentrates on the algorithms for annotation of four chosen objects. The algorithm design is discussed in further details there together with an analysis of the obtained results. Additionally, we can find the user's manual describing how to compile the application from the source code and how to use the software in the appendix of the thesis.

Obsah

1 Úvod

Každý den nás stále více obklopují moderní informační technologie – od mobilních telefonů, tabletů a přenosných počítačů, přes stolní počítače, až po velké výpočetní stanice. Neustále dochází k nárůstu jejich výpočetního výkonu, kapacity paměťových médií a přenosových rychlostí mezi jednotlivými zařízeními.

Tento trend nám čím dál více umožňuje ukládat informace v grafické podobě, tedy takové, která je pro člověka nejpřijatelnější. Jak se uvádí v odborné literatuře, zrak člověku zprostředkovává 80 až 90 % informací o okolním prostředí. Výpočetní stroje však v současné době této formě informace nerozumí, a proto je nutné vyvinout nástroje, které jim to umožní. Tímto problémem se zabývá obor počítačového vidění – jeden z významných vědních oborů současnosti. Metody počítačového vidění se stále více uplatňují v nejrůznějších oborech, jakými jsou lékařství, doprava, robotika, armáda, atd.

Jednou z možností použití počítačového vidění je kategorizace položek v databázích obrázků a videí. K tomu je však zapotřebí vědět, co každý snímek nebo video obsahuje. V současné době se popis fotografií provádí převážně ručně. Cílem této práce je vytvořit program pro automatickou anotaci digitálních snímků.

Předkládaná práce je rozdělena do čtyř částí. V první části probereme teoretické znalosti potřebné k pochopení dané problematiky. Zde je vysvětlen princip anotací a jsou popsány použité metody zpracování snímků. Zaměříme se na použité barevné prostory, vybrané metody předzpracování, segmentaci snímků, analýzu a klasifikaci objektů. V druhé části práce formulujeme řešenou úlohu, definujeme konkrétně cíl práce a provedeme hrubý nástin řešení. Třetí část je věnována návrhu architektury softwaru pro automatickou anotaci digitálních snímků a jeho implementaci. V poslední, čtvrté části práce, popíšeme návrh a implementaci algoritmů sloužících k anotaci vybraných objektů. Na tomto místě zhodnotíme dosažené výsledky a nastíníme další možná vylepšení navržených algoritmů.

V příloze je umístěna stručná dokumentace, která popisuje způsob překladu vyvinutého softwaru a příklad jeho použití. Dále je zde uvedena tabulka zdrojů fotografií použitých v této práci.

2 Teoretický rozbor

V této kapitole se zaměříme na anotace a zmíníme problémy spojené se zpracováním digitálních snímků. Dále probereme metody předzpracování a segmentace snímků a také metody analýzy a klasifikace objektů na snímku.

2.1 Anotace snímků

Podívejme se nyní na to, co anotace jsou, proč je vytvářet a k čemu mohou být užitečné. Dále si ukážeme, jaké jsou způsoby jejich tvorby a jak lze anotace ukládat pro další použití.

2.1.1 Co jsou anotace

Anotace si obecně můžeme představit jako dodatečná data, která jsou přidružená k nějakému dokumentu. V oblasti digitálních dokumentů se jedná především o digitální snímek, video nebo text. Samotné anotace jsou většinou v textové podobě. Hlavní účel anotací spočívá v uchování dodatečných informací, které se k našemu dokumentu přímo vztahují a mohou nám být později užitečné, ale samotný dokument je neobsahuje.

Jak vidíme, možnosti použití anotací jsou poměrně rozsáhlé. V dalším textu se však budeme soustředit výhradně na digitální snímek. Příkladem může být fotografie vytvořená běžným digitálním fotoaparátem. Zde jsou anotace velmi často užívány k uchování informací o fotoaparátu a jeho nastavení (závěrka, délka expozice, ohnisková vzdálenost apod.) nebo také k uložení popisu, původu a jiných podrobností o fotografii.

2.1.2 Proč anotace používat

Pro názornost si nejprve uveďme krátký příklad. Každý, kdo využívá informační technologie, se setkává s problémem rychlého vyhledávání informací. V případě textu jsou vyhledávací algoritmy poměrně sofistikované a efektivní, při vyhledávání obrázků však narážíme na problém. Představme si, že chceme vyhledat digitální snímek, který zobrazuje několik osob v přírodě. V současné době se k vyhledávání snímků velmi často používá tzv. hashování. Snímky se zmenší na velmi malou velikost (např. 16×16 pixelů) a následně se porovnávají. Takové vyhledávání je sice

velmi rychlé, ale také nepřesné. Aby bylo vyhledávání efektivní, je třeba vědět, co každý snímek (v databázi) zobrazuje. Pokud tedy k obrázku vytvoříme anotace, získáme textový popis scény a obrázky můžeme snadněji vyhledávat použitím algoritmů pro vyhledávání textů. V ideálním případě můžeme např. v databázi obrázků najít ty, které zachycují podobnou anebo dokonce stejnou scénu, ale např. z nepatrně jiného úhlu. Anotace jsou tedy velmi užitečnou pomůckou, a to nejen při práci s digitálními snímky.

2.1.3 Jak anotace vytvářet

Nyní se zaměřme na to, jak takové anotace vytvořit. Podle způsobu tvorby je můžeme dělit do tří skupin: **manuální**, **poloautomatické** a **plně automatické**. Každou skupinu si nyní popíšeme a ohodnotíme ji z hlediska několika – dle mého názoru nejdůležitějších – výkonnostních parametrů. Těmi jsou: *propustnost* – počet anotovaných snímků za jednotku času (zvolme hodinu), *kvalita anotací* – procentuální chybovost a *účinnost* – podíl plochy oblastí, které se podařilo anotovat, k celkové ploše snímku.

Manuální anotace

Manuální anotace jsou prováděny výhradně člověkem. Ten vybírá části snímku, které zná a opatřuje je popisem. Snímek zpracovává do té doby, než projde všechny objekty obsažené ve snímku. Výhodou ruční tvorby anotací je bezesporu jejich správnost. Člověk se při pilné práci téměř nesplete a všechny vytvořené anotace budou korektní. Další velkou výhodou je, že budou probrány veškeré objekty na snímku a každé oblasti tak bude přiřazena alespoň jedna anotace. Tyto skvělé vlastnosti manuálních anotací jsou však vykoupeny velkou nevýhodou, jakou je vysoká časová náročnost práce. Dále je třeba si uvědomit, že pokud má člověk tuto činnost vykonávat, musí k tomu mít nějakou motivaci. V první řadě nás jistě napadne finanční ohodnocení. V takovém případě však výsledná práce byla někým zaplacena a tudíž ji pravděpodobně nebude chtít poskytnout k volnému použití např. na Internetu. Tím ji ale nebude možné využít v širším měřítku, což lze jistě také považovat za nevýhodu.

Co se týče výkonnostních parametrů, dosahuje tato metoda v současné době nejlepších výsledků z hlediska chybovosti (v případě kvalitní práce můžeme dosáhnout nulové chybovosti) a stoprocentní účinnosti (celý snímek bude anotován). Rychlost anotování je ovšem nízká. Za jednu hodinu je člověk schopen anotovat řádově pouze jednotky snímků.

Poloautomatické anotace

Dalším typem jsou poloautomatické anotace. Jedná se o částečně automatickou tvorbu anotací, kde je však stále zapotřebí lidské obsluhy. Ta v tomto případě vykonává korekci anotací navržených strojem, případně anotace těch oblastí, které nebylo možné označit automaticky. Jedná se o způsob, jak částečně urychlit manuální anotování, aniž bychom výrazně zhoršili jejich úspěšnost.

U poloautomatických anotací by tedy teoreticky měla být zachována nulová chybovost a stoprocentní účinnost, pokud bude obsluha důsledná a bude kontrolovat veškeré anotace vytvořené softwarem. Navíc by mělo dojít k nepatrnému urychlení, protože anotační software je schopen některé objekty označit automaticky. To ovšem velmi záleží na kvalitě daného softwaru. Urychlení je však nepatrné a rychlost bude i tak pouze několik desítek snímků za hodinu. Navíc nevýhoda v potřebě lidské obsluhy stále přetrvává.

Plně automatické anotace

Třetí skupinu tvoří plně automatická tvorba anotací, která představuje ideální stav, jak anotace vytvářet. Bohužel v současné době ještě není možné tento přístup naplno využít, neboť nebyly vyvinuty dostatečně kvalitní systémy, které by byly schopné úplně nahradit lidskou obsluhu. Velkou výhodou je rychlost zpracování jednotlivých snímků, která se může pohybovat řádově v desítkách až ve stovkách snímků za jednu hodinu. Automatické anotace snímku tak lze provádět ihned po jeho pořízení, resp. anotace a pořízení záznamu nemusí být oddělené. Záznamové zařízení může v ideálním případě rovnou pořizovat anotovaný snímek. Další výhodou je, že lze kdykoli zvýšit propustnost pouhým přidáním dalšího výpočetního stroje, což u manuálního anotování není možné (resp. není až tak jednoduché).

Z hlediska výkonnosti dosahujeme tedy touto metodou vyšší propustnosti, avšak chybovost je v současném stavu nesrovnatelně vyšší, a stejně tak i oblast, kterou nebylo možné anotovat, je větší. Tyto dvě vlastnosti však velmi závisejí na kvalitě navržených algoritmů.

2.1.4 Jak anotace ukládat

Vzhledem k tomu, že se zaměřujeme pouze na anotace digitálních snímků, máme pouze dvě možnosti, jak vytvořené anotace ukládat. Tyto možnosti jsou: uložení do souboru samotného snímku, uložení mimo soubor snímku.

Interní uložení

V případě uložení vytvořených anotací do souboru snímku využíváme toho, že formát souboru umožňuje kromě obrazových dat uložit také metadata. Tento způsob je však závislý na typu formátu snímku, protože ne každý formát může metadata obsahovat, resp. může se lišit způsob jejich uložení. Nevýhodou je tedy závislost zpracování a struktury dat na formátu snímku. Výhodou však zůstává fakt, že data jsou ke snímku pevně přidružena a nemůže dojít k jejich oddělení. Nejběžnější formáty metadat jsou EXIF a ITPC.

EXIF
Umožňuje ukládat mnoho doplňkových informací, včetně náhledu snímku. V novějších verzích specifikace je umožněno ukládat také libovolná uživatelská data ve formátu XML. Metadata jsou v EXIFu ukládána strukturovaně, takže není omezena jejich velikost. Tato specifikace je ovšem podporována pouze u souborových formátů JPEG a TIFF, ve formátech JPEG 2000 a PNG podporována není.

ITPC
Jedná se o standard, který definuje ukládání textových dat opět do obrazových souborů typu JPEG nebo TIFF. Používá se zejména u fotobank při ukládání informací užitečných k vyhledávání. Firma Adobe jej také využívá k popisování obsahu grafických souborů v bitmapovém editoru Adobe Photoshop.

Externí uložení

V případě ukládání anotací mimo soubor snímku máme možnost vytvořit si soubor nový. Potom je struktura souboru nezávazná a můžeme ji zvolit dle svých vlastních požadavků (samotnou strukturou se na tomto místě zatím nebudeme zabývat). Soubor slouží pouze k uložení anotací a neobsahuje tak žádné další rušivé informace, což je velmi užitečné při dalším zpracování. Nevýhodou však je skutečnost, že máme dva soubory, které nejsou fyzicky nijak spojené. Je tedy potřeba zvolit nějaký vhodný způsob pojmenování nebo ukládání souborů.

2.2 Metody zpracování snímků

V této podkapitole uvedeme některé základní pojmy používané průběžně v celé práci, popíšeme vybrané barevné prostory a použité metody zpracování snímků.

2.2.1 Základní pojmy

Nejprve si uveďme některé pojmy, které budeme během práce používat, a je důležité, aby byly jednoznačně definovány. Poznamenejme, že vše uvedené v této podkapitole se bude vztahovat pouze k diskrétním prostorům.

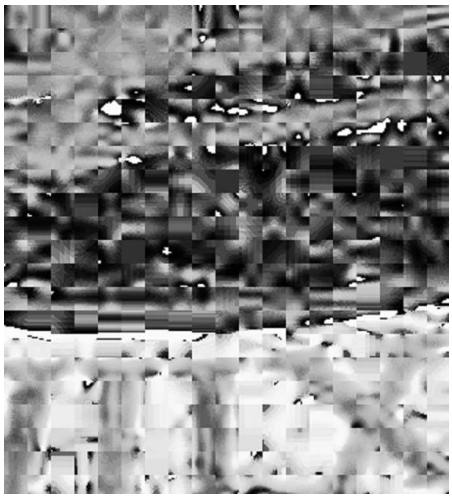
Digitální snímek

Pojmem digitální snímek (nebo krátce snímek) či fotografie budeme v této práci označovat soubor, který obsahuje digitalizovanou vizuální podobu libovolné scény. Připomeňme, že takový snímek se skládá z obrazových bodů (pixelů[1]), kde každý bod má svoji barvu a samozřejmě také polohu, přičemž celkový počet bodů udává tzv. rozlišení snímku. V dnešní době je na trhu mnoho typů a také mnoho výrobců záznamových zařízení, které slouží k tvorbě snímků. Na tomto místě je zapotřebí připomenout, že snímací charakteristiky jednotlivých zařízení se odlišují, tudíž i pořízené snímky jsou rozdílné, což významně ovlivňuje úspěšnost použitých algoritmů. Z tohoto důvodu je vhodné předem vymezit jakousi „oblast působnosti“ a algoritmy navrhnout pouze pro vybraný typ snímků. Abychom však ostatní snímky úplně nezavrhli, pokusíme se pomocí vhodného předzpracování každý nevyhovující snímek upravit tak, aby splňoval vymezené požadavky. Z hlediska použitých algoritmů tím dojde k jakési „normalizaci“ či sjednocení vlastností vstupních snímků a ke zjemnění rozdílů mezi nimi.

Poznamenejme, že digitální snímek může být uložen v mnoha souborových formátech, přičemž každý nakládá s obrazovými daty jiným způsobem. Dle formátu se může lišit způsob uložení, typ zdrojového barevného prostoru, barevná hloubka atp. Odlišnosti spojené s ukládáním dat nás ale ve zpracování snímku neomezují, protože k obrazovým datům budeme přistupovat pomocí jednotného rozhraní (vizte odstavec 4.1). Problémem ovšem může být způsob manipulace s daty před samotným uložením – především ztrátová komprese dat, která je některými formáty používána. Ta je založena na nedokonalostech lidského oka, díky kterým je možné provést redukci dat, aniž by se perceptuálně zhoršila kvalita obrazu. Ve skutečnosti se však kvalita obrazu zhorší, což někdy znemožní použití některých algoritmů. Typickým příkladem může být použití hranového operátoru nad daty z formátu JPEG. Citlivý detektor je schopen detekovat nejen změny jasu v obrazové scéně, ale i přechody mezi jednotlivými bloky, resp. makrobloky, které se při kompresi využívají. Důsledky ztrátové komprese je možné objevit i při převodu do jiného barevného prostoru, např. v prostorech HSV, HSL se často objevují artefakty způsobené kompresí (vizte obrázek 2.1).

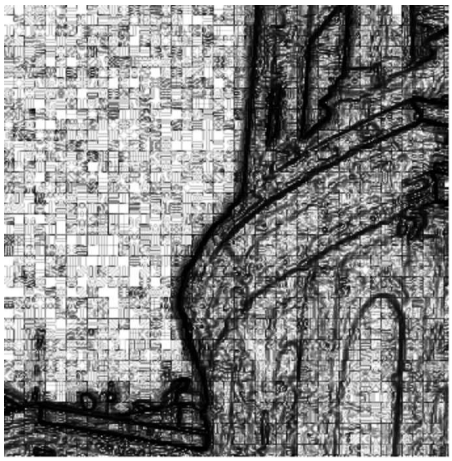
[1]Termín pixel vznikl zkrácením anglických slov picture element – obrazový prvek. Používá se k označení nejmenší jednotky rastrové grafiky.

(a) Původní obrázek



(b) Odstín barvy v prostoru HSV

(c) Původní obrázek



(d) Detekce hran citlivým detektorem

Obrázek 2.1: Artefakty způsobené JPG kompresí. Obrázek (b) ukazuje, že odstín barvy se mezi jednotlivými bloky nemění plynule. Na obrázku (d) pak lze vidět výsledek hranového detektoru. Je zde sice patrný šum ve snímku, čtvercový vzor je však dán ztrátovou kompresí.

Barva a barevná hloubka

Barva je v digitálním podání reprezentována n-rozměrným vektorem. Velikost vektoru a význam jednotlivých složek (tzv. barevných kanálů) je dán příslušným barevným prostorem, kterým se budeme věnovat v podkapitole 2.2.2. Barevná (někdy označovaná jako bitová) hloubka udává počet bitů použitých k uložení buď celého vektoru nebo jeho jednotlivých složek. Nejčastěji se používá 8, 12, 14 a 16 bitů na kanál. V celé naší práci avšak budeme používat barevnou hloubku pouze 8 bitů, která vyplývá ze zvoleného způsobu implementace (vizte odstavec 4.1).

Histogram snímku

Pojmem histogram snímku označujeme grafické znázornění četností výskytu obrazových bodů v závislosti na jejich jasu. Stejně tak jako četnost i histogram snímku rozdělujeme na absolutní a relativní. Absolutní četnost udává celkový počet bodů ve snímku s daným jasem, zatímco relativní četnost výskytu odpovídá poměrnému zastoupení jasu (podíl absolutní četnosti jasu a celkového počtu bodů). Můžeme tedy říci, že relativní histogram udává pravděpodobnost výskytu jednotlivých jasů ve snímku. Poznamenejme, že u barevných snímků se stanovuje pro každý barevný kanál samostatný histogram.

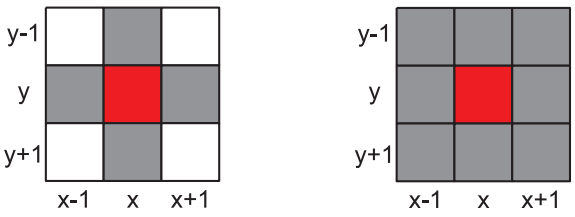
Okolí bodu

Pro bod $p = [x, y]$ označme okolí bodu jako množinu $\delta([x, y])$. Nejčastěji se používá tzv. čtyř- nebo osmi-okolí, které značíme 4-δ resp. 8-δ. Jejich přesná definice je následující:

$$4\text{-}\delta([x,y]) = \{[x-1,y],[x+1,y],[x,y-1],[x,y+1]\}, \tag{2.1a}$$

$$8\text{-}\delta([x,y]) = 4\text{-}\delta([x,y]) \cup \{[x-1,y-1],[x+1,y+1],[x+1,y-1],[x-1,y+1]\}. \tag{2.1b}$$

Na obrázku 2.2 jsou tato okolí znázorněna graficky. Na závěr poznamenejme, že body z okolí nazýváme sousedy bodu p.



(a) Čtyř-okolí (b) Osmi-okolí

Obrázek 2.2: Čtyř- a osmi-okolí bodu $[x, y]$. Sousední body jsou označeny šedou barvou.

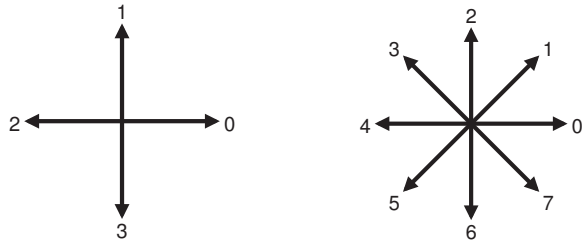
Oblast snímku

Pojmem cesta označujeme posloupnost bodů p_i pro $i = 1, \ldots, n$ takových, že pro každé i platí: p_i je sousedem bodu p_{i+1} ve smyslu zvoleného okolí. Délka takové cesty je rovna n. Oblastí snímku budeme označovat množinu obrazových bodů takových, že mezi každou dvojicí bodů existuje cesta, přičemž všechny body cesty náleží do

této oblasti. Oblast je tedy souvislá množina bodů. Poznamenejme, že v celé práci budeme uvažovat oblasti snímku pouze ve smyslu osmi-okolí.

Freemanův řetězový kód

Freemanův řetězový kód slouží k efektivnímu kódování křivek. Vychází z pozorování, že v diskrétním prostoru nám stačí znát pouze souřadnice počátečního bodu a směr dalšího vývoje křivky, souřadnice ostatních bodů nejsou nutné. Číselné kódování směru je prováděno dle tzv. směrové růžice, jejíž podobu pro čtyř- a osmi-okolí znázorňuje obrázek 2.3. Každý bod (vyjma počátečního) tak lze popsat jedním číslem z intervalu $\langle 0, 3\rangle$ resp. $\langle 0, 7\rangle$.



(a) Čtyř-okolí (b) Osmi-okolí

Obrázek 2.3: Směrové růžice Freemanova řetězového kódu

Momenty

Obecný moment řádu $p + q$ je definován jako:

$$M_{pq} = \sum_x \sum_y x^p y^q \, \mathrm{f}(x, y), \ \forall [x, y] \in D(\mathrm{f}), \tag{2.2}$$

kde $\mathrm{f}(x, y)$ označuje vstupní diskrétní dvourozměrnou funkci (resp. vstupní data) a $D(\mathrm{f})$ její definiční obor. Nevýhodou obecných momentů je, že nejsou invariantní vůči translaci, rotaci a změně měřítka. Z tohoto důvodu se zavádí další momenty. Jako první z nich si uvedeme tzv. *centrální moment*, který je invariantní vůči translaci:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q \, \mathrm{f}(x, y), \ \forall [x, y] \in D(\mathrm{f}),$$

přičemž $\bar{x}$ a $\bar{y}$ představuje tzv. *centroid*. Ten lze stanovit pomocí obecných momentů jako:

$$\bar{x} = \frac{M_{10}}{M_{00}}, \ \bar{y} = \frac{M_{01}}{M_{00}},$$

kde M_{pq} jsou dány rovnicí (2.2). Centrální momenty lze dále rozšířit pro řády $p+q \geq 2$ tak, aby byly invariantní vůči změně měřítka:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\left(1+\frac{p+q}{2}\right)}}.$$

Je možné zavést také další rozšíření zajišťující invarianci vůči rotaci, to již ale uvádět nebudeme.

Momenty mohou být velmi užitečné při popisu objektů nalezených ve scéně. V takovém případě může funkce f odpovídat např. hodnotám jasu popisovaného objektu nebo ji lze definovat jako funkci příslušnosti bodu $[x, y]$ k objektu O:

$$\mathrm{f}(x, y) = \begin{cases} 1, & [x, y] \in O, \\ 0, & [x, y] \notin O. \end{cases}$$

2.2.2 Barevný model a barevný prostor

Barevný model a *barevný prostor* definují matematický způsob reprezentace barev. Každá barva je popsána pomocí n-rozměrného vektoru, přičemž význam tohoto vektoru je dán barevným modelem. Ten popisuje obecnou reprezentaci barev, zatímco prostor většinou označuje již konkrétní barvy. Poznamenejme, že na jednom barevném modelu může být založeno více prostorů. V opačném případě, kdy je na jednom modelu založen pouze jeden prostor, se rozdíl mezi nimi poněkud stírá. V této práci budeme používat barevné prostory RGB, YUV, Lab, HSL a HSV, které k reprezentaci barvy vždy používají třísložkový vektor. Tyto prostory si nyní stručně představíme a uvedeme potřebné převodní vztahy.

RGB

V barevném modelu RGB je každá barva vyjádřena jako směs základních barev červené (R), zelené (G) a modré (B). Číselně je barva reprezentována uspořádanou trojicí hodnot, které odpovídají množství základních barev a to v rozmezí nulové až maximální intenzity. Konkrétní číselné ohodnocení je dáno vybranou barevnou hloubkou. Pro názornost uveďme příklad pro hloubku 8 bitů, v takovém případě lze odlišit 256 různých úrovní jednoho barevného kanálu, přičemž se používá notace, že 0 označuje nulovou intenzitu základní barvy a 255 plnou intenzitu. Trojice $[0, 0, 0]$ pak odpovídá černé barvě, zatímco trojice $[255, 255, 255]$ barvě bílé. Poznamenejme, že v praxi se často provádí normalizace použitých hodnot na interval $\langle 0, 1 \rangle$, čímž se zajistí nezávislost vůči zvolené barevné hloubce.

Na základě modelu RGB je založeno mnoho různých RGB prostorů, které se liší především rozsahem přípustných barev[2] a bílým bodem[3]. Protože v práci používáme přístup k obrazovým datům, který je nezávislý na formátu vstupního snímku, není možné během zpracování zjistit, o jaký prostor se v dané fotografii jedná. V našem případě budeme předpokládat prostor sRGB, který je v digitálních fotografiích nejrozšířenější. Další prostory tak budeme vztahovat právě k sRGB.

YUV

YUV není samostatný barevný prostor, ale jedná se pouze o způsob kódování prostoru RGB. Smysl jednotlivých složek je následující: složka Y udává jas barvy, složka U je úměrná rozdílu $\mathrm{B}-\mathrm{Y}$ a složka V je dána rozdílem $\mathrm{R}-\mathrm{Y}$. Toto kódování zavádíme proto, abychom se u složek U a V zbavili závislosti na jasu. V ideálním případě by tak tyto komponenty odpovídaly pouze chrominanci barvy, ve skutečnosti však vždy bude existovat nepatrná závislost na jasu Y. Převod mezi barevným prostorem YUV a RGB je následující:

$$\begin{aligned}
\mathrm{Y} &= W_R\,\mathrm{R} + W_G\,\mathrm{G} + W_B\,\mathrm{B}, \quad \mathrm{R},\mathrm{G},\mathrm{B},\mathrm{Y} \in \langle 0,1\rangle,\\
\mathrm{U} &= U_{Max}\tfrac{\mathrm{B}-\mathrm{Y}}{1-W_B}, \quad \mathrm{U} \in \langle -U_{Max}, U_{Max}\rangle,\\
\mathrm{V} &= V_{Max}\tfrac{\mathrm{R}-\mathrm{Y}}{1-W_R}, \quad \mathrm{V} \in \langle -V_{Max}, V_{Max}\rangle,
\end{aligned}$$

$$\begin{aligned}
\mathrm{R} &= \mathrm{Y} + (1-W_R)\tfrac{\mathrm{V}}{V_{Max}},\\
\mathrm{G} &= \mathrm{Y} - \tfrac{\mathrm{U}}{U_{Max}}\tfrac{W_B(1-W_B)}{W_G} - \tfrac{\mathrm{V}}{V_{Max}}\tfrac{W_R(1-W_R)}{W_G},\\
\mathrm{B} &= \mathrm{Y} + (1-W_B)\tfrac{\mathrm{U}}{U_{Max}}.
\end{aligned}$$

kde W_R, W_G, W_B jsou váhové koeficienty, přičemž aby byla zachována intenzita barvy, musí platit rovnost $W_R + W_G + W_B = 1$. Hodnoty těchto koeficientů jsme zvolili dle doporučení Rec. BT.601[4] následovně:

$$\begin{aligned}
W_R &= 0{,}299,\\
W_G &= 0{,}587,\\
W_B &= 0{,}114,\\
U_{Max} &= 0{,}436,\\
V_{Max} &= 0{,}615.
\end{aligned}$$

[2]Množina všech barev, které lze v rámci daného barevného prostoru popsat se často označuje termínem *gamut*.

[3]Pojmem bílý bod označujeme nejsvětlejší barvu, kterou lze v daném prostoru vygenerovat.

[4]Doporučení lze nalézt na adrese `http://www.itu.int/rec/R-REC-BT.601`.

Samotné převodní vztahy je pro zvolené hodnoty možné zapsat v přijatelnější maticové formě:

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{U} \\ \mathrm{V} \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}14713 & -0{,}28886 & 0{,}436 \\ 0{,}615 & -0{,}51499 & -0{,}10001 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix}, \tag{2.3}$$

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \begin{bmatrix} 1{,}0 & 0{,}0 & 1{,}13983 \\ 1{,}0 & -0{,}39465 & -0{,}58060 \\ 1{,}0 & 2{,}03211 & 0{,}0 \end{bmatrix} \begin{bmatrix} \mathrm{Y} \\ \mathrm{U} \\ \mathrm{V} \end{bmatrix}.$$

HSV a HSL

Prostory HSL a HSV reprezentují RGB prostor pomocí válcových souřadnic, přičemž se snaží definovat barvy tak, aby co nejlépe odpovídaly lidskému vnímání barev a zároveň zůstaly výpočetně jednoduché. Barva je zde popsána pomocí tří charakteristických vlastností barvy – odstínu, sytosti a světlosti barvy v případě HSL a odstínu, sytosti a jasu v případě HSV. Převod barevných složek z prostoru RGB do prostorů HSL a HSV je následující:

$$M = \max\{\mathrm{R}, \mathrm{G}, \mathrm{B}\},\ m = \min\{\mathrm{R}, \mathrm{G}, \mathrm{B}\},$$

$$\mathrm{H} = \begin{cases} \text{nedefinováno}, & \text{jestliže } M = m, \\ 60° \cdot \frac{\mathrm{G}-\mathrm{B}}{M-m} + 0°, & \text{jestliže } M = \mathrm{R} \text{ a } \mathrm{G} \geq \mathrm{B}, \\ 60° \cdot \frac{\mathrm{G}-\mathrm{B}}{M-m} + 360°, & \text{jestliže } M = \mathrm{R} \text{ a } \mathrm{G} < \mathrm{B}, \\ 60° \cdot \frac{\mathrm{B}-\mathrm{R}}{M-m} + 120°, & \text{jestliže } M = \mathrm{G}, \\ 60° \cdot \frac{\mathrm{R}-\mathrm{G}}{M-m} + 240°, & \text{jestliže } M = \mathrm{B}, \end{cases}$$

$$\mathrm{S_{HSL}} = \begin{cases} 0, & \text{jestliže } \mathrm{L} = 0 \text{ nebo } M = m, \\ \frac{M-m}{M+m} = \frac{M-m}{2\,\mathrm{L}}, & \text{jestliže } 0 < \mathrm{L} \leq \frac{1}{2}, \\ \frac{M-m}{2-(M+m)} = \frac{M-m}{2-2\,\mathrm{L}}, & \text{jestliže } \mathrm{L} > \frac{1}{2}, \end{cases}$$

$$\mathrm{L} = \tfrac{1}{2}(M+m),$$

$$\mathrm{S_{HSV}} = \begin{cases} 0, & \text{jestliže } M = 0, \\ \frac{M-m}{M} = 1 - \frac{m}{M}, & \text{jinak}, \end{cases}$$

$$\mathrm{V} = M,$$

přičemž v uvedených vztazích se předpokládá $\mathrm{R}, \mathrm{G}, \mathrm{B} \in \langle 0, 1\rangle$. Pro výsledné barevné komponenty poté platí

$$\mathrm{H} \in \langle 0°, 360°\rangle,\ \mathrm{S_{HSL}}, \mathrm{L}, \mathrm{S_{HSV}}, \mathrm{V} \in \langle 0, 1\rangle,$$

v praxi se ovšem zavádí normalizace odstínu barvy H na interval $\langle 0, 1\rangle$. Z uvedených vztahů vidíme, že definice odstínu barvy je pro oba modely stejná, ale výpočet sytosti se mezi oběma modely liší. Převod z HSV zpět do RGB je následující:

$$\begin{aligned} H_i &= \left\lfloor \tfrac{\mathrm{H}}{60} \right\rfloor \bmod 6, \quad f = \tfrac{\mathrm{H}}{60} - H_i, \\ p &= \mathrm{V}(1 - \mathrm{S}), \\ q &= \mathrm{V}(1 - f\,\mathrm{S}), \\ t &= \mathrm{V}(1 + f\,\mathrm{S} - \mathrm{S}), \end{aligned}$$

$$[\mathrm{R}, \mathrm{G}, \mathrm{B}] = \begin{cases} [\mathrm{V}, t, p], & \text{jestliže } H_i = 0, \\ [q, \mathrm{V}, p], & \text{jestliže } H_i = 1, \\ [p, \mathrm{V}, t], & \text{jestliže } H_i = 2, \\ [p, q, \mathrm{V}], & \text{jestliže } H_i = 3, \\ [t, p, \mathrm{V}], & \text{jestliže } H_i = 4, \\ [\mathrm{V}, p, q], & \text{jestliže } H_i = 5. \end{cases}$$

Převod z HSL zpět do RGB je definován jako:

$$\begin{aligned} q &= \begin{cases} \mathrm{L}(1 + \mathrm{S}), & \text{jestliže } \mathrm{L} < \frac{1}{2}, \\ \mathrm{L} + \mathrm{S} - \mathrm{L}\,\mathrm{S}, & \text{jestliže } \mathrm{L} \geq \frac{1}{2}, \end{cases} \\ p &= 2\,\mathrm{L} - q H_k = \tfrac{\mathrm{H}}{360}, \end{aligned}$$

$$t_R = H_k + \frac{1}{3},\ t_G = H_k,\ t_B = H_k - \frac{1}{3},$$

pro $i = \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$:

$$t_i' = \begin{cases} t_i + 1{,}0 & \text{jestliže } t_i < 0, \\ t_i - 1{,}0 & \text{jestliže } t_i > 1, \\ t_i & \text{jinak}, \end{cases}$$

$$C_i = \begin{cases} p + 6t_i'(q - p), & \text{jestliže } t_i' < \frac{1}{6}, \\ q, & \text{jestliže } \frac{1}{6} \leq t_i' < \frac{1}{2}, \\ p + 6(q - p)(\frac{2}{3} - t_i'), & \text{jestliže } \frac{1}{2} \leq t_i' < \frac{2}{3}, \\ p, & \text{jinak}, \end{cases}$$

$$[\mathrm{R}, \mathrm{G}, \mathrm{B}] = [C_R, C_G, C_B].$$

XYZ

Prostor XYZ je nezávislý na zařízení a existuje na rozdíl od prostoru RGB pouze jeden. Z tohoto důvodu je pro každý RGB prostor definován speciální převodní vztah. V našem případě jsme použili převod platný pro barevný prostor sRGB:

$$\begin{bmatrix} \mathrm{R}' \\ \mathrm{G}' \\ \mathrm{B}' \end{bmatrix} = \begin{bmatrix} \mathrm{f}(\mathrm{R}) \\ \mathrm{f}(\mathrm{G}) \\ \mathrm{f}(\mathrm{B}) \end{bmatrix}, \mathrm{f}(t) = \begin{cases} \left(\frac{t+0{,}055}{1{,}055}\right)^{2{,}4} & \text{pokud } t > 0{,}04045, \\ \frac{t}{12{,}92} & \text{jinak,} \end{cases}$$

$$\begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{bmatrix} = \begin{bmatrix} 0{,}4124 & 0{,}3576 & 0{,}1805 \\ 0{,}2126 & 0{,}7152 & 0{,}0722 \\ 0{,}0193 & 0{,}1192 & 0{,}9505 \end{bmatrix} \begin{bmatrix} \mathrm{R}' \\ \mathrm{G}' \\ \mathrm{B}' \end{bmatrix},$$

$$\begin{bmatrix} \mathrm{R}' \\ \mathrm{G}' \\ \mathrm{B}' \end{bmatrix} = \begin{bmatrix} 3{,}2406 & -1{,}5372 & -0{,}4986 \\ -0{,}9689 & 1{,}8758 & 0{,}0415 \\ 0{,}0557 & -0{,}2040 & 1{,}0570 \end{bmatrix} \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{bmatrix},$$

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \begin{bmatrix} \mathrm{f}^{-1}(\mathrm{R}') \\ \mathrm{f}^{-1}(\mathrm{G}') \\ \mathrm{f}^{-1}(\mathrm{B}') \end{bmatrix}, \mathrm{f}^{-1}(t) = \begin{cases} 1{,}055\left(t^{\frac{1}{2{,}4}}\right) - 0{,}055 & \text{pokud } t > 0{,}0031308, \\ 12{,}92t & \text{jinak.} \end{cases}$$

Lab

Prostor Lab vznikl nelineární konverzí prostoru XYZ proto, aby lépe odpovídal lidskému vnímání barev. Stejně jako XYZ je i prostor Lab nezávislý na zařízení a existuje pouze jeden. Prostor Lab opět odděluje luminanci (složka L) a chrominanci barvy (složky a a b). Ke konverzi z XYZ je zapotřebí definovat referenční bílou barvu, v našem případě jsme si zvolili bod s označením D65, který má hodnoty $X_n = 95{,}047, Y_n = 100{,}000, Z_n = 108{,}883$. Samotný převod je následující:

$$\begin{aligned} \mathrm{L} &= 116\,\mathrm{f}(\mathrm{Y}/Y_n) - 16, \\ \mathrm{a} &= 500\left[\mathrm{f}(\mathrm{X}/X_n) - \mathrm{f}(\mathrm{Y}/Y_n)\right], \\ \mathrm{b} &= 200\left[\mathrm{f}(\mathrm{Y}/Y_n) - \mathrm{f}(\mathrm{Z}/Z_n)\right], \end{aligned}$$

$$\mathrm{f}(t) = \begin{cases} t^{1/3} & \text{pokud } t > (\frac{6}{29})^3, \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{jinak,} \end{cases}$$

$$\begin{aligned} \mathrm{Y} &= Y_n\,\mathrm{f}^{-1}\left(\tfrac{1}{116}\left(\mathrm{L}+16\right)\right), \\ \mathrm{X} &= X_n\,\mathrm{f}^{-1}\left(\tfrac{1}{116}\left(\mathrm{L}+16\right) + \tfrac{1}{500}\,\mathrm{a}\right), \\ \mathrm{Z} &= Z_n\,\mathrm{f}^{-1}\left(\tfrac{1}{116}\left(\mathrm{L}+16\right) - \tfrac{1}{200}\,\mathrm{b}\right), \end{aligned}$$

$$\mathrm{f}^{-1}(t) = \begin{cases} t^3 & \text{pokud } t > \frac{6}{29}, \\ 3\left(\frac{6}{29}\right)^2 \left(t - \frac{4}{29}\right) & \text{jinak.} \end{cases}$$

Šedotónové snímky

Šedotónový snímek je takový snímek, který obsahuje pouze odstíny šedi. Ten může vzniknout např. zaznamenáním pouze jasové složky barevného snímku. Jak jsme si ale mohli všimnout, výpočet jasu barvy není jednoznačný a každý z uvedných prostorů definuje jiný způsob výpočtu. Pokud tedy v nějakém algoritmu provádíme analýzu barevného snímku pouze na základě jasové informace, je vždy zapotřebí uvést vybraný způsob výpočtu jasu.

2.2.3 Předzpracování

Předzpracováním označujeme přípravu dat za účelem usnadnění dalšího zpracování nebo zvýšení jeho úspěšnosti. V této práci intenzivně využíváme barevnou informaci a je tudíž nutné, aby barvy ve snímku byly správně interpretovány. Použitím předzpracování se pokusíme zajistit jejich správné vyvážení. Předzpracováním můžeme také vstupní data převést do podoby, která je vhodnější k dalšímu zpracování. V našem případě budeme používat v každém algoritmu vybraný barevný prostor, který se bude pravděpodobně lišit od prostoru zdrojového. Změnu prostoru lze svým způsobem považovat také za předzpracování. Posledním druhem předzpracování, jež budeme používat, je filtrace obrazu.

Změna barevného prostoru

Ke změně barevného prostoru je zapotřebí projít vstupní snímek bod po bodu a transformovat hodnotu každého bodu samostatně. K převodu do vybraného prostoru použijeme vztahy uvedené v podkapitole 2.2.2.

Úprava histogramu

Úprava snímku na základě jeho histogramu (resp. úprava histogramu snímku) je zástupcem tzv. globálního předzpracování. To je charakterizováno právě tím, že snímek je zpracováván dle jeho globálních vlastností. V našem případě použijeme úpravu histogramu za účelem zlepšení vyvážení barev ve snímku. Tím se pokusíme zmírnit rozdíly mezi snímky, které byly vytvořeny různými záznamovými zařízeními.

Korekci barev provedeme nezávislou transformací barevných kanálů ve zvoleném barevném prostoru. Vybraný algoritmus je založen na předpokladu, že barva s nejvyšší hodnotou by měla odpovídat bílé barvě, zatímco barva s nejnižší hodnotou barvě černé. Každou složku tedy transformujeme pomocí lineární funkce tak, aby nejnižší (označme c_{min}), resp. nejvyšší (c_{max}) hodnota ve snímku odpovídaly minimální (m), resp. maximální (M) možné hodnotě. Matematicky lze tuto transformaci vyjádřit jako:

$$\mathrm{c}'(i) = \frac{\mathrm{c}(i) - c_{min}}{c_{max} - c_{min}}(M - m), \tag{2.4}$$

kde $\mathrm{c}(i)$ představuje původní hodnotu zpracovávané barevné složky i-tého bodu ve snímku a $\mathrm{c}'(i)$ hodnotu novou. Symbolem c označujeme vybranou barevnou složku, např. R, G nebo B pro barevný prostor RGB.

Při výběru krajních hodnot ve snímku však není vhodné volit absolutně nejmenší a největší hodnotu. Je třeba mít na paměti, že zpracovávaný snímek může obsahovat šum, který tyto hodnoty významně ovlivní. Je tedy lepší vybrat takové hodnoty, které mají ve snímku určitou váhu a jejichž výskyt není náhodný. V našem případě použijeme postup navržený v článku [10]. Zde je volena minimální hodnota c_{min} taková, že pravděpodobnost výskytu nižších hodnot je rovna parametru s_1 a maximální hodnota c_{max} je vybrána tak, aby pravděpodobnost výskytu vyšších hodnot byla rovna parametru s_2. Body na krajích histogramu jsou tedy ignorovány. Tento požadavek můžeme zapsat následovně:

$$\begin{aligned} \mathrm{p}(i < c_{min}) &= s_1, \\ \mathrm{p}(i > c_{max}) &= s_2. \end{aligned}$$

Ve snímku lze tyto hodnoty vyčíslit pomocí následujících vztahů:

$$\begin{aligned} c_{min} &= \max\{h \in \langle 0, 255\rangle : \frac{1}{N}\sum_{i=1}^{h} \mathrm{N_c}(i) < s_1\}, \\ c_{max} &= \min\{h \in \langle 0, 255\rangle : \frac{1}{N}\sum_{i=h}^{N} \mathrm{N_c}(i) > 1 - s_2\}, \end{aligned}$$

kde N je celkový počet bodů ve snímku a $\mathrm{N_c}(i)$ představuje počet bodů, jejichž barevná složka c má hodnotu i. Použitím těchto krajních hodnot ovšem některé body transformujeme dle vztahu (2.4) mimo povolený interval $\langle m, M\rangle$. Zaveďme tedy dodatečnou úpravu:

$$\mathrm{c}''_i = \begin{cases} c_{min} & \mathrm{c}'(i) < c_{min}, \\ \mathrm{c}'(i) & \mathrm{c}'(i) \in \{c_{min}, c_{max}\}, \\ c_{max} & \mathrm{c}'(i) > c_{max}, \end{cases}$$

která provede korekci těchto bodů. Protože víme, že pravděpodobnost jejich výskytu je rovna $s_1 + s_2$ je nutné volit parametry předzpracování velmi malé, aby těchto bodů nebylo mnoho. Tato korekce sice může způsobit nepřirozeně vypadající bílé či černé plochy, avšak v našem případě to nevadí, protože předzpracování používáme k usnadnění anotace snímků a ne za účelem vylepšení jejich kvality.

Na závěr poznamenejme, že výsledek algoritmu velmi záleží na zvoleném barevném prostoru, protože snímek je zpracováván po složkách a v každém barevném prostoru mají barevné složky odlišný význam. V našem případě jsme si zvolili prostor RGB s bitovou hloubkou 8 bitů, které odpovídají hodnoty $m = 0$ a $M = 255$.

Filtrace

Častým důvodem k filtraci obrazu je potlačení šumu a jiných nežádoucích poruch, které se ve snímku mohou vyskytnout mnoha různými způsoby. V našem případě však nebudeme filtrování provádět za tímto účelem, protože předpokládáme, že snímek předložený k anotaci má dostatečnou kvalitu a případný šum je téměř zanedbatelný. Hlavním důvodem, proč jsme vybrali předzpracování typu filtrace, je potlačení detailů (vysokých frekvencí) a zmenšení vlivu lokálních změn v jasu v obraze, čímž se významně usnadní segmentace, potažmo detekce objektů.

Existuje mnoho způsobů filtrování snímků a různých druhů filtrů, které lze k tomuto účelu použít. My si však vystačíme s pouhým konvolučním filtrem. Jedná se o metodu lokálního předzpracování, kde hodnota jednoho bodu je dána jeho lokálním okolím a určíme ji konvolucí okolí s hodnotami filtru. Matematicky lze obecnou dvourozměrnou diskrétní konvoluci zapsat jako:

$$(\mathrm{f} * \mathrm{h})(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \mathrm{f}(x - i, y - j) \cdot \mathrm{h}(i, j). \tag{2.5}$$

V případě filtrace předpokládejme, že funkce f odpovídá vstupnímu snímku o rozměru $m \times n$ a funkce h použitému filtru o rozměru $2k \times 2k$, přičemž platí, že $2k \cdot 2k \ll mn$. Poznamenejme, že definiční obory funkcí f a h předpokládáme ve tvaru:

$$\begin{aligned} D(\mathrm{f}) &= \langle 0, m\rangle \times \langle 0, n\rangle, \\ D(\mathrm{h}) &= \langle -k, k\rangle \times \langle -k, k\rangle. \end{aligned}$$

Pro takto zvolené funkce lze vztah (2.5) upravit do podoby:

$$(\mathrm{f} * \mathrm{h})(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \mathrm{f}(x - i, y - j) \cdot \mathrm{h}(i, j). \tag{2.6}$$

Nevýhodou přímého výpočtu je poměrně velká časová náročnost. Lze jej však výrazně urychlit použitím konvolučního teorému:

$$\mathcal{F}(\mathrm{f} * \mathrm{h}) = \mathcal{F}(\mathrm{f}) \cdot \mathcal{F}(\mathrm{h}) = F \cdot H,$$

přičemž $\mathcal{F}$ značí Fourierovu transformaci, F, H frekvenční obrazy funkcí f, h, a $F \cdot H$ Hadamardův[5] maticový součin. Abychom mohli konvoluční teorém použít, je nutné zajistit, že frekvenční obrazy F a H (resp. funkce f a h) mají stejný rozměr. Z tohodo důvodu definujeme pomocnou funkci h' o rozměru $m \times n$ reprezentující vybraný konvoluční filtr:

$$\mathrm{h}'(x, y) = \begin{cases} \mathrm{h}(x - k, y - k) & x < 2k \wedge y < 2k, \\ 0. \end{cases}$$

Vztah (2.6) lze pak vyjádřit pomocí inverzní Fourierovy transformace $\mathcal{F}^{-1}$ jako:

$$(\mathrm{f} * \mathrm{h})(x, y) = \mathcal{F}^{-1}(\mathcal{F}(\mathrm{f}) \cdot \mathcal{F}(\mathrm{h}')).$$

Experimenty ukazují, že tento postup je při vhodné implementaci diskrétní Fourierovy transformace rychlejší již pro rozměr vstupních dat $mn > 500$.

2.2.4 Segmentace

Segmentace je proces, jehož úkolem je rozdělit snímek na disjunktní oblasti, přičemž nejčastěji se využívá k oddělení objektů od pozadí nebo vzájemně mezi sebou. Metod provádějících segmentaci je velké množství, od těch nejjednodušších jako je *obyčejné prahování*, až po velmi sofistikované jako jsou *aktivní kontury*, *Mean-shift*, *GrowCut*, apod. (více o těchto algoritmech lze najít např. v knize [19]).

Existuje sice několik prací [21, 23, 17], které automatickou anotaci provádí s využitím těchto složitých segmentačních algoritmů, my je však používat nebudeme. V našem případě totiž nebudeme provádět segmentaci snímku podle jasu, ale podle vytvořeného pravděpodobnostního ohodnocení. Díky tomu můžeme předpokládat, že hledané objekty jsou ohodnoceny velmi podobně a můžeme je tak snáze segmentovat. K tomu se nám osvědčily metody automatického prahování, metoda narůstání oblastí a metoda rozšiřování oblastí.

Prahování

Prahování představuje nejjednodušší způsob segmentace. V případě prahování s jedním prahem t se vstupní snímek $\mathrm{f_I}$ segmentuje na dvě oblasti, povětšinou na objekty O a pozadí P. Výsledný segmentovaný snímek $\mathrm{f_O}$ je definován jako:

[5] Hadamardův maticový součin označuje násobení matic po prvcích.

$$\mathrm{f_O}(x,y) = \begin{cases} 0 \;\; (\text{tj. } [x,y] \in P), & \mathrm{f_I}(x,y) < t, \\ 1 \;\; (\text{tj. } [x,y] \in O), & \mathrm{f_I}(x,y) \geq t. \end{cases}$$

Hlavním problémem této metody je však volba prahu t. Existuje několik metod, které volí práh automaticky na základě analýzy snímku, jako např. *metoda Otsu*, *metoda minimální chyby*, *metoda entropie* atd. V našem případě budeme používat první jmenovanou, a proto si ji popíšeme podrobněji.

Otsu

Otsu metoda je jednou z metod automatického prahování s jedním prahem a poprvé byla prezentována v [12]. Vychází z předpokladu, že relativní histogram snímku je složen ze dvou normálních rozdělení, přičemž práh se snaží volit tak, aby rozptyl jednotlivých tříd byl maximální nebo rozptyl uvnitř tříd byl minimální.

Označme si nyní předpokládaná normální rozdělení C_0 a C_1 a pravděpodobnost, že daný jas spadá do jedné z těchto tříd $\mathrm{P}(C_i)$. Pak jistě platí:

$$p_i = \frac{N_i}{N}, \quad \sum_{i=1}^{N} p_i = 1,$$
$$\mathrm{P}(C_0) = \omega_0 = \sum_{i=1}^{k} p_i = \omega(k),$$
$$\mathrm{P}(C_1) = \omega_1 = \sum_{i=k+1}^{N} p_i = 1 - \omega(k),$$
$$\omega_0 + \omega_1 = 1.$$

Nyní se podívejme na parametry normálních rozdělení – střední hodnotu μ_i a rozptyl σ_i. Lze je stanovit jako:

$$\begin{aligned}
\mu_0 &= \sum_{i=1}^{k} i\,\mathrm{P}(i|C_0) = \sum_{i=1}^{k} i\frac{\mathrm{P}(i\cap C_0)}{\mathrm{P}(C_0)} = \sum_{i=1}^{k} i\frac{p_i}{\omega_0} = \frac{1}{\omega_0}\sum_{i=1}^{k} ip_i,\\
\mu_1 &= \sum_{i=k+1}^{N} i\,\mathrm{P}(i|C_1) = \sum_{i=k+1}^{N} i\frac{\mathrm{P}(i\cap C_1)}{\mathrm{P}(C_1)} = \sum_{i=k+1}^{N} i\frac{p_i}{\omega_1} = \frac{1}{\omega_1}\sum_{i=k+1}^{N} ip_i,\\
\sigma_0^2 &= \sum_{i=1}^{k}(i-\mu_0)^2\,\mathrm{P}(i|C_0) = \sum_{i=1}^{k}(i-\mu_0)^2\frac{p_i}{\omega_0},\\
\sigma_1^2 &= \sum_{i=k+1}^{N}(i-\mu_1)^2\,\mathrm{P}(i|C_1) = \sum_{i=k+1}^{N}(i-\mu_1)^2\frac{p_i}{\omega_1},
\end{aligned}$$

přičemž pravděpodobnost $\mathrm{P}(i\cap C_j)$ lze určit:

$$\mathrm{P}(i\cap C_j) = \begin{cases} p_i, & i\in C_j,\\ 0, & i\notin C_j. \end{cases}$$

Celkovou střední hodnotu a celkový rozptyl celého histogramu lze vyjádřit analogicky:

$$\begin{aligned}
\mu_T &= \sum_{i=1}^{N} ip_i,\\
\sigma_T^2 &= \sum_{i=1}^{N}(i-\mu_T)^2 p_i.
\end{aligned}$$

Vztah mezi středními hodnotami potom je: $\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$. Zbývá ukázat souvislost mezi rozptyly σ_0, σ_1 a σ_T:

$$\sigma_T^2 = \sum_{i=1}^{N}(i-\mu_T)^2 p_i = \sum_{i=1}^{k}(i-\mu_T)^2 p_i + \sum_{i=k+1}^{N}(i-\mu_T)^2 p_i =$$

$$= \sum_{i=1}^{k}(i-\mu_T+\mu_0-\mu_0)^2 p_i + \sum_{i=k+1}^{N}(i-\mu_T+\mu_1-\mu_1)^2 p_i =$$

$$= \underbrace{\sum_{i=1}^{k}(i-\mu_0)^2 p_i}_{\omega_0\sigma_0^2} + 2(\mu_0-\mu_T)\underbrace{\sum_{i=1}^{k}(i-\mu_0)p_i}_{\sum ip_i-\mu_0\sum p_i=0} + (\mu_0-\mu_T)^2\underbrace{\sum_{i}^{k} p_i}_{\omega_0} +$$

$$\underbrace{\sum_{i=k+1}^{N}(i-\mu_1)^2 p_i}_{\omega_1\sigma_1^2} + 2(\mu_1-\mu_T)\underbrace{\sum_{i=k+1}^{N}(i-\mu_1)p_i}_{\sum ip_i-\mu_1\sum p_i=0} + (\mu_1-\mu_T)^2\underbrace{\sum_{k+i}^{N} p_i}_{\omega_1} =$$

$$= \omega_0\sigma_0^2 + (\mu_0-\mu_T)^2\omega_0 + \omega_1\sigma_1^2 + (\mu_1-\mu_T)^2\omega_1 =$$

$$= \underbrace{\omega_0\sigma_0^2 + \omega_1\sigma_1^2}_{{\omega_\mathrm{W}}^2} + \underbrace{\omega_0(\mu_0-\mu_T)^2 + \omega_1(\mu_1-\mu_T)^2}_{{\omega_\mathrm{B}}^2}. \quad (2.7)$$

Jak vidíme, celkový rozptyl σ_T je dán součtem rozptylů uvnitř tříd σ_W a rozptylů mezi třídami σ_W (separovatelnost tříd). Dle zadání chceme rozptyl uvnitř tříd minimální a separovatelnost tříd maximální, tj:

$$\begin{aligned} {\omega_\mathrm{W}}^2(k*) &= \min_{1\le k\le N}({\omega_\mathrm{W}}^2(k)), \\ {\omega_\mathrm{B}}^2(k*) &= \max_{1\le k\le N}({\omega_\mathrm{B}}^2(k)). \end{aligned}$$

Protože nelze provést optimalizaci na základě obou kritérií, v praxi se často snažíme o maximalizaci separovatelnosti tříd C_0 a C_1. Hodnotu prahu k pak volíme tak, aby ${\omega_\mathrm{B}}^2(k)$ bylo maximální. Pro výpočet rozptylu ω_B^2 se používá výpočtový vzorec:

$${\omega_\mathrm{B}}^2(k) = \omega_0(\mu_0-\mu_T)^2 + \omega_1(\mu_1-\mu_T)^2 =$$

$$= \omega_0\omega_1(\mu_0-\mu_1)^2 = \frac{[\mu_T\,\omega(k)-\mu(k)]^2}{\omega(k)(Q-\omega(k))}. \quad (2.8)$$

Narůstání oblastí

Metoda narůstání oblastí se snaží nalézt ve snímku takové oblasti, jejichž body se vyznačují stejnou homogenitou. Pojmem homogenita označujeme ohodnocení bodů,

které je založeno na jejich jedné či více vlastnostech (např. jas, barva, textura apod.). Tato metoda spadá do tříd metod typu Split and Merge, algoritmus se tedy skládá z následujících dvou kroků:

- Snímek se rozdělí na disjunktní elementární oblasti, pro které platí, že splňují kritérium homogenity.
- Sousední elementární oblasti se dále spojují do větších oblastí tak, aby kritérium homogenity bylo opět splněno. Spojování probíhá dokud existují sousední elementární oblasti, které lze spojit. Výsledné oblasti jsou homogenní a zároveň maximální.

V našem případě však štěpení snímku přeskočíme a budeme uvažovat dělení po pixelech. Každá oblast tak bude tvořená právě jedním bodem a tím bude vždy homogenní. Více o této metodě lze nalézt např. v pracích [4, 19].

Rozšiřování oblastí

Rozšiřování oblastí se většinou používá ve spojení s poloprahováním. Poloprahováním označujeme segmentaci, kdy prahujeme jenom část snímku a ostatní ponecháme beze změny. V případě segmentace do dvou tříd lze poloprahování zapsat následovně:

$$\mathrm{f_O}(x,y) = \begin{cases} 0 \ \ (\text{tj. } [x,y] \in P), & \mathrm{f_I}(x,y) < t_1, \\ \mathrm{f_I}(x,y), & \mathrm{f_I}(x,y) \in \langle t_1, t_2 \rangle, \\ 1 \ \ (\text{tj. } [x,y] \in O), & \mathrm{f_I}(x,y) > t_2. \end{cases}$$

Jak vidíme, body s hodnotou v intervalu $\langle t_1, t_2 \rangle$ nejsou zařazeny ani do jedné třídy. Tyto body jsou pak na základě analýzy přiřazeny do jedné ze tříd P, resp. O, čímž rozšiřujeme oblasti získané poloprahováním.

2.2.5 Analýza objektů

Předpokládejme, že jistá podoblast (segment) snímku může odpovídat hledanému objektu. Abychom tuto hypotézu potvrdili resp. vyvrátili, je třeba nejprve provést analýzu segmentu. Úkolem analýzy je získání popisu segmentu (tzv. příznaků), podle kterého je možné provést následnou klasifikaci, přičemž jako příznaky jsou vybírány takové vlastnosti segmentu, které ho dostatečně popisují. V našem případě pracujeme s dvourozměrnými segmenty snímku, a proto se zaměříme na tzv. popisy

jednoduchých objektů. Protože některé popisy lze definovat více způsoby, uvedeme si u každého vybraného popisu také zvolený způsob výpočtu. Pro segment X lze stanovit následující vlastnosti:

- S – **velikost (plocha)**: Určíme ji jednoduše jako počet bodů segmentu:

$$S = \sum_{[x,y] \in X} 1. \tag{2.9}$$

- O – **obvod**: Počet bodů vnitřní hranice segmentu (vizte rovnici (2.12)).
- T – **plošné těžiště**: Určíme jako podíl momentů prvního řádu a momentu nultého řádu (vizte podkapitolu 2.2.1), tj:

$$T = [\bar{x}, \bar{y}] = \left[\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right]. \tag{2.10}$$

- α – **směr**: Směr delší strany nejmenšího opsaného obdélníku.
- P – **pravoúhlost**: Podíl plochy segmentu a nejmenšího opsaného obdélníku. Platí $P \in \langle 0, 1 \rangle$.
- D – **podlouhlost**: Poměr mezi délkou a šířkou nejmenšího opsaného obdélníku.
- k – **kruhovitost (nekompaktnost)**: Definujeme jako:

$$k = 4\pi \frac{S}{O^2}, \tag{2.11}$$

  přičemž platí $k \in \langle 0, 1 \rangle$. Kruhovitost představuje míru podobnosti objektu a ideálního kruhu, jehož hodnota je:

$$k = 4\pi \frac{\pi r^2}{(2\pi r)^2} = 1.$$

Podívejme se dále na několik algoritmů, které lze využít k analýze obrazové scény.

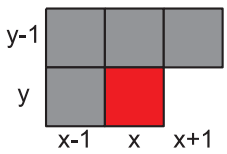
Connected Component Labeling

Algoritmus *Connected Component Labeling* slouží k detekci oblastí (komponent) v binárním snímku[6]. Každou nalezenou komponentu označí jednoznačným identifikátorem – štítkem. Existuje mnoho implementací tohoto algoritmu, v našem případě

[6]Binárním snímkem označujeme snímek obsahující pouze dvě barvy. Většinou jedna barva přísluší pozadí a druhá objektům.

použijeme dvouprůchodový algoritmus s použitím osmi-okolí (vizte rovnici (2.1b)). Princip tohoto algoritmu je následující.

V prvním průchodu se po řádcích analyzuje vstupní binární snímek bod po bodu. V případě, že aktuálně zkoumaný bod není prvkem pozadí, ale spadá do nějaké neidentifikované komponenty, provede se test okolí dle masky na obrázku 2.4. Maska je definována tak, aby se vyhodnocovaly pouze ty body z osmi-okolí, které byly zpracovány již v předchozích iteracích. Pokud se v okolí nachází jedna či více již identifikovaných komponent, použije se v daném bodě jeden z nalezených štítků a všechny štítky se označí jako ekvivalentní – všechny jsou součástí stejné komponenty. Obrázek 2.5 ukazuje členitou komponentu, jejímž částem se v prvním průchodu přiřadí rozdílné štítky. V případě, že v okolí není nalezena žádná komponenta, přiřadí se danému bodu nový štítek. Druhý průchod algoritmu slouží ke sjednocení všech štítků, které byly označeny jako ekvivalentní. Tím se každé komponentě přiřadí vždy jeden unikátní štítek.



Obrázek 2.4: Testovací maska CCL algoritmu.

Obrázek 2.5: Štítky přířazené jedné komponentě po prvním průchodu. Štítky jsou reprezentovány šedou barvou, černá barva symbolizuje pozadí.

Aproximace objektu elipsou

Předpokládejme obecný objekt $X = \{[x_i, y_i] : i = 1, \ldots, N\}$, který se nachází na obrazové scéně. Nyní si ukážeme postup, jak tento objekt jednoduše aproximovat obecnou dvourozměrnou elipsou (označme si ji písmenem E). Na základě článku [14] víme, že použitím centrálních momentů druhého řádu μ_{pq} (vizte podkapitolu 2.2.1) pro funkci příslušnosti f:

$$\mathrm{f}(x, y) \quad = \quad \begin{cases} 1, & [x, y] \in X, \\ 0, & [x, y] \notin X, \end{cases}$$

lze stanovit kovarianční matici:

$$C = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix}, \quad \mu'_{pq} = \mu_{00}\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}},$$

která obsahuje dostatek informací pro konstrukci elipsy. Z definice použitých momentů vyplývá, že matice C je reálná a symetrická, tudíž musí mít právě dvě nezáporná vlastní čísla λ_1, λ_2. Tato vlastní čísla odpovídají druhé mocnině velikosti hlavní a vedlejší poloosy elipsy E, tj:

$$k\lambda_1 = a^2,\ k\lambda_2 = b^2.$$

Úhel natočení elipsy je pak dán vlastním vektorem a lze ho vyjádřit jako:

$$\varphi = \frac{1}{2}\arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right).$$

Protože vlastní čísla poloosám pouze odpovídají, je zapotřebí určit koeficient k, abychom znali jejich skutečnou velikost. Určíme ho tak, abychom minimalizovali chybu aproximace. V tomto případě chybu definujeme jako počet bodů, které přísluší oblasti O a nacházejí se mimo elipsu, a bodů uvnitř elipsy, které naopak oblasti O nepřísluší:

$$\begin{aligned} I =& |X| + |Y|, \\ & X = \{p = [x, y], p \in E \wedge p \notin O\}, \\ & Y = \{p = [x, y], p \notin E \wedge p \in O\}. \end{aligned}$$

Předpokládejme, že tato chyba je nejmenší právě tehdy, když plocha elipsy bude rovna ploše oblasti, tj $S_O = S_E = \pi ab$. Koeficient k pak lze stanovit jako:

$$\begin{aligned} k\lambda_1 = a^2,\ k\lambda_2 = b^2, \\ \frac{\lambda_1}{\lambda_2} = \left(\frac{a}{b}\right)^2,\ b\sqrt{\frac{\lambda_1}{\lambda_2}} = a, \\ S_O = S_E = \pi ab = \pi b^2\sqrt{\frac{\lambda_1}{\lambda_2}}, \\ b^2 = \left(\frac{S_O}{\pi\sqrt{\lambda_1/\lambda_2}}\right), \\ k = \frac{b^2}{\lambda_2} = \left(\frac{S_O}{\pi\sqrt{\lambda_1\lambda_2}}\right). \end{aligned}$$

Jak již bylo zmíněno výše, vlastní čísla jsou nezáporná, a jejich odmocnění lze vždy provést. Uvedeným způsobem jsme tedy získali všechny parametry potřebné

pro konstrukci elipsy ve tvaru:

$$\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 = 1,$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x - \bar{x} \\ y - \bar{y} \end{bmatrix},$$

přičemž $\bar{x}$ a $\bar{y}$ značí střed elipsy, který je totožný s těžištěm objektu, a lze ho určit podle vztahu (2.10).

Algoritmus nalezení vnitřní hranice

Pro oblast X lze definovat vnitřní $\mathrm{H_i}(X)$ a vnější $\mathrm{H_o}(X)$ hranici oblasti jako následující množiny bodů:

$$\begin{aligned} \mathrm{H_i}(X) &= \{q = [x,y] : q \in X \wedge \exists p \in \delta(q) \wedge p \notin X\}, \\ \mathrm{H_o}(X) &= \{q = [x,y] : q \notin X \wedge q \in \delta(p) \wedge p \in \mathrm{H_i}(X)\}. \end{aligned} \tag{2.12}$$

V této praci však budeme používat pouze vnitřní hranici a to s použitím osmi-okolí (vizte rovnici (2.1b)). Nyní popíšeme algoritmus sloužící k jejímu nalezení. Nejprve je třeba nalézt počáteční bod p_0 a definovat počáteční směr prohledávání d_0. Počáteční bod je dán předpisem:

$$\begin{aligned} p_0 = &[x', y'], \\ &x' = \min\{x : (x, y') \in X\}, \\ &y' = \min\{y : (x, y) \in X\}, \end{aligned}$$

jedná se tedy o první bod, na který narazíme při skenování snímku zleva do prava a shora dolů. Směr prohledávání je kódován podle Freemanovy směrové růžice (vizte podkapitolu 2.2.1) a hodnota počátečního směru d_0 je pro osmi-okolí rovna 5. Algoritmus hledání bodů hranice je následující:

1. Nastavíme $k = 0$, $\mathrm{H_i}(X) = \{p_0\}$ a přejdeme na krok 2.
2. Bod hranice p_{k+1} hledáme ve směru d_k od bodu p_k. Jestliže nenarazíme na bod objektu X, přejdeme na krok 2, jinak na krok 3.
3. Pokračujeme v kladném směru prohledávání, tj. nastavíme $d_k = d_k + 1 \bmod 8$ a přecházíme na krok 2.
4. Pokud je nalezený bod p_{k+1} roven počátečnímu bodu p_0, přejdeme na krok 5. V opačném případě přidáme nově nalezený bod do hranice:

$$\mathrm{H_i}(X) = \mathrm{H_i}(X) \cup \{p_{k+1}\},$$

nastavíme směr hledání pro následující iterace:

$$d_{k+1} = \begin{cases} (d_k + 7) \bmod 8, & \text{pro } d_k \text{ sudé,} \\ (d_k + 6) \bmod 8, & \text{pro } d_k \text{ liché,} \end{cases}$$

zvýšíme hodnotu iterace $k = k + 1$ a pokračujeme krokem 2.

5. Algoritmus nalezl konec hranice a končí. Délka nalezené hranice $\mathrm{H_i}$ je rovna $k + 1$.

2.2.6 Klasifikace objektů

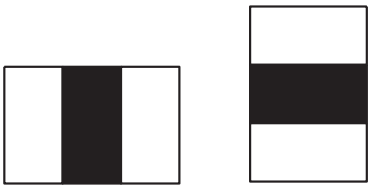
Klasifikace slouží k zařazení objektů do jedné z předem zvolených tříd. K výběru třídy jsou použity informace získané popisem objektu (tzv. *příznaky*). V této práci budeme klasifikátor používat k ověření, zda testovaná část snímku představuje hledaný objekt, či nikoliv. Budeme tedy provádět klasifikaci do dvou tříd – objekt/pozadí (klasifikace do dvou tříd bývá často nazývána *dichotomickou klasifikací*). K detekci objektů si vybereme dva typy klasifikátorů: *Viola-Jones detektor* (AdaBoost klasifikátor) a *jednoduchý kaskádní klasifikátor*.

Viola-Jones detektor

Detektor Viola-Jones slouží k detekci objektů v šedotónových snímcích a poprvé byl uveden v [22]. Skládá se ze tří základních částí: předzpracování snímku – výpočtu integrálního obrazu, výpočtu vybraných příznaků a klasifikace. Jak vidíme, nejedná se tedy o klasifikátor v pravém slova smyslu, ovšem klasifikace je stěžejní části detektoru, a proto jsme jeho popis zařadili do této kapitoly.

Podívejme se nejprve na princip použitého klasifikátoru. V základní implementaci se ke klasifikaci používá tzv. *algoritmus AdaBoost*, který spadá do kategorie algoritmů strojového učení. Základem tohoto klasifikátoru jsou tzv. *slabí žáci*, tedy klasifikátory, jejichž úspěšnost je nepatrně vyšší než 50 %, tj. vyšší než úspěšnost náhodné klasifikace. Každý z těchto slabých klasifikátorů provádí klasifikaci na základě jednoho vybraného příznaku. Hodnota příznaků se stanovuje konvolucí vstupního snímku se zvolenou maskou. Příklady masek znázorňuje obrázek 2.6, kde bílá plocha odpovídá hodnotám +1 a černá plocha hodnotám −1. Výsledný příznak je tedy rozdílem součtů hodnot jasů v černé a bílé části snímku. Dle velikosti a pozice masky ve vstupním snímku pak lze vygenerovat více různých příznaků. V práci [15] je uvedeno, že pro vstupní snímek velikosti 19×19 pixelů je možné vygenerovat přibližně 64 tisíc různých příznaků. Velikost vstupního snímku 19×19 pixelů je stanovena jako kompromis mezi výpočetní náročností a dostatečným optickým rozlišením objektu.

(a) Haarovy příznaky.



(b) Čárové příznaky

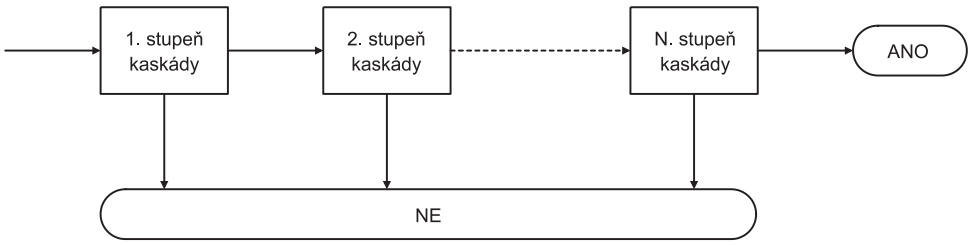
(c) Diagonální příznak.

Obrázek 2.6: Příznaky použité ve Viola-Jones detektoru.

K dosažení požadované úspěšnosti klasifikace je poté z tohoto velkého množství vybrán určitý počet nejvýznamnějších příznaků a každý je zpracováván jedním slabým klasifikátorem. Seskupením těchto klasifikátorů (jejich lineární kombinací) vznikne tzv. *silný žák*, čímž označujeme nelineární klasifikátor o téměř libovolné přesnosti v rámci trénovací množiny dat. Přesnost silného klasifikátoru se zvyšuje s počtem použitých slabých klasifikátorů (v praktických aplikacích jich bývá použito řádově několik stovek). Volba příznaků a nastavení klasifikátoru se provádí během trénovací fáze algoritmu.

Výpočetní náročnost algoritmu je dále snížena využitím předzpracování. Pomocí integrálního obrazu vstupního snímku lze každý příznak rychle a efektivně stanovit a není zapotřebí využívat konvoluci, pro bližší výklad vizte např. [15].

Nastiňme si nyní stručně princip samotného detektoru. Vzhledem k tomu, že dopředu neznáme pozici ani velikost hledaného objektu, zkoumá detektor všechny přípustné možnosti. Pro každou pozici jsou ze snímku vybrána podokna všech velikostí a ta jsou následně předána klasifikátoru k rozhodnutí, zda se v nich daný objekt vyskytuje. Tento postup je samozřejmě výpočetně velmi náročný, dle práce [15] je pro vstupní snímek rozměru 320×240 pixelů zapotřebí klasifikovat přibližně 500 000 podoken. Z tohoto důvodu se při hledání objektů ve snímku velmi často neuvažuje rotace, která by nároky opět výrazně zvýšila. Poznamenejme, že v praxi



Obrázek 2.7: Schématické znázornění kaskádního klasifikátoru.

se detektor používá ve velmi obměněné formě, která zaručuje podstatně vyšší výkon. Zde jsme si uvedli pouze základní myšlenku detektoru.

Kaskádní klasifikátor

Kaskádní zapojení klasifikátorů se používá v případě, kdy je výrazně vyšší pravděpodobnost, že testovaný snímek nezachycuje hledaný objekt. Tento klasifikátor pracuje tak, aby každý stupeň kaskády přijal co nejvíce pozitivních případů (snímek odpovídá objektu) a zamítl část negativnách případů (snímek neodpovídá objektu). K tomu, aby byl objekt detekován, je nutné aby prošel pozitivní detekcí všemi kaskádami. K zamítnutí vzoru stačí, aby byl zamítnut pouze jednou kaskádou. Každá kaskáda může provádět klasifikace na základě všech zvolených příznaků. Podobu kaskádního klasifikátoru schématicky znázorňuje obrázek 2.7.

V našem případě budou jednotlivé kaskády tvořeny jednoduchými klasifikátory, které budou provádět klasifikaci vždy na základě jednoho vybraného příznaku. K tomuto účelu jsme si vybrali binární klasifikátor, který lze matematicky popsat následující rovnicí:

$$\mathrm{y}(p) = \mathrm{sgn}(p - t),$$

kde $t \in R$ představuje zvolený práh klasifikace a $p \in R$ reprezentuje vybraný příznak. Výstup klasifikátoru $y \in \{-1, 1\}$ reprezentuje třídu odpovídající příznaku p. Hodnoty -1, $+1$ označují pozadí resp. hledaný objekt.

2.3 Návrh pravděpodobnostního modelu barev

Pojmem *pravděpodobnostní model barev* označujeme pravděpodobnostní funkci diskrétní náhodné veličiny popisující pravděpodobnost, že daná barva patří vybranému objektu. V této části textu si stručně nastíníme, jak takový model vytvořit. K tomu potřebujeme mít dostatečně velkou množinu snímků (dále trénovací množina) zachycující zvolený objekt. Abychom zajistili co největší invarianci modelu vůči vybraným

snímkům, je vhodné použít co nejméně specifické snímky (tj. takové, které jsou pořízeny různými zařízeními, za různých světelných podmínek apod.).

2.3.1 Relativní histogram četnosti

Jako první sestrojme relativní histogram četnosti barev obsažených v trénovací množině. Každou barvu v histogramu reprezentujeme dle barevného prostoru pomocí číselného vektoru. Dále pro popis barvy předpokládejme pouze třírozměrný vektor, v takovém případě bude mít výsledný histogram četnosti také tři rozměry. Tímto způsobem získáme odhad pravděpodobnosti výskytu každé barvy a histogram bychom již mohli považovat za aproximaci pravděpodobnostní funkce. Je třeba si ale dát pozor na to, aby histogram odpovídal reálnému rozložení pravděpodobnosti. Vlivem špatně zvolené trénovací množiny (např. malý počet vzorků, nerovnoměrné rozložení barev atp.) se může stát, že pravděpodobnost bude dosahovat relativně velkého rozptylu na poměrně malém okolí. Pravděpodobnost v jednom bodě modelu bude vysoká, ale v sousedních bodech bude nabývat malých hodnot a naopak. Z těchto důvodu může být vhodnější použít aproximaci histogramu vybraným pravděpodobnostním rozdělením nebo histogram upravit např. filtrováním.

2.3.2 Obecné normální rozdělení

V mnoha pracích (např. [13, 7]) se pravděpodobnostní model vytváří aproximací relativního histogramu normálním rozdělením $\mathrm{N}(\mu, \sigma)$. Třírozměrné normální rozdělení lze popsat následující rovnicí:

$$\mathrm{N}(\mathbf{x}, \mu, \sigma) = \frac{1}{\sqrt{(2\pi)^2 |\mathbf{C}|}} \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{C}^{-1}(\mathbf{x}-\mu)},$$

přičemž $\mathbf{x} = [x_1, x_2, x_3]$ reprezentuje barvu dle vybraného barevného prostoru. Velkou výhodou použití této aproximace je, že normální rozdělení lze jednoznačně identifikovat pouze pomocí dvanácti parametrů:

$$\mu = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 \end{bmatrix}, \sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix},$$

a není zapotřebí ukládat celý relativní histogram četnosti. Tento postup můžeme použít ale pouze v případě, že relativní histogram četnosti se svým tvarem podobá vybranému rozdělení a lze ho aproximovat s malou chybou. V případě, že tomu tak není, výsledná aproximace nemusí dostatečně odpovídat reálnému rozložení pravděpodobnosti.

2.3.3 Směs normálních rozdělení

Jako poslední způsob tvorby modelu uvedeme přístup navržený v [2]. Zde je pravděpodobnostní model vytvořen jako směs normálních rozdělení s pevně daným rozptylem σ_k a střední hodnotou definovanou dle trénovací množiny. Každý bod množiny odpovídá jednomu normálnímu rozdělení, přičemž střední hodnota je dána barvou bodu. Absolutní hodnota modelu v bodě $[x, y, z]$ je pak dána vztahem:

$$\mathrm{M}([x, y, z]) = \sum_{i=1}^{N} e^{-\frac{1}{2}\frac{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}{\sigma_k^2}},$$

přičemž samotnou pravděpodobnost stanovíme až normalizací modelu na interval $\langle 0, 1\rangle$ jako:

$$\mathrm{p}([x, y, z]) = \frac{\mathrm{M}([x, y, z])}{\sum_{\forall [x,y,z] \in D(\mathrm{M})} \mathrm{M}([x, y, z])},$$

kde N odpovídá počtu bodů v trénovací množině a $D(\mathrm{M})$ definičnímu oboru modelu M. Pro parametr σ_k je v [2] navrhovaná hodnota v rozmezí $\langle 11, 19\rangle$.

3 Formulace úlohy

Na tomto místě bych rád provedl rozbor zadání a nastínil možnosti, které se nám naskýtají při jeho řešení. Jak již bylo zmíněno v úvodu práce, naším úkolem je vytvořit počítačový program pro automatickou anotaci snímků, s cílem opatřit snímky pořízené digitálním fotoaparátem popisem stručně charakterizujícím jejich obsah. Protože toto zadání je velmi obecné, stanovíme v této kapitole konkrétní cíl práce a provedeme hrubý návrh aplikace.

3.1 Cíl práce

Doposud jsme anotaci snímku chápali jako textový popis scény, aniž by bylo přesně řečeno, jak takový popis vypadá. Protože existuje více způsobů popisu scény, je potřeba definovat jeho konkrétní podobu. V této práci jsem se rozhodl charakterizovat snímek pomocí objektů, které obsahuje. Automatická anotace snímků pak bude spočívat v identifikaci objektů na scéně a jejich popsání. *Cílem této práce je tedy výběr několika vhodných objektů a návrh a implementace algoritmů sloužících k jejich detekci na snímku.*

3.2 Nástin řešení

3.2.1 Anotace

V našem případě bude anotace představovat popis jednoho nalezeného objektu, přičemž můžeme předpokládat, že každý objekt na snímku lze identifikovat jeho typem a polohou. Anotace se tedy budou skládat právě z těchto dvou hodnot. Co se týče popisu obrazové scény, v ideálním případě chceme, aby každé oblasti příslušela alespoň jedna anotace. V praxi se však samozřejmě může stát, že nějaká oblast snímku nebude zobrazovat žádný předmět (bude například odpovídat pozadí), a tudíž mu nemusí příslušet žádné označení. Dále je třeba si uvědomit, že nějaká část snímku může odpovídat více objektům. Výsledné anotace tedy nemusí být disjunktní a mohou se překrývat. Mezi nalezenými objekty tak bude existovat geometrický a někdy také logický vztah. Abychom však při vyhodnocování takových vazeb předešli případnému vnášení chyb do generovaného výsledku, budeme zpracovávat každou anotaci samostatně a na tyto vazby nebudeme brát ohled.

3.2.2 Možnosti automatické anotace

Jak jsme již předeslali v úvodu kapitoly, automatickou anotaci budeme provádět identifikací objektu na snímku. Nyní se blíže podíváme na to, jak lze identifikaci provést. V ideálním případě bychom tento proces založili na segmentaci snímku, jejíž úkolem by bylo separovat všechny objekty od pozadí. Následným rozpoznáváním těchto segmentů bychom mohli získat seznam objektů na snímku. Implementace tohoto postupu je však v dnešní době velmi obtížná, ba dokonce nemožná. Velký problém představuje již samotná segmentace snímku. Přestože existují velmi sofistikované segmentační metody, i tak není snadné dosáhnout vždy dobrých výsledků, neboť každému typu objektu může vyhovovat odlišný přístup segmentace. Neméně náročným úkolem je následná klasifikace objektů, resp. jejich rozpoznávání[1].

Proto se v této práci pokusím o přístup opačný. Místo toho, abychom se snažili určit, co daná část snímku zobrazuje, budeme daný objekt hledat cíleně. Takový objekt si zvolíme předem, a to nezávisle na vstupním snímku. Při jejich výběru nejsme ničím omezeni a můžeme si tak vybrat objekty libovolného charakteru. Vybraný objekt pak můžeme důkladně analyzovat, díky čemuž ho budeme velmi dobře znát a ve snímku ho pak můžeme teoreticky nalézt nezávisle na velikosti, natočení, barvě apod. Výhodou tohoto postupu je teoreticky přesnější anotace. V případě, že ho ve snímku nenalezneme, můžeme s velkou pravděpodobností říct, že tam prostě není, což může být v mnoha aplikacích také velmi užitečné. Takový postup se samozřejmě již využívá, ovšem většinou pouze ve spojení s algoritmy strojového učení (jako příklad uveďme neuronové sítě, boost algoritmy apod.). V takovém případě se navíc za účelem zjednodušení algoritmu velmi často využívá pouze jasová informace obrazu. V této práci se však pokusíme k detekci objektů využít pravděpodobnostní přístup, při kterém budeme využívat především barevnou informaci snímku.

Zvoleným postupem jsme ve snímku samozřejmě schopni nalézt pouze objekty na které jsme se předem zaměřili, čímž je také dán maximální počet typů anotací, které jsme schopni ke snímku přiřadit. Univerzálnost takového postupu se pak bude zvyšovat s počtem vybraných objektů.

3.2.3 Návrh programu

V této podkapitole stručně nastíníme, jak budeme navržený způsob automatické anotace realizovat algoritmicky. Protože anotace snímku spočívá v samostatné detekci objektů, rozhodl jsme se celý software implementovat pomocí tzv. modulové architektury. Každý anotační algoritmus provádějící detekci vybraného typu objektu

[1]Klasifikací označujeme proces, kdy objekty zařazujeme do předem daných tříd. Při rozpoznávání nejsou třídy definovány a vznikají při samotném procesu.

bude zapouzdřen do samostatného modulu. Hlavní program pak bude schopen načíst a aplikovat všechny dostupné moduly a sám žádné anotace provádět nebude. Díky tomu je možné ho navrhnout a implementovat odděleně, aniž bychom předem znali princip jednotlivých modulů. Výhodné je, že celý program lze kdykoli rozšířit přidáním dalších modulů, aniž by bylo nutné ho dále jakkoli upravovat.

Vzhledem k tomu, že aplikace a anotační moduly jsou navzájem nezávislé, budeme se v následujících kapitolách zabývat jejich návrhem a implementací také odděleně.

4 Software

Nyní již přikročíme k popisu návrhu a implementaci samotného softwaru. Protože některé detaily návrhu vycházejí z použité implementace, podíváme se nejprve na použité softwarové prostředky. Poté se zaměříme na návrh architektury softwaru, definici důležitých datových struktur a použitých aplikačních rozhraních, provedeme návrh hlavní aplikace, jejího uživatelského rozhraní a formátu vstupních a výstupních souborů. V závěru kapitoly stručně popíšeme zásadní části implementace softwaru.

4.1 Použité softwarové prostředky

K implementaci softwaru jsem zvolil nízkoúrovňový jazyk C/C++ a pro práci s multimediálními daty jsem se rozhodl použít multiplatformní knihovnu **Qt**[1] verze 4.8.2 od firmy Nokia. Ta poskytuje mnoho užitečných nástrojů, které celou implementaci velmi usnadní a také zpřehlední. Knihovna je dostupná pod licencí LGPL a v našem případě ji tedy lze použít zdarma. Během implementace jsem dále použil také knihovnu **alglib**[2] verze 3.7.0. Ta nabízí velmi mnoho nástrojů převážně z oblasti numerické analýzy a zpracování dat, v našem případě využijeme algoritmy lineární algerby a matematické statistiky. Tato knihovna je dostupná stejně jako Qt projekt v licenci GPL. Dále jsem použil knihovnu **OpenCV**[3] verze 2.3.1, která poskytuje mnoho nástrojů z oblasti počítačového vidění a je pro nás tedy nesmírně užitečná. Knihovna OpenCV je dostupná s licencí BSD, a pro akademické účely je zdarma. K implementaci Fourierovy transformace jsem použil knihovnu **FFTW**[4] verze 3.3.3. Ta je dostupná pod licencí GNU GPL a pro naše účely je opět zdarma.

Programátorskou dokumentaci k navrženému softwaru jsem vytvořil pomocí nástroje **doxygen**[5] verze 1.8.3.1. UML diagramy obsažené v této práci byly vytvořeny ze zdrojového kódu pomocí programu **StarUML**[6] verze 5.0.2.1570.

[1] Více informací o projektu Qt lze nalézt na stránkách `http://qt.digia.com/`.
[2] Domovské stránky knihovny jsou `http://www.alglib.net/`.
[3] Projekt OpenCV je dostupný na `http://opencv.org/`.
[4] Knihovnu lze najít na `http://www.fftw.org/`.
[5] Nástroj doxygen je možné dohledat na `www.doxygen.org/`.
[6] Domovská stránka programu StarUML je `http://staruml.sourceforge.net/`.

4.2 Návrh aplikace

4.2.1 Architektura

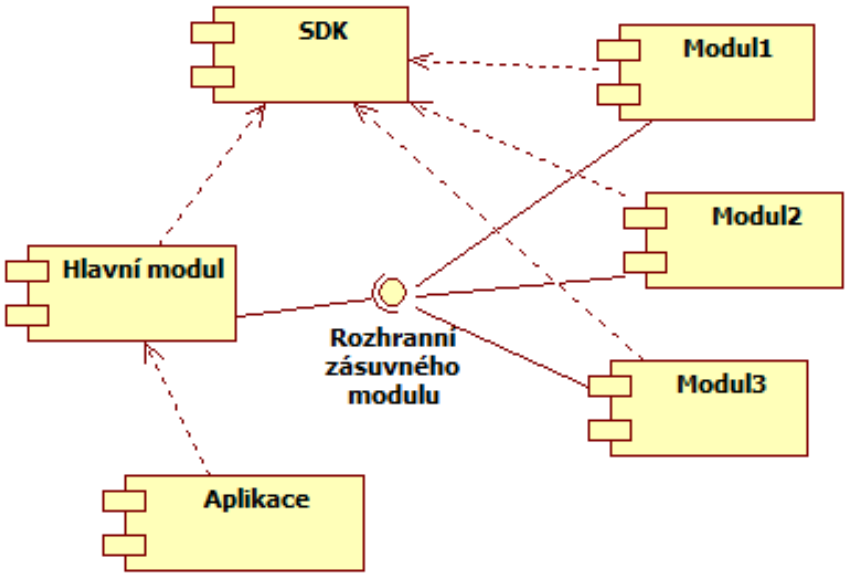
V kapitole 3 jsme během návrhu programu nastínili, že základem architektury softwaru bude aplikace, kterou lze rozšířit pomocí zásuvných modulů. Abychom zajistili vyšší univerzálnost vytvořeného softwaru, rozdělíme aplikaci z pohledu architektury dále na dvě komponenty – hlavní modul a aplikaci. Hlavní modul navrhneme a implementujeme jako knihovnu, a to z toho důvodu, aby bylo možné vytvořené algoritmy využívat i v ostatních projektech. Jeho úkolem bude provádět automatickou anotaci a za tímto účelem bude využívat vytvořené zásuvné moduly. Aplikace bude pak sloužit pouze k demostraci navrženého celku.

Zásuvné moduly budou sice implementovány odděleně od hlavního modulu, ale při běhu aplikace si budou mezi sebou předávat data zapouzdřená do stejných datových struktur. Z tohoto důvodu je nutné zajistit, aby definice použitých struktur byla mezi moduly sdílená. Vytvoříme proto SDK[7], který bude definice datových struktur obsahovat. Tento SDK pak bude zapotřebí jak při implementaci nových modulů, tak i při běhu celé aplikace. Navržená architektura bude tedy celkem tvořena aplikací, SDK, hlavním modulem a jednotlivými zásuvnými moduly. Jejich vzájemné spojení znázorňuje komponentový diagram UML na obrázku 4.1.

4.2.2 SDK

V této práci je úkolem SDK definovat datové typy, které jsou sdíleny hlavním modulem, zásuvnými moduly a případně aplikací. V našem případě půjde především o definice datového typu pro anotace, které jsou vytvářeny v anotačních modulech a předávány dále hlavnímu modulu k dalšímu zpracování, a definice datového typu reprezentující případné chyby, které mohou při výpočtech nastat a je nutné je v řetězci zpracování propagovat dále. Kromě sdílených datových struktur bude SDK obsahovat také několik nástrojů pro práci se snímky a pamětí. Protože SDK bude distribuováno za účelem implementace nových anotačních modulů, bude součástí SDK i definice rozhraní zásuvného modulu. Navrženou strukturu SDK zobrazuje obrázek 4.2.

[7]Zkratka SDK pochází z anglického spojení slov Software Developmnet Kit – sada nástrojů pro vývoj softwaru.



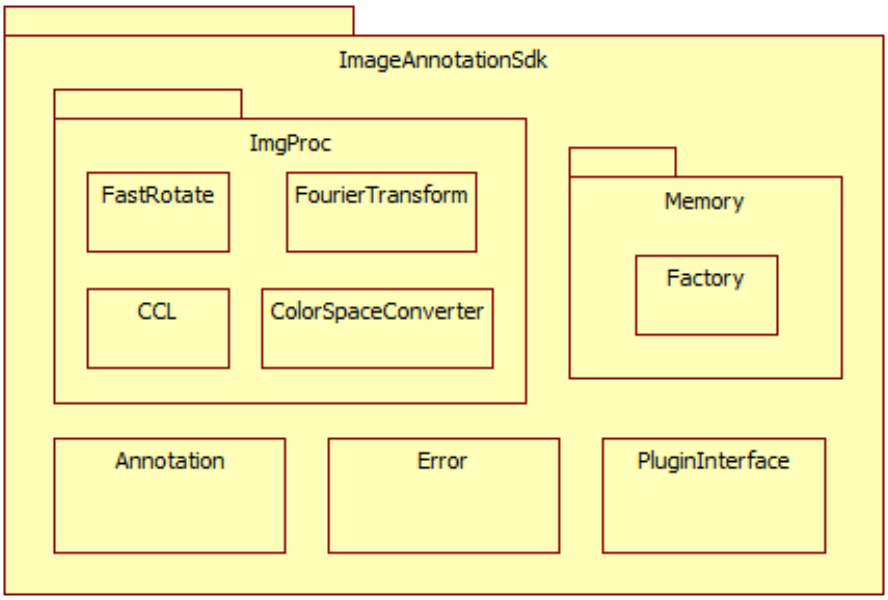
Obrázek 4.1: Architektura navrženého softwaru.

Datová struktura - anotace

Anotace určují nalezený objekt na scéně, a to jak jeho typem tak také polohou. Typ nalezeného objektu budeme kódovat nezáporným celým číslem, přičemž význam kódu bude dán anotačním modulem. Polohu objektu je nutné uložit tak, aby ho později bylo možné co nejsnadněji opět lokalizovat. V ideálním případě bychom použili obalující křivku, která odděluje objekt od jeho okolí. Takovou křivku by bylo možné snadno uložit např. pomocí Freemanova řetězového kódu, který jsme uvedli v 2.2.1. V našem případě si ale pro jednoduchost zvolíme obdélník rovnoběžný s okraji snímku. Jedná se o často používaný způsob identifikace objektů na snímku. Konvexní objekty lze tímto způsobem snadno ohraničit, problém může nastat u objektů konkávních. Ohraničení bude totiž vyznačovat také část snímku, která k objektu nepřísluší (příklad ohraničení objektů znázorňuje obrázek 4.3). K anotacím proto přidáme také informaci o tom, z jaké části zaplňuje objekt plochu své obdélníkové hranice. Anotace budou v SDK reprezentovány datovou strukturou `Annotation`, kterou zobrazuje obrázek 4.4.

Datová struktura - chyby

Chyby jsou v SDK, potažmo v celé aplikaci reprezentovány celým číslem. Pro specifikaci chyby jsou povoleny kladné i záporné hodnoty, přičemž používáme konvenci,



Obrázek 4.2: Navržená struktura SDK

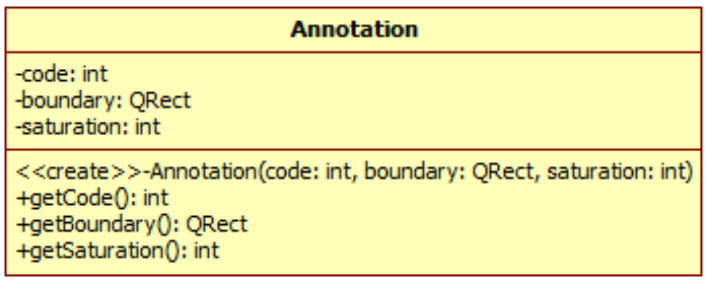
(a) Ohraničení konkávního objektu. (b) Ohraničení konvexního objektu.

Obrázek 4.3: Příklad ohraničení objektů obdélníkem.

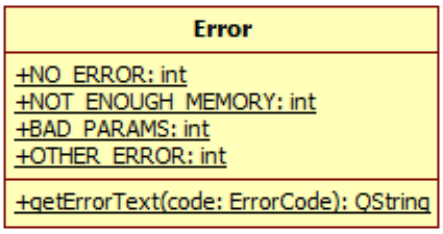
že kladné hodnoty znamenají chyby, záporné pouze varování a hodnota 0 signalizuje úspěšnou operaci. Pro práci s chybami je v SDK navržena třída `Error`. Ta definuje následující základní chyby:

- 0 (`NO_ERROR`) – při zpracování nedošlo k chybě,
- 1 (`NOT_ENOUGH_MEMORY`) – nelze alokovat paměť,
- 2 (`BAD_PARAMS`) – metoda odbržela špatné parametry,
- 3 (`OTHER_ERROR`) – jiná chyba.

K těmto chybám třída `Error` definuje také metodu pro převod chybového kódu na text. Definici datové struktury znázorňuje obrázek 4.5.



Obrázek 4.4: Datová struktura reprezentující anotace.



Obrázek 4.5: Datová struktura reprezentující chyby v hlavním a zásuvných modulech.

4.2.3 Rozhraní zásuvného modulu

Rozhraní modulu definuje, jakým způsobem lze pracovat se zásuvnými moduly. Tato definice je závazná a není možné ji později změnit, resp. po každé změně je nutné provést reimplementaci všech částí aplikace, což může být velmi nepříjemné. Proto je nutné rozhraní navrhnout pečlivě.

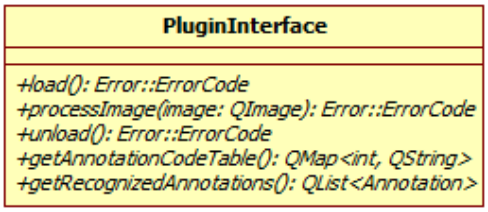
Hlavním úkolem modulu je provést analýzu snímku za účelem detekce vybraného objektu. Aby bylo možné lépe reagovat na chyby vzniklé během analýzy, rozdělíme tento proces na dvě metody:

```
processImage(Image): Error,
getRecognizedAnnotations(): List<Annotation>.
```

První metoda provede analýzu snímku, přičemž její návratová hodnota bude signalizovat, zda nedošlo během výpočtu k chybě. Druhá metoda bude sloužit k získání nalezených objektů/anotací. Modul by měl také samozřejmě poskytnout seznam všech typů objektů/anotací, které je schopen rozpoznat. K tomu bude sloužit metoda:

`getAnnotationsCodeTable(): Map<int, String>,`

pomocí které modul může zároveň definovat popis jednotlivých typů. Kromě uvedených metod je vhodné definovat také metody pro vytvoření a zrušení instance pluginu. Oddělením této činnost od konstruktoru a destruktoru třídy lze lépe reagovat na vzniklé chyby. Proto zavedeme metody `load(): Error`, `unload(): Error`. Navržené rozhraní je znázorněno na obrázku 4.6.



Obrázek 4.6: Definice rozhraní zásuvného modulu.

4.2.4 Hlavní modul

Tento modul představuje část architektury, která je dále rozšiřitelná pomocí zásuvných modulů. Jak jsme si již řekli, navrhneme tuto část jako samostatný modul, aby ho bylo možné používat i v jiných softwarech. Rozhraní modulu je podobné zásuvným modulům, neboť provádí stejnou činnost. Přidány jsou metody pro načtení a zrušení všech nalezených modulů a získání počtu dostupných modulů:

```
loadPlugins(): void,
unloadPlugins(): void,
getPluginsCount(): int.
```

Poslední přidaná funkce:

`getAnnotationImage(): Image`

slouží k vytvoření tzv. *vysvětlovacího snímku*. Ten vznikne anotováním objektů nalezených ve vstupním snímku.

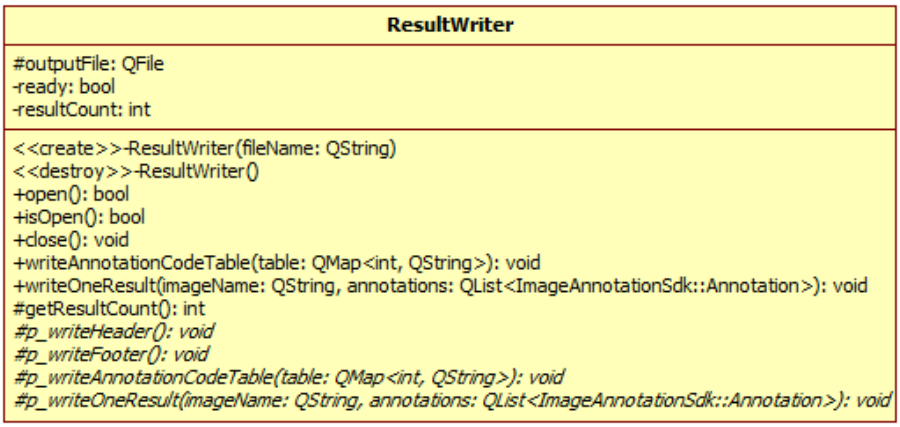
Kromě zmíněných funkcí bude dále hlavní modul definovat také nástroj pro uložení nalezených anotací. K tomuto účelu bude definována abstraktní třída `ResultWriter`. Třída je definována jako abstraktní proto, aby bylo možné ukládat výsledky více způsoby. Definice hlavního modulu je znázorněna na obrázku 4.7, definice třídy `ResultWriter` na obrázku 4.8.

Obrázek 4.7: Struktura hlavního modulu.

Uložení anotací

Protože nyní již máme představu o tom, jak bude popis scény vypadat a co bude jeho účelem, můžeme specifikovat způsob, jakým lze vytvořené anotace uložit. V teoretickém rozboru jsme uvedli, že máme na výběr mezi interním a externím uložením. Vzhledem k tomu, že interní uložení je závislé na typu souborového formátu snímku a musíme při něm dodržovat jistá pravidla, vybereme si externí způsob uložení. Díky tomu máme veškerou volnost při volbě formátu výstupního souboru. Aby bylo možné vygenerované výsledky snadno a rychle prohlížet, budeme data ukládat v textové podobě. K tomuto účelu si vybereme formát XML, který je pro ukládání strukturovaných dat velmi rozšířený a je také široce podporovaný v ostatních softwarech. Vzhledem k tomu, že při zpracování vzniká úplně nový soubor, je třeba zajistit, aby vstupní a výstupní soubor byly nějak přehledně spojeny. V této práci to budeme zajišťovat vhodným pojmenováním výstupního souboru, které bude vždy odvozeno od vstupního souboru digitálního snímku. Název souborů bude shodný a odlišovat se budou pouze příponou, pro anotační soubor budeme používat příponu XML.

Kromě abstraktní třídy `ResultWriter` bude modul poskytovat také konkrétní třídu `XmlResultWriter`, která bude sloužit k uložení do XML.



Obrázek 4.8: UML diagram třídy `ResultWriter`.

Návrh výstupního souboru

Nyní provedeme návrh souboru sloužícího k uložení nalezených anotací. Každá anotace bude v souboru reprezentována svým typem, ohraničujícím obdélníkem a koeficientem saturace. Obdélník bude reprezentován souřadnicemi levého horního rohu (v pořadí x, y) a svou šířkou a výskou. Vzhledem k tomu, že typ anotovaného objektu je identifikován číselným identifikátorem, je v souboru uložena také kódová tabulka všech dostupných anotací. Ta ke každému kódu obsahuje text definovaný v zásuvném modulu, který vysvětluje jeho význam. Abychom zajistili lepší spojení se vstupním snímkem, bude výstupní soubor obsahovat název a cestu ke vstupnímu souboru. Kromě toho uložíme také časovou značku udávající kdy byla anotace provedena. K uložení těchto dat použijeme již zmíněný jazyk XML. Formát celého souboru je závazný a je dán XSD šablonou. Pořadí jednotlivých XML elementů nelze měnit a je následující: časová značka, kódová tabulka, vstupní soubor a seznam nalezených anotací. Pro názornost uvedeme jednoduchý příklad výstupního souboru:

```
<ImageAnnotationTool
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="ImageAnnotationTool.xsd">
  <TimeStamp date="2013-01-15" time="12:32:02"/>
  <AnnotationCodeTable>
    <Annotation code="1" value="face"/>
    <Annotation code="2" value="sky"/>
    <Annotation code="3" value="forest"/>
  </AnnotationCodeTable>
  <Image file="file_1.jpg" path="D:/files"/>
  <Result>
```

```
    <Annotation code="1" position="[60, 36, 59, 11]" saturation="73"/>
    <Annotation code="3" position="[480, 0, 11, 5]" saturation="62"/>
  </Result>
</ImageAnnotationTool>
```

4.2.5 Aplikace

Aplikace bude sloužit především k demonstrování funkčnosti navrženého softwaru. Jejím hlavním úkolem bude provádět I/O operace, tj. načtení vstupního snímku (vstupních snímků) a uložení výstupních souborů.

Uživatelské rozhraní

Aplikace nebude obsahovat řádkové ani grafické uživatelské rozhraní, neboť jejím hlavním úkolem je vygenerovat anotační soubor a při této činnosti není zapotřebí přílišná interakce s uživatelem. Veškeré nastavení bude možné specifikovat během spuštění aplikace, přičemž jsou podporovány následující parametry. Hlavní a povinný parametr `-i` bude označovat vstupní soubor digitálního snímku, resp. vstupní adresář, ve kterém se může nacházet libovolné množství vstupních souborů. Další parametr `-e` slouží k vygenerování vysvětlovacího digitálního snímku, který bude znázorňovat, kde byly objekty lokalizovány. Poslední parametr `-v` povolí výpis jednotlivých kroků aplikace.

Poznamenejme, že jméno výstupního souboru není možné v aplikaci změnit. Bude vždy určeno názvem vstupního snímku a lišit se bude pouze příponou. Program lze tedy spustit následujícím způsobem:

```
image_annotation_app vstup [nastavení]

vstup:
-i (input) cesta vstupního snímku či adresáře

nastavení:
-v (verbose) povolí výpis informačních zpráv
-e (explain) bude generovat vysvětlovací snímek
```

Formáty vstupních souborů

Podporované formáty vstupních souborů jsou dané knihovnou Qt pomocí které bude program implementován. V současné době jde o formáty: BMP, GIF, JPG, JPEG, PNG, PBM, PGM, PPM, TIFF, XBM, XPM.

Formát výstupních souborů

Aplikace bude produkovat dva typy výstupních souborů: soubor vysvětlovacího snímku a soubor s anotacemi. K uložení vysvětlovacího snímku jsme zvolili formát JPG, protože je v oblasti digitálních snímků nejrozšířenější. Poznamenejme, že tento formát bude použit vždy a nezávisle na formátu vstupního snímku. Pro ukládání nalezených anotací jsme zvolili jazyk XML, přičemž návrh formátu výstupního souboru jsme již provedli výše. K vygenerování XML souboru bude aplikace využívat nástroje hlavního modulu.

4.3 Implementace

4.3.1 Architektura

Hlavní modul a SDK jsou implementovány jako sdílené knihovny, konkrétně se jedná o soubory `image_annotation_tool` a `image_annotation_sdk`. Aplikace je implementována jako standardní C++ aplikace, která tyto sdílené knihovny využívá, konkrétně se jedná o soubor `image_annotation_app`. Přípony souborů neuvádíme, protože jsou závislé na použité platformě.

4.3.2 Hlavní modul

Hlavní modul obsahuje nástroje pro rychlou rotaci snímku a nástroje pro výpočet diskrétní Fourierovy transformace. Rotace je realizována pomocí tří operací zkosení. Dvourozměrná diskrétní Fourierova transformace je realizována pomocí knihovny FFTW.

4.3.3 Zásuvné moduly

Zásuvné moduly jsou implementovány jako Qt Pluginy. Každý zásuvný modul je v Qt implementován právě jednou sdílenou knihovnou. Každý modul (knihovna) je reprezentován jedním souborem, přičemž typ souboru je závislý na platformě. Podporované typy knihoven je možné najít v dokumentaci, programově lze typ knihovny ověřit pomocí metody `QLibrary::isLibrary(...)`. Pro práci s Qt pluginy je v Qt k dispozici třída `QPluginLoader`. Ta poskytuje především nástroje k jejich alokaci a dealokaci. Jedna instance třídy `QPluginLoader` pak pracuje právě s jedním pluginem. Předpokládejme dále operační systém `Linux` a knihovnu `library.so`. Tento plugin lze dynamicky načíst následovně:

```
QPluginLoader *loader = new QPluginLoader("library.so");
if(loader->load()) {
  QObject *plugin = loader->instance();
}
```

přičemž metodu `load()` není nutné volat explicitně, pokud ji nezavoláme postará se o to metoda `instance()`. Protože knihovna může implementovat libovolný Qt Plugin, musíme dále provést jeho přetypování na námi požadovaný datový typ:

```
if(plugin) {
  ImageAnnotationSdk::PluginInterface *my_plugin =
      qobject_cast<ImageAnnotationSdk::PluginInterface *>(plugin);
  if(my_plugin) {
    // everything is ok, now we can use my_plugin
  }
}
```

Funkce `T qobject_cast(QObject *object)` slouží k převedení instance `object` na typ `T`. Třída `T` musí být oděděna od třídy `QObject` a musí deklarovat makro `Q_OBJECT` (vizte podkapitolu 5.1). Toto přetypování je možné pouze v případě, že object je typu `T`, v opačném případě metoda vrací `0`.

Dealokaci pluginu a zrušení instance `QPluginLoaderu` pak provedeme příkazem:

```
loader->unload();
delete loader;
```

4.3.4 Načtení snímků

Protože vstupem aplikace může být soubor i adresář, musíme nejprve tyto dva případy od sebe odlišit. To lze provést např. příkazem `QFileInfo::isDir()`. Pokud se jedná o adresář, musíme provést načtení všech souborů v adresáři. Jejich seznam lze získat metodou `QDir::entryList(...)`. Celé načtení může vypadat takto:

```
QList<QString> inputFiles;
if(QFileInfo(inputPath).isFile()) {
  inputFiles << inputPath;
} else if(QFileInfo(inputPath).isDir()) {
  QDir selectedDir(inputPath);
  inputFiles << selectedDir.entryList(QDir::Files);
}
```

Tím získáme seznam jmen všech zadaných souborů (buď jeden vstupní soubor, nebo všechny soubory v zadaném adresáři). Nyní musíme provést samotné načtení snímků, k tomu nám poslouží třída `QImageReader`. Nejprve je však vhodné zkontrolovat, zda daný soubor opravdu představuje digitální snímek. To lze provést příkazem:

```
if(!QImageReader::supportedImageFormats().contains(
                      QFileInfo(my_file).suffix())) {
  // everything is ok, file is image
}
```

Načtení jednoho snímku pak lze provést pomocí:

```
QImageReader imageReader;
...
imageReader.setFileName(my_file);
QImage image = imageReader.read();
if(!image.isNull()) {
  // image is ok
}
```

Protože snímek může být v Qt uložen v mnoha formátech, převedeme ho do společného formátu, abychom se všemi snímky mohli pracovat stejným způsobem. Vybereme si formát `ARGB32`, kde je každý barevný kanál reprezentován 8 bity. Převod pak provedeme následovně:

```
image = image.convertToFormat(QImage::Format_ARGB32);
```

Anotaci tohoto snímku pak provedeme následujícím způsobem:

```
ImageAnnotationTool tool;
...
tool.processImage(&image);
```

4.3.5 Zápis výstupních souborů

Uložení XML souboru

K ukládání dat do souboru formátu XML lze v Qt využít třídu `QXmlStreamWriter`. Způsob zápisu XML dokumentu si uvedeme na krátkém příkladu. Uvažujme následující dokument:

```
<RootElement>
  <Element1>
    <Element2 att1="val1" att2="val2"/>
  </Element1>
</RootElement>
```

Ten lze vygenerovat pomocí třídy `QXmlStreamWriter` takto:

```
QXmlStreamWriter xml;
xml.writeStartDocument();
xml.writeStartElement("RootElement");
xml.writeStartElement("Element1");
xml.writeStartElement("Element2");
xml.writeAttribute("att1", val1);
xml.writeAttribute("att2", val2);
xml.writeEndElement();
xml.writeEndElement();
xml.writeEndElement();
xml.writeEndDocument();
```

Formát námi zvoleného výstupního souboru je dán XSD šablonou `ImageAnnotationTool.xsd`. Protože definice této šablony je poměrně rozsáhlá, nebudeme ji zde uvádět. Je však dostupná na přiloženém DVD.

Uložení snímku

Vysvětlovací snímek, který aplikace generuje, je ukládán ve formátu JPG. K uložení snímku použijeme metodu `save()`, která je dostupná přímo ve třídě `QImage`:

```
QImage image = ...;
...
image.save(my_name);
```

5 Moduly

Během své práce jsem se zaměřil na vývoj modulů pro detekci lidského obličeje, oblohy, zeleně a detekci uměleckých snímků. V této kapitole se na vytvořené moduly postupně zaměříme, přičemž u každého z nich popíšeme základní teorii, na které je anotační algoritmus založen, nejzásadnější části implementace a také výsledky anotace. Jako metriku k vyhodnocení výsledků použijeme relativní počet správných detekcí, relativní počet chybných detekcí a relativní počet nenalezených objektů. Za správnou detekci považujeme stav, kdy algoritmus anotuje pouze hledaný objekt. Chybná detekce je taková, kdy algoritmus anotuje hledaný objekt spolu s pozadím, nebo část snímku, kde se hledaný objekt nevyskytuje. Objekt považujeme za nenalezený, pokud se na snímku nachází, ale algrotimus jej neanotoval.

Před samotným popisem vyvinutých algoritmů se nyní ještě podívejme na to, jak navrhnout a implementovat zcela nový modul.

5.1 Vytvoření nového modulu

Každý modul je v aplikaci reprezentován třídou implementující rozhraní `ImageAnnotationSdk::PluginInterface`. Definice tohoto rozhraní a ostatních datových struktur potřebných k implementaci jsou k dispozici v přiloženém SDK, které je distribuováno spolu s aplikací jako knihovna `image_annotation_sdk`. Třídu modulu je nutné implementovat – stejně jako samotnou aplikaci – v prostředí Qt, aby byla zaručena jejich kompatibilita. Z tohoto důvodu musí být splněny následující podmínky:

- základní třída musí být oděděna od třídy `QObject`,
- definice této třídy musí obsahovat makra,

  `Q_OBJECT,`
  `Q_INTERFACES(ImageAnnotationSdk::PluginInterface).`

- na konci zdrojového souboru musí být proveden export nového modulu.

  `Q_EXPORT_PLUGIN2(jmeno_projektu, jmeno_tridy)`

Definice základní třídy nového modulu by tedy mohla vypadat např. takto:

```
-- plugin.h --
...
class Plugin: public QObject,
              public ImageAnnotationSdk::PluginInterface {
  Q_OBJECT
  Q_INTERFACES(ImageAnnotationSdk::PluginInterface)

  public:
    ImageAnnotationSdk::Error::ErrorCode load();
    ImageAnnotationSdk::Error::ErrorCode processImage(QImage *image);
    ImageAnnotationSdk::Error::ErrorCode unload();

    QMap<int, QString> getAnnotationCodeTable();
    QList<ImageAnnotationSdk::Annotation> getRecognizedAnnotations();
};
...

-- plugin.cpp --
...
Q_EXPORT_PLUGIN2(jmeno_projektu, Plugin)
```

5.2 Modul pro detekci lidského obličeje

Obličej představuje jeden z nejčastěji anotovaných objektů v počítačovém vidění. Algoritmy pro detekci obličeje jsou již běžně dostupné ve fotoaparátech či softwarech obsluhujících kamery všeho druhu. Požadavek na detekci obličeje je kladen také v oblasti verifikačních a identifikačních sotfwarů. Používané algoritmy jsou však většinou založeny na strojovém učení, přičemž se převážně zpracovává pouze jasová informace obrazu. V našem případě se pokusíme o pravděpodobnostní přístup využívající především barevnou informaci.

5.2.1 Návrh algoritmu

Charakteristickými vlastnostmi obličeje jsou bezesporu jeho tvar a barva kůže. Protože cílový tvar se ve snímku hledá velmi obtížně, budeme obličej hledat pouze podle barvy a informaci o tvaru použijeme až v rámci klasifikace.

Navržený algoritmus se skládá z pěti kroků: předzpracování, výpočet pravděpodobnostního ohodnocení, segmentace, klasifikace a anotace.

Předzpracování

Hledání oblastí snímku, které mohou příslušet lidské kůži, budeme provádět na základě barevné informace. Abychom zvýšili úspěšnost detekce těchto části, provedeme předzpracování, jehož úkolem bude zajistit lepší vyvážení barev ve snímku. K tomu využijeme postup uvedený v podkapitole 2.2.3. Snímek dále rozmažeme pomocí konvolučního filtru abychom zmenšili rozptyly barev a potlačili změny v jasu.

Výpočet pravděpodobnostního ohodnocení

Při použití barvy k detekci obličeje se musíme vypořádat s mnoha problémy. Barva lidské kůže na snímku je ovlivněna mnoha faktory (osvětlení, pozice obličeje vzhledem k záznamovému zařízení apod.), různé záznamové zařízení navíc produkují odlišnou barvu pro stejný obličej za stejných podmínek. Barva kůže se liší podle rasy člověka a je navíc odlišná i mezi jednotlivci v rámci jedné rasy. Podle mnoha prací se však tyto odlišnosti projevují především v jasu barvy a ne v její chrominanci. Z tohoto důvodu budeme snímky zpracovávat pouze podle chrominance barev, přičemž použijeme takový barevný prostor, který luminanci a chrominaci barvy odděluje.

K dispozici máme mnoho takových barevných prostorů, ovšem dle práce [26] je pro detekci obličeje nejlepší použít barevný prostor typu H-S (Hue-Saturation), který důsledně odděluje barvu a jas. Jako pravděpodobnostní model byl použit relativní histogram četnosti (vizte podkapitolu 2.3.1) chrominance barev a jako metriku pro tuto analýzu autoři zvolili korektnost a chybu detekce v závislosti na prahovací hodnotě.

V naší práci však provedeme vlastní analýzu. Sestavili jsme proto 160 snímků (celkem 80 milionu bodů), které zobrazují pouze obličej. Vybereme si takový barevný prostor, který na této množině vykazuje nejmenší závislost luminance a chrominance, resp. jejich nejlepší oddělení. Otestujeme čtyři vybrané barevné prostory – YUV, HSL, HSV, Lab a jako metriku použijeme Pearsonův korelační koeficient:

$$\varrho_{X,Y} = \frac{\mathrm{E}(XY) - \mathrm{E}(X)\,\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2) - \mathrm{E}^2(X)}\sqrt{\mathrm{E}(Y^2) - \mathrm{E}^2(Y)}}$$

pro náhodně veličiny X, Y. Provedením experimentu získáme hodnoty dle tabulky 5.1. Na základě této analýzy vybereme prostor Lab, u kterého dosahuje korelační koeficient luminance a chrominanci minimální hodnoty.

HSL	$\varrho_{H,L}$	$\varrho_{S,L}$	$\lvert\max\{\varrho_{H,L}, \varrho_{S,L}\}\rvert$
	-0,0692284	0,267699	0,267699
HSV	$\varrho_{H,V}$	$\varrho_{S,V}$	$\lvert\max\{\varrho_{H,V}, \varrho_{S,V}\}\rvert$
	-0,132946	-0,317598	0,317598
YUV	$\varrho_{Y,U}$	$\varrho_{Y,V}$	$\lvert\max\{\varrho_{Y,U}, \varrho_{Y,V}\}\rvert$
	-0,336402	0,250726	0,336402
Lab	$\varrho_{L,a}$	$\varrho_{L,b}$	$\lvert\max\{\varrho_{L,a}, \varrho_{L,b}\}\rvert$
	0,0723135	0,265891	0,265891

Tabulka 5.1: Výsledky analýzy vybraných barevných prostorů.

Nyní již přistoupíme k samotného ohodnocení. Každému bodu přiřadíme reálné číslo z intervalu $\langle 0, 1\rangle$, které bude odpovídat pravděpodobnosti, s jakou daný bod představuje obličej. K tomu použijeme spojitý pravděpodobností model, který sestavíme dle podkapitoly 2.3.3. K jeho vytvoření použijeme pouze informaci o chromacitě, informaci o luminanci zanedbáme. Experimentální cestou se ovšem ukazuje, že ohodnocení v prostoru YUV dává také velmi dobré výsledky, proto ho k ohodnocení použijeme také. Protože jsme si vybrali dva barevné prostory, sestavíme pro každý barevný prostor jeden model a výsledný model definujeme jako jejich aritmetický průměr:

$$\mathrm{B}(c) = \frac{\mathrm{B}_{\mathrm{UV}}(c_U, c_V) + \mathrm{B}_{\mathrm{ab}}(c_a, c_b)}{2},$$

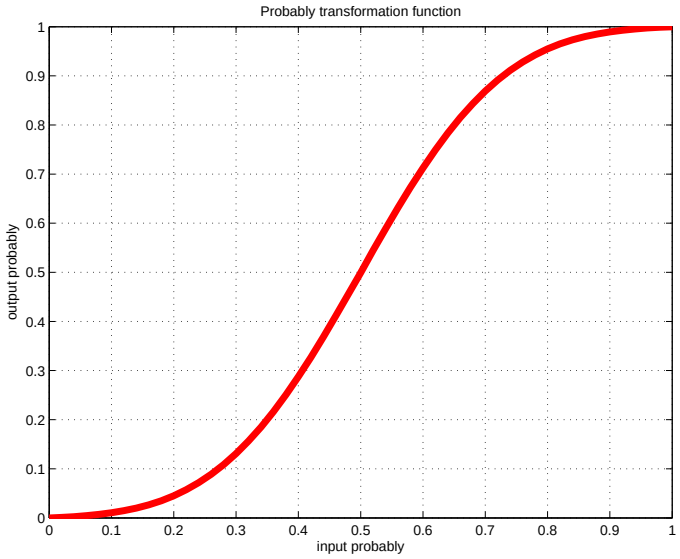
přičemž c označuje barvu a c_U, c_V, resp. c_a, c_b barevné kanály této barvy v prostoru YUV, resp. Lab. Ukazuje se, že toto ohodnocení však vykazuje následující problémy.

Stává se, že některým barvám z pozadí přiřadíme malou pravděpodobnost, přestože ve skutečnosti kůži nepřísluší. Předpokládáme, že je to způsobeno ignorováním jasu během ohodnocování barev. Dále se ukazuje, že v případěch, kdy dochází ke změně osvětlení v rámci jednoho obličeje, vykazuje ohodnocení velký rozptyl hodnot. To je pravděpodobně dáno tím, že luminance a chrominance barev není v daném barevném prostoru úplně nezávislá a změny jasu se tak nepatrně projevují i v chrominanci.

Abychom tyto nežádoucí jevy potlačili, provedeme dodatečnou transformaci pravděpodobnostního ohodnocení a to pomocí nelineární funkce. Tu definujeme tak, aby potlačovala malé hodnoty a posilovala vysoké hodnoty pravděpodobnosti. Potlačením malých hodnot nepřímo stanovíme oblast (vrstevnici), ve které se mohou souřadnice UV, resp. ab nacházet, a barvám mimo tento interval bude přidělena nulová pravděpodobnost. Posílení vyšších hodnot bude mít za následek zmenšení rozptylu ohodnocení v rámci oblasti lidké kůže. V našem případě se nám experimentálně nejvíce osvědčila funkce:

$$\begin{aligned} \mathrm{F}(p) &= \begin{cases} \mathrm{t}(p), & p \leq \frac{1}{2}, \\ 1 - \mathrm{t}(1-p), & p > \frac{1}{2}, \end{cases} \\ \mathrm{t}(p) &= -22{,}30p^5 + 19{,}04p^4 - 1{,}33p^3 + 0{,}61p^2 + 0{,}04p, \end{aligned}$$

jejíž průběh ukazuje obrázek 5.1.



Obrázek 5.1: Průběh zvolené transformační funkce.

Segmentace oblastí lidské kůže

Ve vytvořeném pravděpodobnostním ohodnocení je dále zapotřebí najít oblasti, které mohou reprezentovat lidskou kůži. Předpokládejme, že analyzovaný snímek

obsahuje alespoň jednu takovou oblast (např. obličej), potom se na snímku vyskytují právě dva typy objektů (dvě třídy) ve smyslu kůže/pozadí. Úkolem segmentace je tyto dva typy objektů od sebe oddělit. V našem případě k tomu použijeme populární metodu Otsu, jež jsme odvodili v podkapitole 2.2.4 a která se snaží o maximální separabilitu obou tříd.

Pokud však na snímku nebude žádná oblast příslušet kůži, metoda Otsu se pokusí na dvě třídy segmentovat pouze pozadí a výsledkem bude práh s velmi nízkou hodnotou. Toho lze ovšem dobře využít, neboť jednoduchým porovnáním obdrženého prahu s parametrem t_{otsu} lze rychle rozhodnout, zda jsou na snímku oblasti, jejichž barva odpovídá barvě lidské kůže.

Dále předpokládejme, že jsme nalezli jednu oblast O, která odpovídá vysokému pravděpodobnostnímu ohodnocení. V ideálním případě by byl celý tento segment dostatečně oddělen od pozadí. Může se však stát, že spolu s oblastmi zachycující kůži budou segmentovány také části pozadí. Také může dojít k nechtěnému spojení více oblastí příslušejících kůži do jednoho segmentu. Abychom eliminovali tento nežádoucí jev, zapojíme do řetězce zpracování další segmentaci. Ta je založena na jednoduchém prahování složky Z v barevném prostoru XYZ. Práh je stanoven podle následujícího vztahu:

$$t = pZ_{max} + (1 - p)Z_{min}, \tag{5.1}$$

kde $p \in \langle 0, 1 \rangle$ je parametr segmentace a Z_{max}, resp. Z_{min} odpovídá maximální, resp. minimální hodnotě Z v rámci komponenty O:

$$Z_{max} = \max_{\forall [X,Y,Z] \in O} \mathrm{Z}, \quad Z_{min} = \min_{\forall [X,Y,Z] \in O} \mathrm{Z}\,.$$

Prostor XYZ a vztah (5.1) je volen experimentálně a vychází pouze z pozorování vlastností testovacích snímků v prostoru XYZ. Výsledky ukazují, že tato segmentace je úspěšná v případě, kdy podíl pozadí v oblastí není příliš velký.

Protože se ukazuje, že dodatečná segmentace není zapotřebí vždy, resp. v některých případech její použití výsledky zhoršuje, je třeba rozhodnout, kdy ji máme použít. K tomuto rozhodnutí dospějeme na základě výsledku prvních tří kaskád použitého kaskádního klasifikátoru (vizte dále). Vyhodnotí-li zvolená část klasifikátoru analyzovanou komponentu jako pozitivní (tj. může být obličejem), pak dodatečná segmentace není zapotřebí a přistoupí se k finální klasifikaci. V případě, že komponentu klasifikátor zamítne, pokusíme se ji za účelem nalezení potenciálního obličeje dále segmentovat.

Klasifikace a anotace obličeje

V předchozích krocích jsme nalezli segmenty s barvou lidské kůže, které mohou odpovídat obličeji. Nyní je zapotřebí provést jejich klasifikaci a určit, které z nich opravdu lidský obličej zobrazují. Navržený klasifikátor se skládá ze dvou části: jednoduchého kaskádního klasifikátoru a detektoru Viola-Jones.

Kaskádní klasifikátor používá ke svému rozhodnutí pouze informace o tvaru segmentu a nikoli samotný obsah komponenty (vizuální data). Tento klasifikátor má za úkol přijmout pouze ty komponenty, které svým tvarem mohou obličeji odpovídat, a tím zmenšit počet komponent předložených detektoru Viola-Jones. Abychom lépe zachytili tvar segmentu aproximujeme jej elipsou, k čemuž lze efektivně využít algoritmus uvedený v podkapitole 2.2.5. Použitý klasifikátor se skládá ze čtyř kaskád, které vyhodnocují čtyři zvolené příznaky: p_1 – absolutní velikost segmentu (vztah (2.9)), p_2 – chybu aproximace (vztah (2.2.5)), p_3 – podlouhlost aproximační elipsy a p_4 – kruhovitost původního segmentu (vztah (2.11)). Příznaky jsou vyhodnocovány v uvedeném pořadí a jsou porovnávány s hodnotami t_1, t_2, t_3 a t_4. V případě, že kaskádní klasifikátor označí segment pozitivně, tj. všechny vybrané příznaky jsou vyhovující, předá se segment ke klasifikaci Viola-Jones detektoru.

Princip detektoru Viola-Jones jsme si stručně nastínili v podkapitole 2.2.6. Jak bylo uvedeno, jeho hlavní nevýhodou je potřeba testovat všechny pozice a velikosti podoken ve vstupním snímku. V našem případě však tyto parametry známe a stačí nám pouze potvrdit, zda vybraná část snímku obličej obsahuje, či nikoliv. Kromě toho můžeme efektivně využít aproximaci segmentu elipsou, protože víme, že tuto aproximaci je možné provést s malou chybou. Můžeme tedy usuzovat, že úhel natočení elipsy φ odpovídá úhlu natočení segmentu a tedy i potenciálního obličeje. Tento úhel pak lze využít ke zpětnému otočení obličeje tak, aby se vždy nacházel v klasické svislé poloze. Díky této jednoduché úpravě lze detekovat také obličeje, které v původní podobě detektor Viola-Jones nezachytí. Obrázek 5.2 ukazuje příklad zpětné rotace obličeje v průběhu detekce. Poznamenejme, že úhel natočení elipsy je vždy v rozmezí $\pm 90°$ vůči vertikální ose a tudíž inverzní rotace je účinná pouze v případě, kdy je obličej otočen o tento úhel. Větší úhel je totiž shodný s doplňkovým úhlem $\varphi - 180°$ (pro $\varphi \geq 0°$), resp. $\varphi + 180°$ (pro $\varphi < 0°$) a rotovaný obličej bude zrcadlen vůči horizontální ose.

Na závěr algoritmu anotujeme takové části snímku, které byly detektorem Viola-Jones klasifikovány jako obličej. K tomu opět využijeme ohraničení pomocí obdélníku, který je v tomto případě velmi názorný.

Obrázek 5.2: Ukázka inverzní rotace obličeje. Snímek je převzat ze zdroje I dle tabulky B.1.

5.2.2 Implementace

Navržený algoritmus je implementován zásuvným modulem `plugin_face_detect`, konkrétně třídou `PluginFaceDetect`. Metoda `load()` provádí načtení obou pravděpodobnostních modelů barev a také alokaci OpenCV datových struktur nezbytných pro klasifikátor Viola-Jones. Jejich dealokaci naopak provádí metoda `unload()`. Modely jsou uloženy v binárních souborech `face_model_yuv.mtx` a `face_model_lab.mtx`, přičemž k jejich načtení slouží metoda `loadModel(...)`.

Předzpracování, ohodnocení, segmentace

Vyvážení barev je prováděno dle postupu, který jsme uvedli v podkapitole 2.2.3. Pro parametry předzpracování s_1 a s_2 jsme experimentálně zvolili hodnoty $s_1 = s_2 = 1{,}2$ %. Konvoluční filtr použitý pro rozmazání snímku jsme zvolili následující:

$$F = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

přičemž rozmazání je prováděno v prostoru RGB a pro každý barevný kanál zvlášť. Parametry segmentace jsme volili experimentálně: $t_{otsu} = 0{,}5$, $p = 8$ %.

Klasifikace

Jednoduchý klasifikátor jsme experimentálně nastavili tak, aby klasifikoval segmenty pro které platí: absolutní velikost je větší než 0,5 % plochy snímku, chyba aproximace je menší než 20 %, podlouhlost je menší než 2,5 a kruhovitost větší než 0,5. Omezení na kruhovitost a podlouhlost vyplývají z pozorování tvaru průměrného obličeje.

Detektor Viola-Jones je v naší práci realizován pomocí knihovny OpenCV. Knihovna OpenCV používá pro uložení obrazových dat jiné datové struktury než knihovna Qt a proto je nutné provést jejich transformaci. Konkrétně budeme převádět data z třídy `QImage` do struktury `IplImage`.

V první jmenované třídě je snímek uložen po řádcích, přičemž k uložení budeme používat pouze formát `ARGB32`. Barva každého bodu je tak dána RGB hodnotami, případně alfa kanálem,. Každý kanál je reprezentován nezáporným celým číslem, které je kódováno 8 bity, celkem je tedy potřeba 32 bitů pro jeden bod. Tento formát je dodržen i v případě, kdy se alfa kanál ve snímku nevyskytuje. K samotným datům je možné přistupovat přímo pomocí metody `bits()`, která vrací ukazatel na jednorozměrné pole typu `unsigned char`, přičemž jednotlivé kanály jsou v tomto poli uloženy v pořadí BGRA.

Třída `IplImage` nabízí široké možnosti, jak obrazová data uložit, přičemž jedna z nich odpovídá právě popsanému způsobu uložení ve třídě `QImage`. Díky tomu můžeme celou konverzi implementovat následujícím způsobem. Nejprve nastavíme potřebné parametry struktury `IplImage`:

```
IplImage *imgHeader = cvCreateImageHeader(
                        cvSize(qimg->width(), qimg->height()),
                        IPL_DEPTH_8U,
                        4);
```

kde parametr `IPL_DEPTH_8U` říká, že každý kanál je kódován nezáporným celým číslem (odpovídá použitému typu `unsigned char`) a hodnota `4` udává počet kanálů. Nyní již stačí zkopírovat samotná obrazová data. Protože metoda `bits()` vrací pouze ukazatel na data (mělkou kopii), vytvoříme jejich kopii, aby obě struktury mezi sebou neměly sdílené položky a nedocházelo k problémům při jejich mazání:

```
uchar *newdata = (uchar *) malloc(sizeof(uchar) * qimg->byteCount());
memcpy(newdata, qimg->bits(), qimg->byteCount());
imgHeader->imageData = (char *) newdata;
```

5.2.3 Výsledky

Navržený algoritmus byl testován na fotografiích z internetové databáze [5]. Vzhledem k tomu, že tato databáze je velmi rozsáhlá (obsahuje více než 13 000 fotografií) vybrali jsme z nich náhodně 400 snímků. Výsledky testování shrnuje tabulka 5.2. Algoritmus jsme dále testovali na 110 běžných fotografiích z reálného prostředí, získané výsledky zobrazuje tabulka 5.3. Z uvedených výsledků vyplývá, že úspěšnost

detekce navrženého algoritmu je celkem zhruba 60 %. Protože výsledná úspěšnost je závislá na kvalitě použité klasifikátoru, provedli jsme na vybraných snímcích také test detektoru Viola-Jones. Obě tabulky obsahují pro porovnání i výsledky získané tímto testováním.

	navržený algoritmus		detektor Viola-Jones	
	absolutní počet	relativní počet	absolutní počet	relativní počet
správně detekováno	289	65,98 %	410	93,60 %
chybně detekováno	1	0,25 %	24	6,00 %
nenalezeno	149	34,02 %	28	6,40 %

Tabulka 5.2: Výsledky anotace lidského obličeje na snímcích z databáze [5].

	navržený algoritmus		detektor Viola-Jones	
	absolutní počet	relativní počet	absolutní počet	relativní počet
správně detekováno	81	58,27 %	113	81,29 %
chybně detekováno	4	3,64 %	20	18,18 %
nenalezeno	58	41,73 %	26	18,71 %

Tabulka 5.3: Výsledky anotace lidského obličeje na reálných fotografiích.

Příklady několika úspěšných detekcí jsou k vidění na obrázku 5.3, obrázek 5.4 naopak zobrazuje snímky, na kterých se obličej nepodařilo najít. Jak můžeme vidět, hlavní rozdíl mezi úspěšně a neúspěšně detekovanými obličeji tkví ve výsledcích segmentace. Analýzou se ukazuje, že segmentační fáze je zodpovědná za většinu neúspěšných detekcí. Tento jev je pravděpodobně způsoben tím, že histogram pravděpodobnostního ohodnocení není v některých případech možné dobře aproximovat směsí normálních rozdělení. Experimenty dále ukazují, že výsledky pravděpodobnostního ohodnocení jsou velmi závislé na vstupním snímku, resp. na způsobu vyvážení barev. Výsledná detekce se liší v závislosti na použitých parametrech, přičemž se ukazuje, že pro dobře vyvážené snímky je použití předzpracování kontraproduktivní. Za účelem zvýšení úspěšnosti detekce by bylo zapotřebí použít kvalitnější předzpracování snímku a lepší segmentační fázi.

Na závěr poznamenejme, že v porovnání s detektorem Viola-Jones jsme dosáhli méně chybných detekcí, což je dáno použitím barevné informace. Navržený algoritmus je navíc oproti detektoru Viola-Jones schopen detekovat obličeje, které jsou natočené o úhel $\pm 90°$. Ve výsledku se však ukazuje, že celková úspěšnost detekce je v porovnání s detektorem nižší.

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

Obrázek 5.3: Příklady detekce obličejů ve snímku. Sloupec (a) zobrazuje originální snímek, (b) výsledky ohodnocení, (c) výsledky segmentace a (d) zobrazuje vytvořené anotace. Snímky jsou získány (bráno od shora) ze zdrojů II, I, II, I, II, I dle tabulky B.1.

(a) (b) (c)

Obrázek 5.4: Příklady snímku, kde se nepodařilo obličej anotovat. Význam sloupců je stejný jako u předchozího obrázku. Všechny snímky pochází ze zdroje II dle tabulky B.1.

5.3 Modul pro detekci zelené vegetace

Další modul se soustřed'uje na hledání částí snímku, které odpovídají vegetaci. Ta je velmi často k vidění na venkovních fotografiích, nehledě na tom, zda jsou pořízeny v přírodě nebo městských oblastech. Speciálním případem takových fotografií mohou být také snímky pořízené během sportovních utkání apod.

5.3.1 Návrh algoritmu

Abychom získali informace o tom, jak taková vegetace na digitálním snímku nejčastěji vypadá, sestavili jsme testovací množinu čítající zhruba 60 snímků (obsahující celkem 180 milionů bodů), které zobrazují pouze vegetaci, a provedli jsme vizuální analýzu. Na základě tohoto pozorování můžeme říci, že nejdůležitějšími vlastnostmi vegetace jsou barva a textura. Jak se však ukazuje, vegetace nabývá velmi rozsáhlého množství barev, od zelené přes hnědou až po žlutou a červenou, což velmi ztěžuje její nalezení. Z tohoto důvodu se v naší práci uchýlíme ke zjednodušení – zaměříme se pouze na zelenou vegetaci.

Navržený algoritmus se skládá ze čtyř kroků: výpočet pravděpodobnostního ohodnocení, zpracování pravděpodobnostního ohodnocení, segmentace oblastí vegetace a anotace.

Výpočet pravděpodobnostního ohodnocení

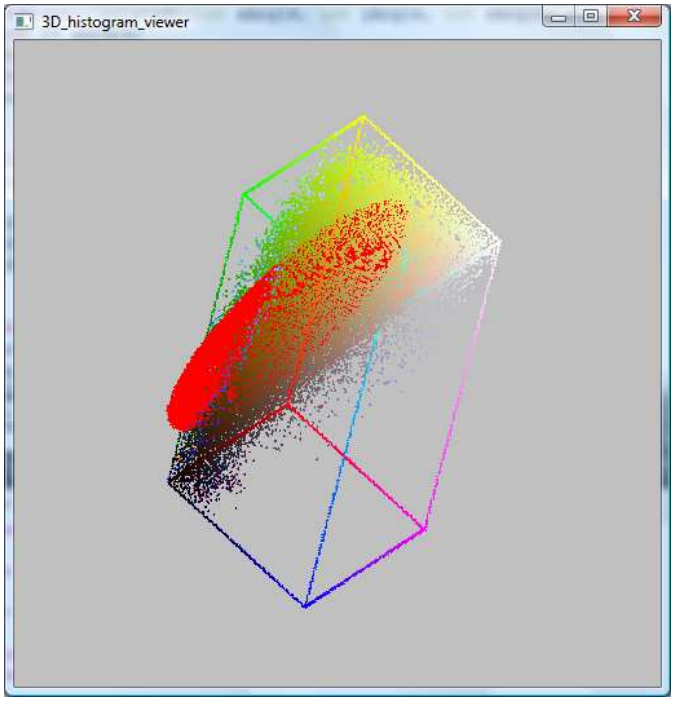
Každý bod ohodnotíme pravděpodobností, která bude vyjadřovat míru příslušnosti daného bodu zelené vegetaci. Na základě uvedeného pozorování stanovíme pravděpodobnost každého bodu dle barvy a textury.

Podívejme se nejprve na problém ohodnocení bodů z hlediska barvy. K tomu využijeme pravděpodobnostní model, který každé barvě přiřazuje pravděpodobnost, že odpovídá vegetaci. Vzhledem k tomu, že travnaté plochy zachycené na snímcích velmi často obsahují různé stíny, rozhodli jsme se, že pravděpodobnostní model sestavíme pouze na základě chrominance barvy. Vypuštěním jasové informace zmírníme závislost na množství zachyceného světla, čímž zvýšíme robustnost detekce vzhledem k osvětlení. Dále je zapotřebí vybrat vhodný barevný prostor, se kterým budeme pracovat. Všechny barvy obsažené ve snímcích z testovací množiny zobrazíme v barevných prostorech uvedených v podkapitole 2.2.2, abychom získali představu o rozložení berev zeleně. Ze všech prostorů by pak bylo vhodné vybrat takový, ve kterém budou barvy tvořit kompaktní podmnožinu a půjdou tak snáze separovat. Experimenty ale ukazují, že testovací množina není v žádném prostoru

dostatečně kompaktní. Vybereme tedy prostor YUV, protože odděluje chrominanci a luminance barvy a z prostoru RGB lze získat nejsnáze. V práci [24] navrhují k vytvoření modelu barev v tomto prostoru využít aproximaci histogramu barev obecným normálním rozdělením (vizte podkapitolu 2.3.1). Experimentální výsledky na námi zvolené testovací množině však nebyly příliš přesvědčivé, jak ukazuje obrázek 5.5. Zdá se, že tvar histogramu barev dostatečně neodpovídá normálnímu rozdělení, a proto jej není možné kvalitně aproximovat. Z tohoto důvodu k vytvoření modelu barev použijeme postup dle podkapitoly 2.3.3. Označme výsledný model jako funkci $\mathrm{B}(\mathrm{U}, \mathrm{V})$. Tím získáme pravděpodobnostní model barev, pomocí kterého každý bod ohodnotíme podle chrominance jeho barvy:

$$\mathrm{p}_{\mathrm{barva}}(x, y) = \mathrm{B}(\mathrm{U}(x, y), \mathrm{V}(x, y)),$$

Nyní se zaměřme na přiřazení pravděpodobnosti bodu dle textury snímku. Z provedeného pozorování vyplývá, že travnaté plochy jsou charakterizovány velmi hrubou texturou. Z tohoto důvodu ohodnotíme každý bod přímo úměrně dle jeho textury, přičemž hodnotu 1 přiřadíme maximální nalezené textuře. V práci [24] autoři uvádějí, že charakteristika textury není shodná po celé ploše snímku, ale mění se dle zaostření a vzdálenosti vegetace od objektivu, přičemž změny textury trávy se nejvíce projevují v jasu barvy. Aby se zvýšila úspěšnost detekce, navrhují autoři



Obrázek 5.5: Ukázka aproximace histogramu barev vegetace obecným normálním rozdělením.

využít analýzu ve více rozlišení a v každém rozlišení ohodnotit texturu samostatně. K ohodnocení přitom používají diferenci jasu ve snímku. Analýza ve více rozlišeních však vyžaduje provedení decimace (down-scaling) a následné interpolace (up-scaling) signálu pro každé zvolené rozlišení, což je nevýhodné. Proto v našem algoritmu provedeme ohodnocení textury poněkud odlišným způsobem. Abychom zvýšili robustnost algoritmu vůči proměnám v textuře, myšlenku zpracovávání ve více rozlišeních zachováme, přičemž k ohodnocení textury využijeme dvourozměrnou první derivaci, kterou vypočteme pomocí aproximace diferencí s proměnným krokem. Konkrétně jsme k výpočtu zvolili centrální diferenci:

$$\begin{aligned} \Delta_{\mathrm{hor}}\mathrm{f}(x,y,h) &= \frac{\mathrm{f}(x+h,y)-\mathrm{f}(x-h,y)}{2h}, \\ \Delta_{\mathrm{ver}}\mathrm{f}(\,x,y,h) &= \frac{\mathrm{f}(x,y+h)-\mathrm{f}(x,y-h)}{2h} \end{aligned}$$

a levou resp. pravou diferenci v místech, kde souřadnice $x+h$, $y+h$ resp. $x-h$ a $y-h$ nejsou platné:

$$\Delta_{\mathrm{hor}}\mathrm{f}(x,y,h) = \frac{\mathrm{f}(x+h,y)-\mathrm{f}(x,y)}{h},\ \Delta_{\mathrm{ver}}\mathrm{f}(x,y,h) = \frac{\mathrm{f}(x,y+h)-\mathrm{f}(x,y)}{h},$$

$$\Delta_{\mathrm{hor}}\mathrm{f}(x,y,h) = \frac{\mathrm{f}(x,y)-\mathrm{f}(x-h,y)}{h},\ \Delta_{\mathrm{ver}}\mathrm{f}(x,y,h) = \frac{\mathrm{f}(x,y)-\mathrm{f}(x,y-h)}{h}.$$

Funkcí $\mathrm{f}(x,y)$ reprezentujeme hodnotu daného barevného kanálu ($\mathrm{Y}(x,y)$, $\mathrm{U}(x,y)$, nebo $\mathrm{V}(x,y)$) na pozici $[x,y]$. Pro krok diference h zvolíme hodnoty $h = 1,2,4,8,16,32$, čímž napodobíme analýzu ve více rozlišeních a to v rozmezí od vstupního rozlišení $h = 1$ až po velmi malé rozlišení $h = 32$. Výsledné ohodnocení v daném směru stanovíme jako maximum z absolutních hodnot všech diferencí:

$$\Delta_{\mathrm{i}}\mathrm{f}(x,y) = \max\{|\,\Delta_{\mathrm{i}}\mathrm{f}(x,y,h)| : h \in \{1,2,4,8,16,32\}\},$$

kde i symbolizuje horizontální nebo vertikální diferenci. Každému bodu pak přiřadíme hodnotu odpovídající součtu maximální vertikální a maximální horizontální diference:

$$\Delta\mathrm{f}(x,y) = \Delta_{\mathrm{hor}}\mathrm{f}(x,y) + \Delta_{\mathrm{ver}}\mathrm{f}(x,y).$$

Hodnotu textury snímku v bodě $[x,y]$ potom stanovíme jako součet diferencí všech barevných složek:

$$\mathrm{T}(x,y) = \Delta\mathrm{Y}(x,y) + \Delta\mathrm{U}(x,y) + \Delta\mathrm{V}(x,y).$$

Jak již bylo řečeno, každému bodu přiřadíme takovou pravděpodobnost, aby maximální nalezené textuře odpovídala hodnota 1. Abychom však vyloučili, že toto globální maximum je dáno šumem, provedeme pomocí konvolučního filtru filtraci

diferencí T, čímž získáme modifikovanou texturu T′. Na závěr každému bodu přiřadíme pravděpodobnost:

$$\mathrm{p}_{\mathrm{textura}}(x,y) = \frac{\mathrm{T}'(x,y)}{T'_{max}}, \tag{5.2}$$

$$T'_{max} = \max_{[x,y]\in D(\mathrm{T}')} \mathrm{T}'(x,y), \tag{5.3}$$

přičemž $D(\mathrm{T}')$ značí definiční obor $\mathrm{T}'(x,y)$, tedy množinu všech bodů ve snímku.

Zpracování

Ukazuje se, že vypuštěním jasové složky dochází k nepatrnému pozitivnímu ohodnocení barev, které určitě nepříslušejí zeleni. Abychom tento nežádoucí jev potlačili, provedeme dodatečnou úpravu pravděpodobnosti. Experimentálně se nám osvědčila funkce, která potlačuje malé hodnoty pravděpodobností, ale vyšší hodnoty ponechává beze změny, konkrétně:

$$\mathrm{F}(p) = \begin{cases} p, & p > t_2, \\ \max\left\{\frac{t_2}{t_2-t_1}p - \frac{t_1 t_2}{t_2-t_1}, 0\right\}, & p \le t_2, \end{cases} \tag{5.4}$$

kde $t_1, t_2 \in \langle 0,1\rangle$ jsou parametry transformace, přičemž platí $\mathrm{F}(t_1) = 0$ a $\mathrm{F}(t_2) = t_2$. Parametr t_1 udává interval $\langle 0, t_1\rangle$, který bude transormován na nulu a parametr t_2 udává interval $\langle t_2, 1\rangle$, který bude ponechán beze změny. Zbylý interval (t_1, t_2) je lineárně transformován na interval $(0, t_2)$. Graf této funkce pro zvolené parametry zobrazuje obrázek 5.6.

Abychom však nepotlačovali ohodnocení bodů v oblastech zeleně, budeme transformaci ohodnocení bodu provádět lokálně na základě hodnot v jeho okolí (velikost okolí označme t_3). Pokud maximální hodnota v okolí je větší než parametr t_4, předpokládáme, že daný bod může tvořit zeleň a jeho ohodnocené ponecháme beze změny. V opačném případě předpokládáme, že bod je součást pozadí a ohodnocení transformujeme:

$$\mathrm{p}'_{\mathrm{barva}}(x,y) = \begin{cases} \mathrm{p}_{\mathrm{barva}}(x,y), & \mathrm{p}^{\max}_{\mathrm{barva}}(x,y) \ge t_4, \\ \mathrm{F}(\mathrm{p}_{\mathrm{barva}}(x,y)), & \mathrm{p}^{\max}_{\mathrm{barva}}(x,y) < t_4, \end{cases}$$

$$\mathrm{p}^{\max}_{\mathrm{barva}}(x,y) = \max\{\mathrm{p}_{\mathrm{barva}}(x+i, y+j) : i,j = -t_3, \ldots, t_3\}.$$

Experimentální výsledky dále ukazují, že metrika textury vykazuje v rámci vegetace velké fluktuace. Tento jev je pravděpodobně způsoben velkým rozptylem barev v oblastech vegetace, především na úrovni jednotlivých pixelů. Jak ukazuje obrázek

5.7, body nabývají široké škály barev. Tyto změny se projevují ve výsledcích i přes to, že používáme analýzu s proměnlivým krokem. Vliv fluktuací se pokusíme zmenšit tak, že hodnotu každého bodu definujeme jako maximum ze všech hodnot z jeho okolí:

$$\mathrm{p}'_{\mathrm{textura}}(x,y) = \max\{\mathrm{p}_{\mathrm{textura}}(x+i, y+j) : i,j = -t_5, \ldots, t_5\},$$

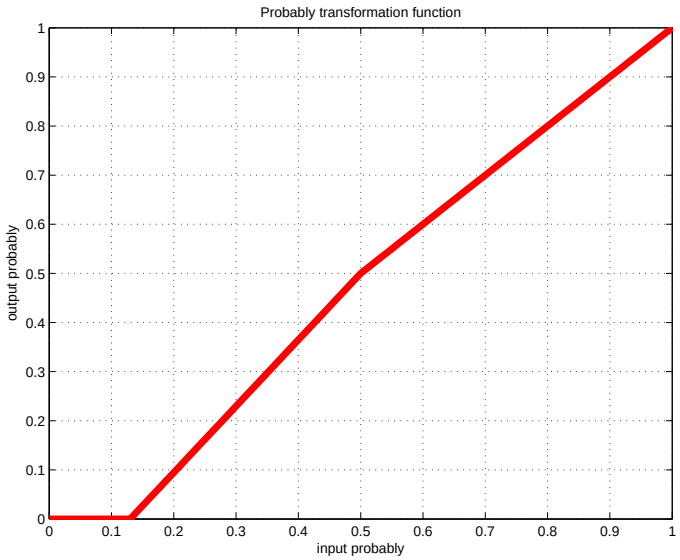
kde t_5 je velikost použitého okolí. Tím se fluktuace pravděpodobnosti v rámci oblasti vegetace zmenší, aniž by se tím zvýšilo ohodnocení pozadí. Protože chceme, aby v rámci zelené vegetace byly splněny oba požadavky, stanovíme celkovou pravděpodobnost jako následující součin:

$$\mathrm{p}(x,y) = \mathrm{p}'_{\mathrm{barva}}(x,y)\,\mathrm{p}'_{\mathrm{textura}}(x,y).$$

Uvedeným zpracováním se však zvýrazní také místa v pozadí, která obsahují šum. Pro potlačení tohoto nežádoucího jevu, použijeme filtraci finální pravděpodobnosti pomocí konvolučního filtru (vizte podkapitolu 2.2.3). Filtrací by mělo dojít k potlačení zašuměných částí, avšak hodnoty pravděpodobností v rámci zelených ploch by se neměly významně potlačit.

Segmentace a klasifikace

Abychom nalezli oblasti odpovídající zeleni, provedeme pomocí následující metody segmentaci pravděpodobnostního ohodnocení. Nejprve aplikujeme poloprahování



Obrázek 5.6: Průběh zvolené transformační funkce pro parametry $t_1 = 0{,}13$, $t_2 = 0{,}5$.

(a) (b)

Obrázek 5.7: Příklad travnaté plochy. Obrázek (a) ukazuje část plochy zobrazující trávu, obrázek (b) zobrazuje 12,5 krát zvětšený výřez prvního snímku. Zde je vidět, že travnaté plochy nabývají široké škály barev.

s prahy t_6 a t_7 ($t_6 > t_7$). Parametr t_6 volíme relativně velký, čímž získáme oblasti, které s velkou pravděpodobností odpovídají vegetaci. Předpokládáme, že takových oblastí ve snímku nebude mnoho a jejich plocha bude velmi malá. K jejich rozšíření použijeme metodu rozšiřování oblastí (vizte podkapitolu 2.2.4), přičemž další bod k oblasti připojíme, pokud bude jeho hodnota vyšší než práh t_7. Tímto způsobem získáme oblasti, jejichž body mají hodnotu vyšší než práh t_7, avšak obsahují také body s hodnotami vyššími než práh t_6. Díky tomu můžeme práh t_7 volit poměrně nízký (blízko hodnotám šumu), ale oblasti způsobené pouze šumem segmentovány nebudou.

Protože nelze předem dělat jakékoli předpoklady o tvaru oblastí vegetace, připouštíme v rámci anotací libovolný tvar. V navrženém algoritmu tak bude proces segmentace zastupovat také klasifikaci, neboť všechny nalezené segmenty budou klasifikovány jako vegetace.

Anotace

Na závěr algoritmu provedeme anotaci všech segmentů, jejichž plocha je větší než t_8. Vyhovující oblasti ohraničíme obdélníkem. U ostatních oblastí předpokládáme, že jsou vzhledem ke své velikosti zanedbatelné a jedná se pravděpodobně o šum. Protože vegetace může nabývat libovolného tvaru, může se stát, že některé obdélníky budou vnořené do jiných. V takovém případě předpokládáme, že vnořené anotace ztrácejí svůj význam a proto je do výsledku nebudeme zahrnovat.

5.3.2 Implementace

Algoritmus pro detekci zelené vegetace je implementován v modulu `plugin_vegetation_detect`. Pravděpodobnostní model barev je uložen v souboru `vegetation_model.mtx`. K načtení modelu slouží metoda `loadModel(...)`, přičemž alokace modelu probíhá v metodě `load()` během vytváření instance zásuvného modulu. Dealokace paměti modelu naopak probíhá v metodě `unload()`.

Experimentální výsledky algoritmu byly nejlepší pro následující parametry. Parametry transformační funkce F jsme zvolili $t_1 = 0{,}13$, $t_2 = 0{,}5$. Transformovány jsou tedy pravděpodobnosti menší než 0,5, přičemž hodnotám nižším než 0,13 je přiřazena nula. Konvoluční filtr jsme zvolili takový, který přikládá všem bodům stejný význam, přičemž jsme zvolili filtr o rozměru 11×11. Konkrétní hodnoty jsou tedy následující:

$$F = \frac{1}{121} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}.$$

Pro zpracování pravděpodobnostního ohodnocení jsme zvolili práh $t_4 = 0{,}5$ a okolí o velikosti $t_3 = t_5 = 4$. Menší hodnota okolí zhoršovala úspěšnost detekce, zatímco se zvyšujícím se okolím se zvyšoval počet chybných detekcí. Parametry segmentace jsme zvolili $t_6 = 0{,}5$ a $t_7 = 0{,}1$. Anotovány jsou pak takové segmenty, jejichž absolutní velikost je větší než 0,5 % celkové plochy snímku.

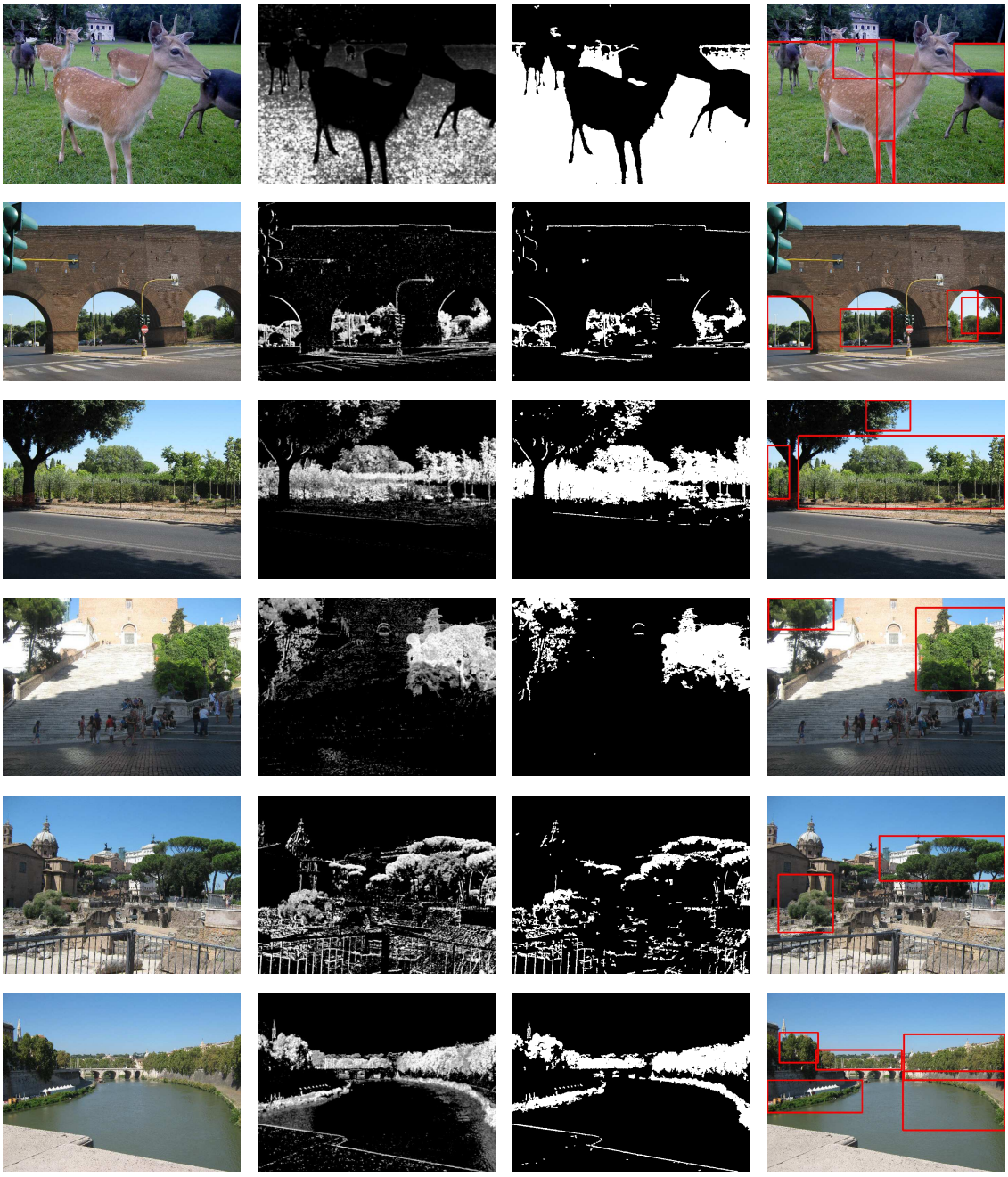
5.3.3 Výsledky

K otestování navrženého algoritmu jsme použili 300 snímků. Výsledky získané jejich anotací znázorňuje tabulka 5.4. Během testování na vybraných snímcích se ukazuje, že algoritmus dosahuje úspěšnosti detekce vyšší než 90 %. Obrázek 5.8 zobrazuje příklady vybraných úspěšných detekcí. Algoritmus však vykazuje také mnoho chybných detekcí. Ukazuje se, že pro některé zelené objekty není metrika založená na základě barvy a textury dostačující. Příklad několika chybných detekcí ukazuje obrázek 5.9.

	absolutní počet	relativní počet
správně detekováno	183	93,84 %
chybně detekováno	73	24,33 %
nenalezeno	12	6,15 %

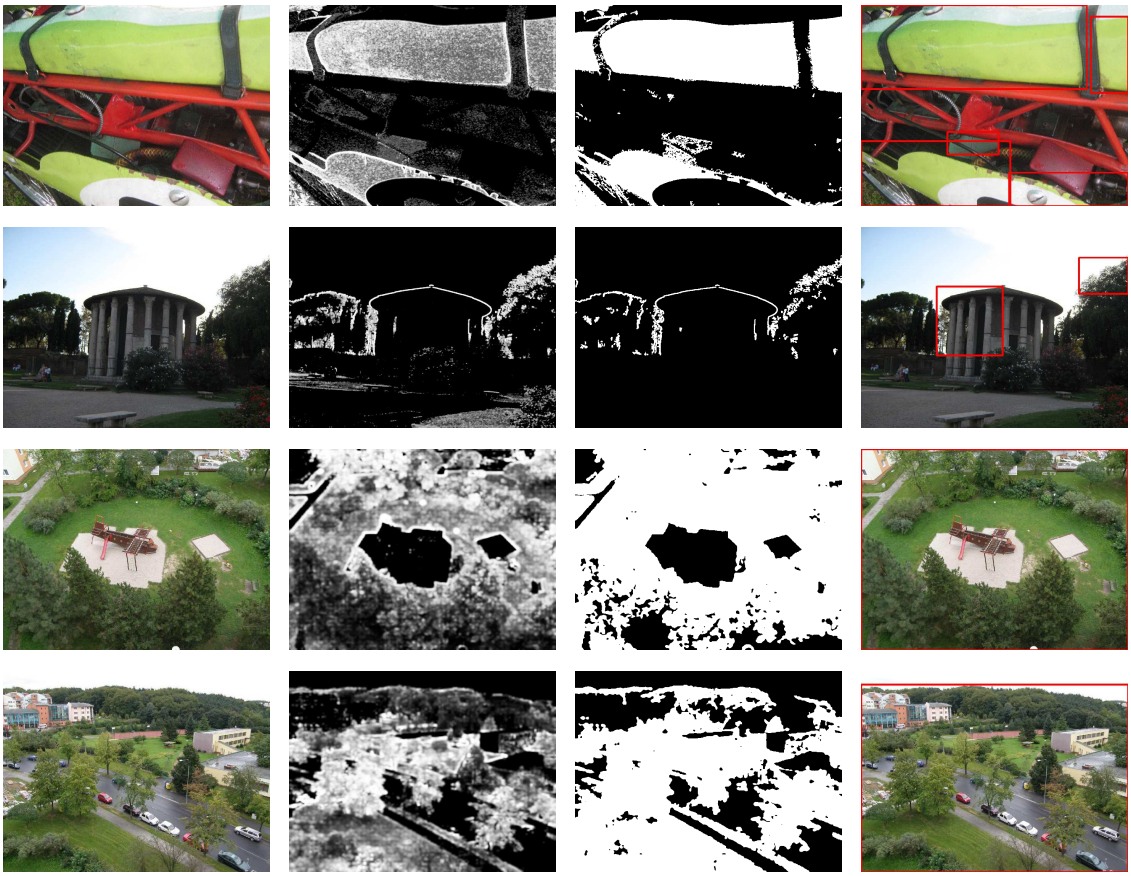
Tabulka 5.4: Výsledky anotace zelené vegetace.

Další nevýhodou algoritmu je provádění anotací pomocí ohraničení obdélníkem. Z výsledků je patrné, že tím ztratíme velmi cenné informace o tvaru nalezených oblastí, jeden takový příklad ukazuje obrázek 5.9. Proto by bylo lepší zvolit jiný způsob identifikace nalezených oblastí.



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

Obrázek 5.8: Úspěšně detekovaná zelená vegetace. Obrázky ve sloupci (a) zobrazují originální snímky, sloupec (b) odpovídá pravděpodobnostnímu ohodnocení, (c) segmentaci a (d) zobrazuje navržené anotace. Všechny snímky pochází ze zdroje II dle tabulky B.1.



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

Obrázek 5.9: Neúspěšná detekce vegetace. Obrázek zobrazuje chybné detekce a také problém obdélníkového ohraničení. Význam sloupců je stejný jako v předchozím obrázku. Snímky jsou převzaty ze zdroje II podle tabulky B.1.

5.4 Modul pro detekci modrého nebe

V této části se zaměříme na návrh a implementaci algoritmu pro anotaci modré oblohy. Ta je součástí mnoha snímků a je typická především pro venkovní fotografie. Její rozpoznání tak může významně pomoci právě při jejich klasifikaci.

5.4.1 Návrh algoritmu

Obloha může nabývat velkého množství barev, od modré přes červenou až po bílou. Stejně tak případné mraky mohou být velmi rozdílné a mohou nabývat různých podob, tvarů a také barev. Z tohoto důvodu je velmi obtížné provést dostatečně univerzální zjednodušení, které by umožnilo detekovat všechny uvedené případy. Abychom si situaci trochu ulehčili, budeme dále předpokládat převážně jednobarevné nebe, které může být reprezentováno především modrou barvou, někdy i barvou bílou. Na základě vizuální analýzy několika snímků takové oblohy můžeme konstatovat, že část snímku příslušející obloze je charakterizována především svou barvou, texturou a polohou. Barva je většinou po celé ploše téměř neměná, pokud ke změnám barvy dochází, projevují se velmi pomalu a většinou pouze ve vertikálním směru. Pokud budeme předpokládat správně orientovaný snímek, můžeme říci, že nebe je umístěno v horní polovině snímku. Samozřejmě existují snímky, u kterých toto splněno není, předpokládáme však, že jich nebude mnoho.

Navržený algoritmus pro detekci nebe se skládá z pěti kroků: předzpracování snímku, výpočet pravděpodobnostního ohodnocení, segmentace, klasifikace a anotace. Jednotlivé kroky si nyní popíšeme podrobněji.

Předzpracování

Pro snazší analýzu a následné zpracování provedeme nejprve jeho rozmazání, čímž potlačíme drobné lokální změny jasu, které se ve snímku mohou vyskytovat. K tomuto účelu lze využít konvoluční filtr, o kterém jsme pojednávali v podkapitole 2.2.3.

Výpočet pravděpodobnostního ohodnocení

Úkolem této části algoritmu je stanovit pravděpodobnost, zda daný bod snímku představuje nebe. Na základě uvedeného pozorování důležitých vlastností oblohy vyjádříme pravděpodobnost každého bodu součinem tří pravděpodobnostních funkcí: $\mathrm{p}_1(x, y) = \mathrm{p}_{\mathrm{barva}} \cdot \mathrm{p}_{\mathrm{textura}} \cdot \mathrm{p}_{\mathrm{pozice}}$. Význam těchto základních funkcí je následující.

První funkce $\mathrm{p}_{\mathrm{barva}}$ přiřazuje danému bodu pravděpodobnost na základě jeho barvy. V našem případě jsme k ohodnocení zvolili diskrétní funkci:

$$\mathrm{p}_{\mathrm{barva}}(x,y) = \begin{cases} 1, & \vec{\mathrm{c}}(x,y) \in C, \\ 0, & \vec{\mathrm{c}}(x,y) \notin C, \end{cases}$$

kde vektor $\vec{\mathrm{c}}(x,y)$ reprezentuje barvu bodu na pozici $[x,y]$ ve vybraném barevném prostoru a C označuje podprostor zvoleného barevné prostoru, který obsahuje pouze takové barvy, jenž mohou příslušet nebi. Bodům, jejichž barva spadá do množiny C, je pak přiřazena hodnota 1, ostatním bodům hodnota 0. Barevný prostor vybereme takový, v němž bude možné podprostor C separovat co nejjednodušeji. Pro lepší představu o rozložení barev oblohy v jednotlivých barevných prostorech sestavíme množinu snímků, které zobrazují pouze oblohu, a všechny barvy obsažené v těchto snímcích pak zobrazíme v prostorech HSL, HSV, RGB a YUV. Obdržené výsledky zobrazují obrázky 5.10a - 5.10d. Jak můžeme vidět, v prostorech RGB a YUV není možné vzniklý podprostor jednoduše separovat. Zato v prostorech HSV a HSL je situace o mnoho přijatelnější. Na základě tohoto pozorování vybereme barevný protsor HSL, ve kterém lze podprostor snadno definovat s využitím po částech lineární funkce. Za tímto účelem se nám osvědčil následující parametrický popis podprostoru:

$$C = \{[H,S,L] : (H > 175^\circ \wedge H < 270^\circ \wedge S > 20 \wedge L > 20) \vee (L > 98)\}.$$

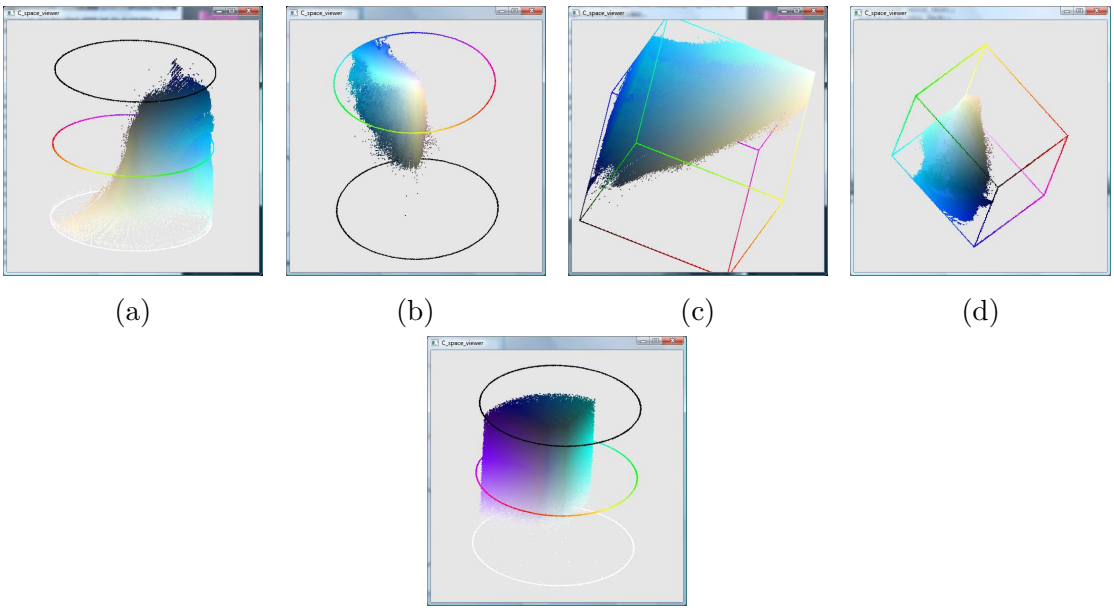
Množinu všech přípustných barev pro lepší představu znázorňuje obrázek 5.10e.

Další funkce $\mathrm{p}_{\mathrm{textura}}$ slouží k ohodnocení textury snímku. Protože předpokládáme, že v rámci oblohy nebude docházet k žádným změnám, definujeme funkci tak, aby bodům přiřazovala pravděpodobnost nepřímo úměrně dle změn ve snímku. Při definování této funkce jsme se inspirovali prací [25], výsledný předpis funkce je:

$$\begin{aligned} \mathrm{p}_{\mathrm{textura}} &= \mathrm{p}_{\mathrm{SAD}} \cdot \mathrm{p}_{\mathrm{grad}}, \\ \mathrm{p}_{\mathrm{SAD}} &= e^{-[C_{textura}(\mathrm{SAD}_{\mathrm{hor}} + \mathrm{SAD}_{\mathrm{ver}})]^2}, \\ \mathrm{p}_{\mathrm{grad}} &= e^{-[C_{textura}(\mathrm{grad}_{\mathrm{hor}} + \mathrm{grad}_{\mathrm{ver}})]^2}, \end{aligned}$$

přičemž funkce $\mathrm{SAD_i}$ a $\mathrm{grad_i}$ reprezentují použitý hranový detektor a lze je stanovit jako:

$$\begin{aligned} \mathrm{SAD}_{\mathrm{hor}}(x,y) &= \frac{1}{N_{SAD}} \sum_{i=-w}^{w} \sum_{j=-w}^{w-1} |\,\mathrm{B}(x+i,y+j) - \mathrm{B}(x+i,y+j+1)|, \\ \mathrm{SAD}_{\mathrm{ver}}(x,y) &= \frac{1}{N_{SAD}} \sum_{i=-w}^{w-1} \sum_{j=-w}^{w} |\,\mathrm{Y}(x+i,y+j) - \mathrm{Y}(x+i+1,y+j)|, \\ \mathrm{grad}_{\mathrm{hor}}(x,y) &= \frac{1}{N_{grad}} \left(\sum_{i=-w}^{w} \sum_{j=-w}^{-1} \mathrm{B}(x+i,y+j) - \sum_{i=-w}^{w} \sum_{j=1}^{w} \mathrm{B}(x+i,y+j) \right), \\ \mathrm{grad}_{\mathrm{ver}}(x,y) &= \frac{1}{N_{grad}} \left(\sum_{i=-w}^{-1} \sum_{j=-w}^{w} \mathrm{Y}(x+i,y+j) - \sum_{i=1}^{w} \sum_{j=-w}^{w} \mathrm{Y}(x+i,y+j) \right), \end{aligned}$$



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

(e)

Obrázek 5.10: Zobrazení barev oblohy ve vybraných barevných prostorech. Na obrázcích (a) - (d) jsou zobrazeny barvy z testovací množiny v barevných prostorech HSL, HSV, RGB a YUV. Obrázek (e) zobrazuje vybraný podprostor prostoru HSL.

kde B, Y označuje modrou složku, resp. jas, w velikost okolí a N_{SAD}, N_{grad} plochu použitého okolí. Jak vidíme, k ohodnocení textury je použita exponenciální funkce. Ta byla vybrána proto, aby hodnota 1 odpovídala bodům, v jejichž okolí nedochází k žádným změnám, a malé hodnoty (blízké 0) odpovídaly bodům, které se na změnách ve snímku podílejí. Platí tedy $\mathrm{p_{SAD}}, \mathrm{p_{grad}} \in \langle 0, 1 \rangle$ a tudíž i $\mathrm{p_{textura}} \in \langle 0, 1 \rangle$. Při výpočtu se předpokládá, že $\mathrm{R}, \mathrm{G}, \mathrm{B} \in \langle 0, 1 \rangle$, přičemž jasová složka Y je definována dle vztahu (2.3). Vlivem exponenciální funkce je navrhovaný detektor velmi citlivý, přičemž rychlost klesání a tedy citlivost použitého hranového detektoru lze určit parametrem $C_{textura}$. Jako hranový detektor jsou použity funkce absolutních odchylek[1] a funkce gradientu. Protože není rozumné posuzovat texturu pouze podle jednoho bodu, jsou tyto funkce počítány v okolí až do vzdálenosti w. Poznamenejme, že oproti práci [25] jsme provedli modifikaci, neboť horizontální část ohodnocení počítáme z modré složky (B), místo původního výpočtu z jasové informace Y. Úkolem této úpravy je zmenšit citlivost na změny, které jsou v nebi způsobeny především mračny, a tím zvýšit úspěšnost detekce. Podle práce [8] vykazuje totiž modrá složka menší kontrast v rámci nebe a naopak větší kontrast mezi nebem a pozadím než jasová informace. Výsledky tohoto pozorování jsme si experimentálně potvrdili. Samozřejmě by bylo vhodné použít modrou složku i při výpočtu vertikální části textury.

[1] Zkratka SAD (sum of absolute differences) označuje sumu absolutních změn.

Tím bychom si ovšem zkomplikovali zpracování na snímcích, které např. zobrazují přechod mezi nebem a mořem apod.

Pravděpodobnostní funkce polohy je převzata z [25] a je definována jako:

$$\mathrm{p}_{\mathrm{pozice}}(x, y) = e^{-\left(\frac{y}{\mathrm{V}}\right)^2},$$

kde V označuje výšku snímku. Jak vidíme, tato funkce je definována tak, že bodům přiřazuje pravděpodobnost v intervalu $\langle e^{-1} = 0{,}36; e^0 = 1\rangle$, přičemž 1 přísluší bodům na horním okraji snímku ($y = 0$) a hodnota 0,36 bodům na dolním okraji snímku ($y = V$).

Segmentace

V předchozím kroce jsme získali pravděpodobnostní ohodnocení všech bodů snímku. Abychom v dalším zpracování neuvažovali body, jejichž pravděpodobnost je velmi malá, provedeme jednoduché prahování a body s hodnotou nižší než t_1 zanedbáme:

$$\mathrm{p}_2(x, y) = \begin{cases} \mathrm{p}_1(x, y), & \mathrm{p}_1(x, y) \geq t_1, \\ 0, & \mathrm{p}_1(x, y) < t_1. \end{cases} \tag{5.5}$$

K volbě prahu použijeme speciální metodu, která je navržena v práci [25]. Práh je stanoven na základě vážené distribuční funkce pravděpodobnostního ohodnocení, kterou získáme jako:

$$\mathrm{H}'(i) = \mathrm{W}(i)(1 - \mathrm{H}(i)),$$

kde $\mathrm{H}'(i)$ označuje váženou distribuční funkci, $\mathrm{H}(i)$ distribuční funkci pravděpodobnostního ohodnocení a $\mathrm{W}(i)$ váhovou funkci. Ta je volena tak, aby potlačovala malé hodnoty a posilovala velké hodnoty pravděpodobnosti. Připomeňme, že distribuční funkce $\mathrm{H}(i)$ udává pravděpodobnost, s jakou se v pravděpodobnostním ohodnocení snímku vyskytují hodnoty v itervalu $\langle 0, i\rangle$. Podle práce [25] bude funkce $\mathrm{H}'(i)$ obsahovat vždy jedno globální maximum, přičemž hodnota i, ve které funkce nabývá svého maxima, je zvolena jako práh t_1:

$$t_1 = \arg \max_{i \in \langle 0,1\rangle} (\mathrm{H}'(i)).$$

Váhovou funkci $\mathrm{W}(i)$ jsme v našem případě zvolili lineární, aby váha vždy odpovídala pravděpodobnosti i:

$$\mathrm{W}(i) = i.$$

Protože dále již není rozumné snímek zpracovávat na základě jednotlivých bodů, použijeme segmentaci pravděpodobnostního ohodnocení, abychom nalezli segmenty

se stejnými vlastnostmi. K tomu použijeme segmentační metodu typu Split and Merge, kterou jsme uvedli v podkapitole 2.2.4. Během slučování provedeme spojení dvou sousedních bodů, jestliže hodnota jejich pravděpodobností se liší o méně než parametr t_2. Tímto způsobem obdržíme takové segmenty snímku, pro které platí: barva segmentů odpovídá obloze, v rámci segmentů nedochází k velkým změnám a segment je umístěn převážně v horní části snímku.

Každému nalezenému segmentu S pak přiřadíme hodnotu na základě pravděpodobností jeho bodů a jeho polohy:

$$p_3^S = p_{avg}^S \cdot p_{pozice}^S,$$

kde p_{avg}^S představuje průměrnou hodnotu pravděpodobností bodů segmentu S a pravděpodobnost p_{pozice}^S ohodnocuje segment dle jeho pozice. Tyto hodnoty lze stanovit podle následujících vztahů:

$$\begin{aligned} p_{avg}^S &= \frac{\sum_{[x,y]\in S} \mathrm{p}_1(x,y)}{\sum_{[x,y]\in S} 1}, \\ p_{pozice}^S &= \max\left\{\left(1 - \frac{y_{min}^S}{4 \cdot y_{avg}}\right), 0\right\}, \end{aligned}$$

kde y_{min}^S odpovídá nejmenší vertikální souřadnici segmentu S a y_{avg} vážené průměrné vertikální souřadnici z celého snímku, přičemž jako váha je použita pravděpodobnost p_2. Tím je zaručeno, že nejsou uvažovány body, které nesplnily podmínku (5.5). Pravděpodobnost p_{pozice}^S je definována tak, aby segmentům na horním okraji snímku byla přiřazena hodnota 1, a segmentům, jejichž horní poloha je menší nebo rovna čtyřnásobku průměrné vertikální polohy byla přiřazena hodnota 0. Jak vidíme, přechod mezi těmito krajními hodnotami je lineárně závislý na vertikální poloze. Hodnotu 4 jsme zvolili na základě exprimentů, neboť ohodnocení s tímto parametrem vykazovalo velmi dobré výsledky. Minimální a průměrnou vertikální polohu segmentu lze vypočítat jako:

$$\begin{aligned} y_{min}^S &= \min_{\forall x}\{y, [x,y] \in S\}, \\ y_{avg} &= \frac{\sum_{[x,y]} y \cdot \mathrm{p}_2(x,y)}{\sum_{[x,y]:\mathrm{p}_2(x,y)>t_1} 1}. \end{aligned}$$

Klasifikace a anotace

Na závěr je potřeba přijmout takové segmenty, které s velkou pravděpodobností odpovídají nebi a ty poté anotovat. V tomto algoritmu klasifikujeme jako nebe takové segmenty, jejich absolutní velikost je větší než t_3, a kde přiřazená finální hodnota pravděpodobnosti segmentu je vyšší než t_4.

Samozřejmě předpokládáme, že jako nebe může být klasifikována více než jedna komponenta. Protože však obloha jako objekt může být na snímku pouze jedna, označíme všechny nalezené segmenty pouze jedinou anotací, resp. anotujeme sjednocení všech nalezeních segmentů. Tento přístup je velmi výhodný, protože segmentace oblohy nemusí být naprosto precizní, chyba v řádu několika desítek bodů nebude mít na nalezenou hranici téměř žádný vliv. Máme tedy jistou benevolenci, co se týče chyby v segmentační fázi.

5.4.2 Implementace

Navržený algoritmus je implementován v modulu `plugin_sky_detect`. Protože algoritmus je velmi snadno implementovatelný, popíšeme si v této podkapitole pouze parametry zvolené během implementace.

Předzpracování

Koeficienty konvolučního filtru jsme zvolili tak, aby největší váhu měl právě zpracovávaný bod a váha ostatních bodů exponenciálně klesala se vzdáleností. Vybrané hodnoty jsou dány dvourozměrným normálním rozdělením s rozptyly $\sigma_x^2 = \sigma_y^2 = 2$. Velikost filtru jsme vybrali 5×5, konkrétní hodnoty filtru jsou následující:

$$F = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}.$$

Výpočet prvotní pravděpodobnosti

Pro nastavení citlivosti hranového detektoru jsme zvolili parametr $C_{textura} = 20$. Vyšší hodnota již téměř není použitelná, neboť jsme omezeni šumem ve snímku a také artefakty způsobenými JPG kompresí. Pro výpočet funkce $\mathrm{SAD_i}$ a $\mathrm{grad_i}$ jsme použili okno velikosti $w = 3$, jemuž odpovídají normalizační hodnoty $N_{SAD} = (2w + 1) \cdot 2W = 42$, $N_{grad} = (2w + 1) \cdot W = 26$.

Segmentace

Prahovací hodnota t_1 je v algoritmu volena adaptivně. Poznamenejme, že při výpočtu vážené distribuční funkce pravděpodobnostního ohodnocení jsme pravděpodobnostní interval rozdělili na 1000 částí. Parametr t_2 metody Split and Merge jsme zvolili 0,1. Ke spojení dvou sousedních bodů dojde, jestliže rozdíl jejich hodnot není vyšší než 0,1.

Klasifikace a anotace

Parametr segmentace t_3 je automaticky nastavován tak, aby odpovídal 1 % plochy celého snímku. Parametr t_4 je volen 0,7, segment je tedy klasifikován jako obloha, jestliže je jeho pravděpodobnost vyšší než 0,7.

5.4.3 Výsledky

K otestování navrženého algoritmu jsme vybrali 284 snímků, které mohou obsahovat libovolnou oblohu, od čisté až po úplně zamračenou. Vybrané snímky pocházejí z několika různých zdrojů a jsou pořízené především během dne. Výsledky anotace zobrazuje tabulka 5.5. Jak vidíme, úspěšnost detekce oblohy na vybrané množině snímků je vyšší než 90 %.

	absolutní počet	relativní počet
správně detekováno	143	91,08 %
chybně detekováno	24	8,45 %
nenalezeno	14	8,92 %

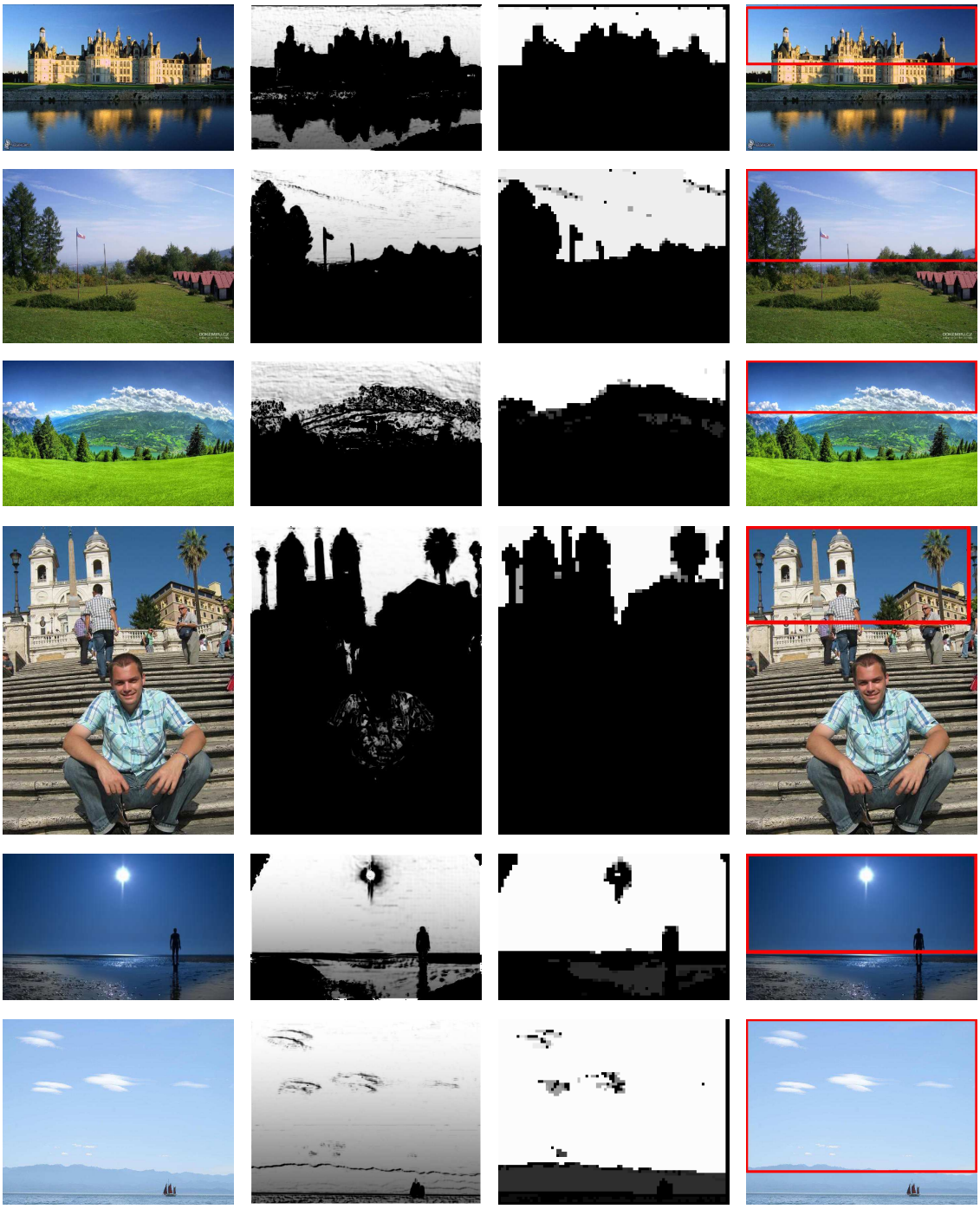
Tabulka 5.5: Výsledky anotace nebe.

Obrázek 5.11 zobrazuje úspěšně detekované nebe na několika vybraných snímcích. První snímek obsahuje čisté nebe spolu s modrou vodní plochou, které jsou však vzájemně mezi sebou oddělené. Druhý snímek zobrazuje lehce zamračené nebe, třetí částečně čisté a částečně zamračené nebe, čtvrtý ukazuje výsledek detekce v případě, kdy je nebe rozděleno na několik oblastí. Na snímku je navíc modrý objekt v podobě košile. Pátý a šestý snímek zobrazují nebe i moře. Naopak obrázek 5.12 zobrazuje snímky, na kterých nebyla detekce úplně úspěšná. První snímek zachycuje velmi zamračenou oblohu. V takovém případě algoritmus označí pouze podoblast nebe. Druhý případ zobrazuje kromě oblohy také hory, přičemž přechod mezi nebem a horami je nepatrný. Algoritmus pak anotuje jako oblohu oba zmíněné objekty.

Na tomto místě bych rád navrhl možná vylepšení uvedeného algoritmu. Ukazuje se, že pravděpodobnostní funkci p_{barva} by bylo vhodnější navrhnout spojitě, aby nedocházelo ke striktnímu zamítnutí některých barev, které jsou na mezi pravděpodobnosti a v některých případech mohou nebe představovat. Pokud obloha obsahuje mraky, může se na fotografiích objevovat mnohem více odstínů barev, především pak odstíny červené barvy. Rozšířením pravděpodobnostního modelu bychom u takových snímků zvýšili úspěšnost detekce.

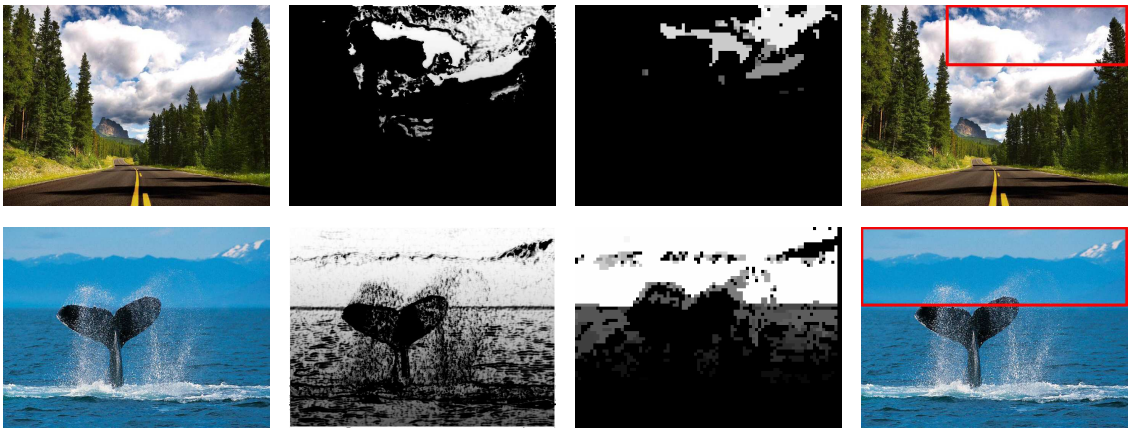
Ve výpočtu textury se pro její vertikální část používá jasová informace, proto aby se nezhoršila detekce oblohy na snímcích s mořem. Pokud bychom však použili speciální algoritmus pro detekci linie mezi nebem a mořem, mohli bychom analýzu modré složky použít, čímž bychom zvýšili odolnost algoritmu proti případným barevným změnán v obloze. Při výpočtu textury by možná bylo také výhodné využít analýzu ve více rozlišeních, zmenšením snímku může totiž dojít k potlačení některých lokálních změn v textuře, které narušují souvislost oblastí oblohy.

Dále by bylo možné zpřesnit segmentaci metodou narůstání oblastí, např. dle postupu uvedeného v [3]. Zde se pravděpodobnostní hodnoty segmentu oblohy aproximují pomocí dvoudimenzionální polynomiální funkce, přičemž klasifikace segmentů je pak prováděna podle aproximační chyby. Využívá se toho, že pravděpodobnosti segmentů oblohy nepodléhají přílišním změnám, tudíž půjdou polynomiální funkcí dobře aproximovat. Výsledná chyba aproximace bude velmi malá. Naopak segmenty neodpovídající obloze půjdou aproximovat mnohem hůře, čímž dojde k razantnímu zvýšení aproximační chyby. Postup je použit také v [25].



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

Obrázek 5.11: Úspěšně detekované nebe. Sloupec (a) zobrazuje originální snímek, (b) navrhované pravděpodobnostní ohodnocení, (c) výsledky segmentace, (d) navržené anotace. Snímky jsou získány (bráno od shora) ze zdrojů VIII, XV, IV, II, III, XVI dle tabulky B.1.



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

Obrázek 5.12: Nejednoznačně detekované nebe. Význam sloupců je stejný jako v předchozím obrázku. Snímky jsou převzaty (bráno od shora) ze zdrojů V, VI podle tabulky B.1.

5.5 Modul pro detekci speciálních snímků

Hlavním úkolem tohoto modulu je detekce speciálních snímků. Na ukázku jsme si vybrali detekci speciálních efektů low key a high key a detekci ostrých částí snímku. Tento modul by měl demonstrovat návrh a implementaci algoritmů, které neprovádí detekci objektů, ale vytvářejí anotace na základě vlastností snímků.

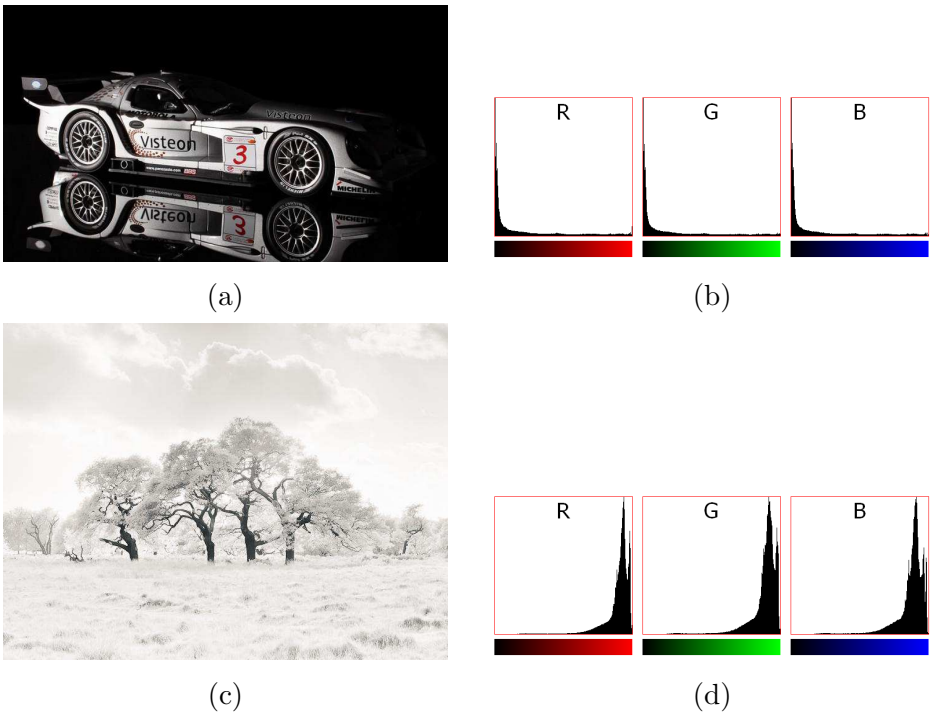
5.5.1 Návrh algoritmu

Low a High key

Nejprve se zaměříme na vybrané efekty low key a high key. Protože se jedná o umělecké úpravy fotografií, nelze specifikovat přesnou hranici mezi takto upravenými a ostatními snímky. Můžeme však říci, že ve snímku typu low key převládají tmavé odstíny barev, zatímco v high key převládají světlé odstíny barev. Obrázek 5.13 ukazuje typické zástupce těchto typů fotografií včetně histogramů barevných složek v prostoru RGB. Z uvedeného příkladu je patrné, že low a high key snímky lze odlišit právě podle jejich histogramu. Na základě tohoto pozorování navrhněme následující velmi jednoduchou metodu. Nejprve pro každý snímek vypočteme relativní histogram četnosti jasu. Vzhledem k tomu, že relativní četnost jasu ve snímky odpovídá pravděpodobnosti, s jakou se daný jas ve snímku vyskytuje, můžeme histogram snímku považovat za pravděpodobnostní funkci diskrétní náhodné veličiny. Definiční obor náhodné veličiny je dán použitou barevnou hloubkou. Aby byl výpočet univerzální, provedeme normalizaci definičního oboru náhodné veličiny na interval $\langle 0, 1\rangle$, čímž zajistíme nezávislost na použité barevné hloubce. Pro takto upravenou pravděpodobnostní funkci dále stanovíme p-procentní kvantil který použijeme ke klasifikaci snímku. Pokud se kvantil bude nacházet v dolní části definičního oboru, tj. $p \in \langle 0, t_1\rangle$, je pravděpodobné, že snímek obsahuje především tmavé odstíny barev, a tudíž se může jednat o snímek typu low key. Naopak, pokud se kvantil bude nacházet v intervalu $\langle t_2, 1\rangle$, je více pravděpodobný výskyt světlých odstínů. Potom by se jednalo o high key fotografii. Experimenty ukazují, že uvedená metrika je ke klasifikaci snímků dostačující. Dokonce se ukazuje, že ani není potřeba provádět analýzu všech složek modelu RGB, ale stačí pouze analýza jasové složky. V takovém případě k výpočtu jasu použijeme vztah (2.3).

Nalezení ostrých částí snímku

Druhou částí modulu je detekce ostrých částí snímku. Ta by měla sloužit k identifikaci např. takových snímků, kde je ostrá pouze oblast snímku obsahující foto-



Obrázek 5.13: Příklad low a high key snímků. Obrázek (a) zobrazuje typický příklad low key snímku, (b) histogram barevných kanálů v prostoru RGB a obdobně obrázek (c) zobrazuje high key snímek a (d) jeho histogram. Snímky jsou získány (bráno od shora) ze zdrojů XIV, VII dle tabulky B.1.

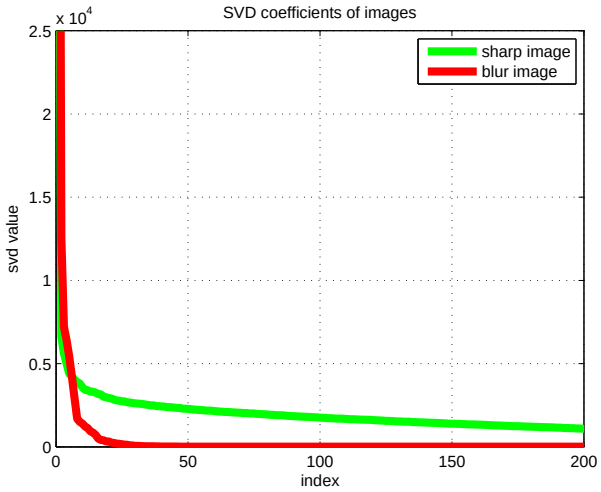
grafovaný objekt a ostatní části jsou rozmazané. Detekce ostré části pak spolu s dalšími algoritmy může posloužit k identifikaci uměleckých a jinak specifických snímků. Mnoho prací k tomuto úkolu využívá diskrétní Fourierovu transformaci [11, 9], diskrétní waveletovou transformaci [20, 16], či detekci a následnou analýzu hran [27, 6, 1]. V této práci se budeme soustředit na přístup publikovaný v [18], který je založen na singulárním rozkladu matice. Dle této teorie pro každou matici $I \in \mathbb{R}^{n\times m}$ (předpokládejme $n < m$) existují unitární matice $U \in \mathbb{C}^{n\times n}$, $V \in \mathbb{C}^{m\times m}$ takové, že:

$$I = U\Sigma V^*, \tag{5.6}$$

kde $\Sigma \in \mathbb{R}^{n\times m}$ je diagonální matice, jejíž prvky jsou singulární čísla uspořádaná v sestupném pořadí. Protože matice Σ je diagonální, můžeme rovnici (5.6) převést do tvaru:

$$I = \sum_{i=1}^{n} \sigma_i u_i v_i^*,$$

kde u_i a v_i jsou i-té sloupce matic U a V a σ_i je prvek matice Σ na pozici $[i, i]$. Pokud si tento vztah pozorně prohlédneme, můžeme si všimnout, že matice I je vlastně lineární kombinací matic $u_i v_i^*$, přičemž hodnoty σ_i představují koeficienty



Obrázek 5.14: Porovnání hodnot prvních 200 singulárních čísel pro ostrý a rozmazaný snímek.

této lineární kombinace. Označme si nyní součet pouze prvních k členů jako I_k, tj:

$$I_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*.$$

Matici I_k lze považovat za aproximaci vstupní matice I, přičemž parametr k určuje míru aproximace (bude jistě platit $I_0 = 0$ a $I_n = I$). Tohoto principu se velmi často využívá při kompresi dat, ovšem v našem případě jej použijeme právě k detekci ostrých částí snímku. V takovém případě nám postačí pouze analýza singulárních čísel, neboť lze předpokládat, že pokud je hodnota singulárního čísla σ_k zanedbatelná (vzhledem k prvnímu singulárnímu číslu), je v celkovém snímku zanedbatelný celý součin $\sigma_k u_k v_k^*$. Díky uspořádatelnosti singulárních čísel lze navíc vypustit i všechny následující členy, neboť jejich hodnoty budou také málo významné.

Předpokládejme nyní, že matice I odpovídá hodnotám jasu analyzovaného snímku. Pro ostrý snímek bude jistě zapotřebí použít vyšší stupeň aproximace, než v případě, kdy snímek bude rozmazaný. Pro názornost ukazuje obrázek 5.14 průběh hodnot singulárních čísel pro rozmazaný a ostrý snímek. Jako metriku ostrosti snímku navrhují v práci [18] použít podíl:

$$\beta_1 = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{n} \sigma_i},$$

jehož účelem je vyjádřit poměr k nejvýznamnějších a všech singulárních čísel. Tato metrika je sice pro některé snímky velmi účinná, ale ukazuje se, že není dostatečně

obecná a výsledky jsou velmi rozdílné. Hlavním problémem je totiž volba parametru k. Volba pevného k pro všechny typy a rozlišení snímků není vhodná, proto navrhneme jinou metriku. Hodnotu k zvolíme jako index takového singulárního čísla, které dosahuje m-procentní hodnoty prvního, tj. nejvýznamnějšího singulárního čísla. Metriku ostrosti poté definujeme jako podíl:

$$\beta_2 = \frac{k}{n},$$

kde n odpovídá počtu všech singulárních čísel. Tato metrika se jeví jako univerzálnější, neboť parametr k je volen podle prvního singulárního čísla a tudíž je nepřímo závislý na zpracovávaném snímku. Hodnota β_2 je z intervalu $(0, 1\rangle$, přičemž 0 odpovídá úplně rozmazanému snímku (nejsou potřeba žádná data) a 1 ostrému snímku (jsou potřeba všechna data singulárního rozkladu). Tyto hodnoty jsou však pouze hraniční a ve skutečnosti se k nim nikdy nepřiblížíme.

Uvedeným způsobem lze však dle ostrosti ohodnotit pouze celý snímek. Abychom zjistili, které části na snímku jsou ostré, rozdělíme zpracovávaný snímek do bloků a na každý blok aplikujeme tento postup. K tomuto dělení lze použít dva odlišné přístupy, snímek lze rozdělit buď na bloky pevné velikosti, nebo na pevný počet bloků. Pokud použijeme první možnost, riskujeme, že daný blok nezachytí důležité části snímku, neboť velikost bloku nebude úměrná rozlišení snímku. Může se stát, že pro menší snímky bude blok příliš velký, zatímco pro velké snímky bude zase příliš malý. Použijeme tedy dělení na pevně stanovený počet bloků, tj. snímek rozdělíme horizontálně na f dílů a každý díl rozdělíme vertikálně na stejný počet dílů. Získáme tím $f \times f$ bloků. Každý blok pak ohodnotíme reálným číslem z intervalu $\langle 0, 1\rangle$, které je úměrné ostrosti bloku, čímž získáme $f \times f$ hodnot. Abychom zajistili plynulejší přechody mezi koeficienty jednotlivých bloků, provedeme jejich filtraci. K tomu využijeme konvoluční filtr, který jsme uvedli v podkapitole 2.2.3. Nyní nám zbývá určit, které části snímku jsou opravdu ostré. Abychom mohli případnou ostrou část anotovat s větší přesností než jsou rozměry jednoho bloku, dopočteme koeficienty ostrosti na rozměr snímku pomocí bilineární interpolace. Tímto způsobem ohodnotíme každý bod snímku. Označme výsledek interpolace symbolem M. K rozdělení matice M na ostré a neostré části pak použijeme segmentační metodu Otsu, uvedenou v podkapitole 2.2.4. Pomocí ní obdržíme práh t, odpovídající maximální separovatelnosti ostrých a neostrých částí snímku. Každý blok pak prohlásíme za ostrý, pokud bude splněna nerovnost $\beta_2 > t$. Pokud bude celý snímek rozmazaný, nemá cenu provádět dělení snímku na ostré a neostré části. Tento případ lze rozpoznat tak, že práh t nalezený metodou Otsu bude velmi malý (víme, že bude platit nerovnost $0 < t < \max M$, přičemž hodnoty matice M jsou sami o sobě malé). Pokud je tedy hodnota prahu t menší než parametr t_{otsu}, můžeme snímek prohlásit za rozmazaný, resp. konstatovat, že neobsahuje žádné ostré části.

5.5.2 Implementace

Za účelem přehlednější implementace navržených algoritmů jsme se rozhodli vytvořit dva samostatné zásuvné moduly – `plugin_key_detect`, `plugin_sharp_detect`.

První zmíněný modul slouží k anotaci low/high key snímků. K jejich klasifikaci jsme využili 50% kvantil (medián), přičemž pro parametry t_1 a t_2 jsme experimentálně zvolili hodnoty $t_1 = 0{,}12$, $t_2 = 0{,}82$. Pokud tedy vypočtený kvantil náleží do intervalu $\langle 0; 0{,}12\rangle$, snímek je klasifikován jako low key, resp. pokud je prvkem intervalu $\langle 0{,}82; 1\rangle$, snímek je klasifikován jako high key. Jestliže snímek nespadá ani do jednoho z uvedených intervalů, není mu přiřazena žádná anotace. Klasifikovat snímek jako normální je nadbytečné.

Druhý vytvořený modul `plugin_key_detect` slouží k detekci ostrých částí snímků a při implementaci jsme zvolili následující parametry. Snímek je vertikálně i horizontálně rozdělen na 20 dílů. Hodnota byla zvolena proto, aby plocha jednoho dílu odpovídala 0,25 % celkové plochy snímku. Experimentálně se ukazuje, že uvedené dělení odpovídá dobrému kompromisu mezi vizuálním rozlišením detailů snímku a výpočetní náročností algoritmu. Drobnou nevýhodou se může jevit vyšší výpočetní náročnost singulárního rozkladu pro větší rozměr bloku. Pro výpočet singulárního rozkladu matice jsme využili knihovnu alglib (vizte odstavec 4.1). Metrika β_2 je vyhodnocována dle indexu takového singulárního čísla, jehož úroveň je 1% vzhledem k prvnímu singulárnímu číslu. Práh t_{otsu} pro metodu Otsu byl zvolen 8 %, přičemž klasifikovány jsou pouze ty oblasti jejichž velikost je vyšší než 4 díly, tedy 1 % celkové plochy snímku.

5.5.3 Výsledky

K otestování navrženého algoritmu pro klasifikaci snímků typu low a high key jsme použili 400 snímků získaných převážně z Internetu. Každý snímek byl vizuálně klasifikován do jedné ze tří kategorií: low key, high key a normální snímek. Poznamenejme, že normální snímky byly ve většině případu pořízené za denního světla. Výsledky získané tímto experimentem zobrazuje tabulka 5.6. Jak můžeme vidět, úspěšnost detekce low a high key snímků je skoro 90 % a navržená metrika se dá považovat za dostatečnou.

V případě, že bychom chtěli dosáhnout vyšší přesnosti detekce, by bylo možné algoritmus dále rozšířit použitím více než jednoho kvantilu. Každému snímku by se tím přiřadilo více příznaků (příznakový vektor), díky čemuž by bylo možné zpřesnit klasifikační fázi. Za účelem klasifikace bychom následně mohli použít např. Bayesův klasifikátor.

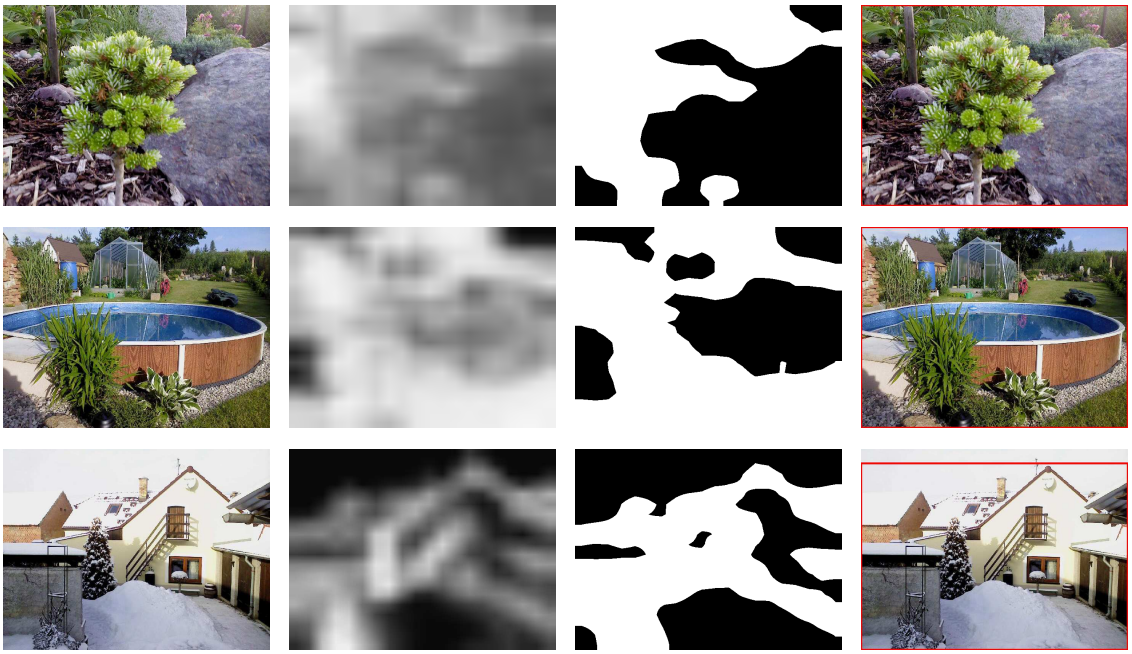
		low key	normální	high key
	celkem	77	257	73
detekováno jako	low key	69	2	0
	normální	8	255	9
	high key	0	0	64
absolutní počet	správně detekováno	69	255	64
	chybně detekováno	8	2	9
relativní počet	správně detekováno	89,61 %	99,22 %	87,67 %
	chybně detekováno	10,39 %	0,78 %	12,33 %

Tabulka 5.6: Výsledky anotace low a high key snímků.

Algoritmus pro hledání ostrých částí snímku byl testován na 300 snímcích získaných také z Internetu, abychom předcházeli závislosti na záznamovém zařízení. K otestování nezávislosti algoritmu na rozměrech vstupních fotografií jsme zvolili snímky, jejichž rozlišení se pohybovalo od 0,22 MPx do 18 Mpx. Výsledky testování ukazují, že navržený algoritmus velmi dobře funguje na snímcích, které obsahují ostrý objekt a rozmazané pozadí, a hranice mezi ostrými a neostrými částmi snímku je tedy dobře patrná. Příklad takových fotografií znázorňuje obrázek 5.15. V případě, že snímek je celý ostrý nebo hranice mezi ostrými a rozmazanými částmi snímku není tak výrazná, jsou výsledky nejednoznačné. Příklad vybraných fotografií ukazuje obrázek 5.16.

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

Obrázek 5.15: Příklady detekce ostrých částí snímku. Na snímcích se nachází jeden ostrý objekt na neostrém pozadí. Ve sloupci (a) je originální snímek, v (b) výsledky interpolace, v (c) výsledky Otsu prahování a v (d) vytvořené anotace. Snímky jsou získány (bráno od shora) ze zdrojů IX, X, XI, XII, XIII dle tabulky B.1.



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

Obrázek 5.16: Příklady detekce ostrých částí snímku. Uvedené fotografie neobsahují zřetelnou hranici mezi ostrými a neostrými částmi. Význam jednotlivých sloupců je opět shodný s předchozím obrázkem. Všechny snímky jsou získány ze zdroje II podle tabulky B.1.

6 Závěr

Práce se věnuje problematice automatické anotace digitálních snímků. Předpokládá se, že mnoho fotografií je pořizováno ve venkovním prostředí a zachycuje tak přírodu. Další velkou část tvoří společenské fotografie, na kterých se nacházejí především lidé. Proto jsme si v této práci vybrali anotaci lidského obličeje, zelené vegetace, modré oblohy a speciálních snímků.

V rámci práce byl proveden návrh a implementace algoritmů pro anotaci vybraných objektů. Detekce objektů je založena na principu pravděpodobnostního ohodnocení snímku. To se zdá velmi výhodné, neboť hledané objekty mají velmi podobné nebo dokonce stejné hodnoty. Segmentace snímku podle ohodnocení je tak významně snazší než v případě, kdy se provádí na základě jasu nebo barvy objektu. Problém segmentace se však přenáší na problém vhodného ohodnocení. Objekty, které jsou charakterisitcké spíše svým tvarem než barvou, není možné snadno ohodnotit.

K segmentaci pravděpodobnostního ohodnocení byla často používána metoda Otsu. Výsledky segmentace byly sice většinou velmi dobré, přesto se některé snímky nedařilo kvalitně segmentovat. Metoda Otsu se snaží o maximální separaci dvou tříd, přičemž v každé třídě předpokládá normální rozdělení pravděpodobnosti. Tento předpoklad však nemusí být vždy splněn. V další práci by bylo zajímavé vyzkoušet jiné metody automatického prahování. Také se ukazuje, že výsledky některých algoritmů jsou velmi závislé na vstupním snímku. Použitým předzpracováním se podařilo tuto závislost zmenšit pouze částečně. V další práci by bylo užitečné otestovat kvalitnější metody vyvážení barev a jiné sofistikované metody předzpracování snímků.

Dále byl implementován přenositelný program, který provádí automatickou anotaci digitálních snímků. Program je implementován jako knihovna, aby jej bylo možné používat v ostatních projektech. Knihovna je navržena ve stylu pluginové architektury a je ji tedy možné kdykoli rozšířit přidáním nových modulů pro anotaci dalších objektů. Výsledky generované vytvořeným programem neobsahují sémantické vazby mezi objekty, aby nedocházelo k zanášení zbytečných chyb do výsledků. Software lze však rozšířit i o sémantický analyzátor.

Jednotlivé algoritmy byly otestovány vždy na více než 250 fotografiích pocházejících z mnoha rozdílných zdrojů. Každý algoritmus byl testován samostatně, dosažené výsledky jsou podrobněji popsány u jednotlivých modulů. Z uvedených výsledků je patrné, že navržené algoritmy dosahují na zvolených fotografiích relativně vysoké úspěšnosti. Práci tímto považuji za úspěšně dokončenou, neboť byly dosaženy všechny stanovené cíle.

Literatura

[1] ANCHURI, Aditya. Image blur metrics. Stanford University, 2011.

[2] CAI, Junxia; GOSHTASBY, A. Detecting human faces in color images. *Image and Vision Computing*, 1999, 18.1: 63-75.

[3] GALLAGHER, Andrew C.; LUO, Jiebo; HAO, Wei. Improved blue sky detection using polynomial model fit. In: *Image Processing, 2004. ICIP'04. 2004 International Conference on.* IEEE, 2004. p. 2367-2370.

[4] HOFHANS, Tomáš. *Systém pro uložení grafické informace.* Plzeň, 2010. Diplomová práce. Západočeská univerzita, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky.

[5] HUANG, Gary B., et al. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In: *Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition.* 2008.

[6] CHO, Taeg Sang, et al. Image restoration by matching gradient distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2012, 34.4: 683-694.

[7] JIN, Zhong, et al. Face detection using template matching and skin-color information. *Neurocomputing*, 2007, 70.4: 794-800.

[8] LAUNGRUNGTHIP, Nuchjira et al. Sky detection in images for solar exposure prediction. 2008, p. 78-83.

[9] LIM, Suk Hwan; YEN, Jonathan; WU, Peng. Detection of out-of-focus digital photographs. *Hewlett-Packard Laboratories Technical Report HPL*, 2005, 14: 2005.

[10] LIMARE, Nicolas; Jose-Luis Lisani, Jean-Michel Morel, Ana Belén Petro, and Catalina Sbert, Simplest Color Balance, Image Processing On Line, 2011. `http://dx.doi.org/10.5201/ipol.2011.llmps-scb`

[11] LIU, Renting; LI, Zhaorong; JIA, Jiaya. Image partial blur detection and classification. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008. p. 1-8.

[12] OTSU, Nobuyuki. A threshold selection method from gray-level histograms. *Automatica*, 1975, 11.285-296: 23-27.

[13] PONZER, Martin. Face detection and recognition. Brno, 2009.

[14] PRAKASH, J.; RAJESH, K. Human face detection and segmentation using eigenvalues of covariance matrix, Hough transform and raster scan algorithms. *Proc. of Word academy of science, engineering and technology*, 2008, 29: 372-380.

[15] PŘINOSIL, Jiří; KROLIKOWSKI, Bc Martin. Využití detektoru Viola-Jones pro lokalizaci obličeje a očí v barevných obrazech. *Elektrorevue-Internetový časopis (`http://www.elektrorevue.cz`)*, 2008, 1-16.

[16] ROOMS, Filip; PIZURICA, Aleksandra; PHILIPS, Wilfried. Estimating image blur in the wavelet domain. In: *IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING.* IEEE; 1999, 2002. p. 4190-4190.

[17] SCHMITT, Frank; PRIESE, Lutz. Sky detection in CSC-segmented color images. *VISAPP (2)*, 2009, 101-106.

[18] SU, Bolan; LU, Shijian; TAN, Chew Lim. Blurred image region detection and classification. In: *Proceedings of the 19th ACM international conference on Multimedia.* ACM, 2011. p. 1397-1400.

[19] SZELISKI, Richard. *Computer vision: algorithms and applications.* Springer, 2010.

[20] TONG, Hanghang, et al. Blur detection for digital images using wavelet transform. In: *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on.* IEEE, 2004. p. 17-20.

[21] VILAPLANA, Veronica, et al. Object detection and segmentation on a hierarchical region-based image representation. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on.* IEEE, 2010. p. 3933-3936.

[22] VIOLA, Paul; JONES, Michael J. Robust real-time face detection. *International journal of computer vision*, 2004, 57.2: 137-154.

[23] YANG, Ming-Hsuan; AHUJA, Narendra. Detecting human faces in color images. In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on.* IEEE, 1998. p. 127-130.

[24] ZAFARIFAR, Bahman, et al. Grass detection for picture quality enhancement of TV video. In: *Advanced Concepts for Intelligent Vision Systems. Springer Berlin Heidelberg*, 2007. p. 687-698.

[25] ZAFARIFAR, Bahman; PETER, H. N. Adaptive modeling of sky for video processing and coding applications. In: *in 27 th Symposium on Information Theory in the Benelux.* June 2006, p. 31-38.

[26] ZARIT, Benjamin D.; SUPER, Boaz J.; QUEK, Francis KH. Comparison of five color models in skin pixel classification. In: *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on.* IEEE, 1999. p. 58-63.

[27] ZHOU, Linna, et al. Blur detection of digital forgery using mathematical morphology. In: *Agent and Multi-Agent Systems: Technologies and Applications.* Springer Berlin Heidelberg, 2007. p. 990-998.

Přílohy

A Uživatelská dokumentace

A.1 Přiložené DVD

Součástí práce je DVD, které obsahuje tento dokument v elektronické podobě (včetně všech uvedených obrázků) a implementovaný software. Ten je umístěn v adresáři `software`, jehož struktura je následující:

```
software
  - project
    - image_annotation_app
    - image_annotation_sdk
    - image_annotation_tool
    - plugin_sharp_detect
      - alglib
    - plugin_face_detect
    - plugin_vegetation_detect
    - plugin_key_detect
    - plugin_sky_detect

  - documentation
    - html
    - pdf
```

Adresář `project` obsahuje zdrojové kódy aplikace, SDK, hlavního modulu a všech zásuvných modulů. Adresář dále obsahuje soubor Qt projektu `image_annotation.pro`, pomocí kterého lze aplikaci přeložit, a to buď pomocí příkazů `qmake` a `make` anebo pomocí vývojového prostředí `Qt Creator`, které je distribuováno spolu s projektem Qt. Poznamenejme, že k úspěšnému překladu aplikace je třeba mít správně nainstalovanou knihovnu OpenCV, která je v našem projektu používána. Knihovna alglib je součástí DVD a je umístěna ve stejnojmeném adresáři.

Adresář `documentation` obsahuje dokumentaci navrženého softwaru ve formátu HTML a PDF. Ta byla vytvořena pomocí nástroje `doxygen`.

A.2 Spuštění aplikace

Předpokládejme adresář `fotografie`, který bude obsahovat jednu fotografii s názvem `snimek.jpg`. Automatickou anotaci spustíme příkazem:

```
image_annotation_app -i snimek.jpg -e,
```

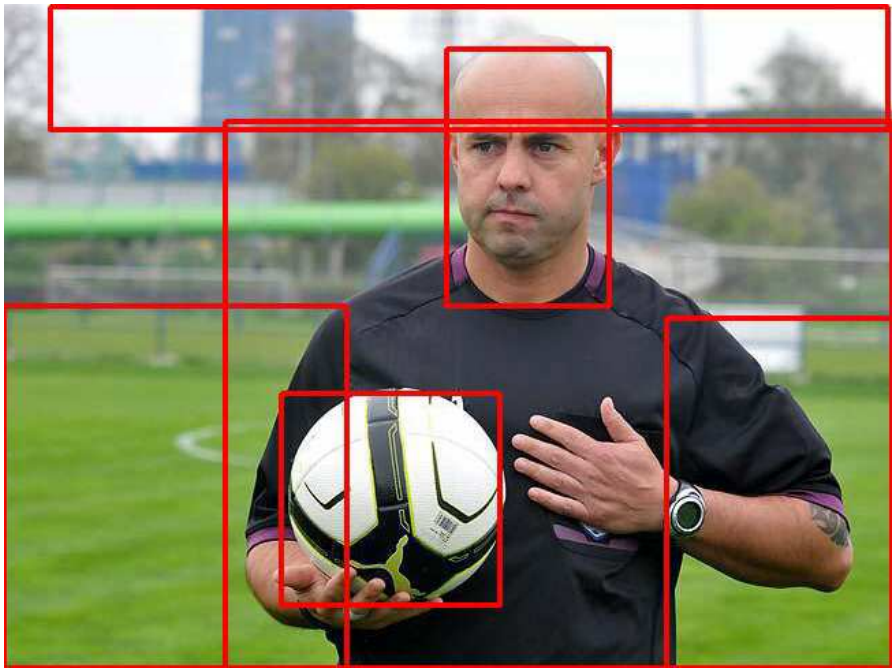
resp.

```
image_annotation_app -i fotografie -e,
```

čímž obdržíme výstupní soubory `snimek.xml` a `snimek_explain.jpg`. Pro snímek z obrázku A.1a bude výstupní snímek vypadat tak, jak ukazuje obrázek A.1b a výstupní XML soubor bude mít následující podobu:

```
<ImageAnnotationTool
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="ImageAnnotationTool.xsd">
  <TimeStamp date="2013-05-02" time="15:14:26"/>
  <AnnotationCodeTable>
    <Annotation code="1" value="PluginSharpDetect::SharpRegion"/>
    <Annotation code="2" value="PluginFaceDetect::Face"/>
    <Annotation code="3" value="PluginVegetationDetect::Vegetation"/>
    <Annotation code="4" value="PluginKeyDetect::LowKey"/>
    <Annotation code="5" value="PluginKeyDetect::HighKey"/>
    <Annotation code="6" value="PluginSkyDetect::Sky"/>
  </AnnotationCodeTable>
  <Image file="snimek.jpg" path="/fotografie"/>
  <Result>
    <Annotation code="1" position="[162, 86, 488, 400]" saturation="58"/>
    <Annotation code="2" position="[324, 33, 119, 188]" saturation="77"/>
    <Annotation code="3" position="[0, 221, 251, 265]" saturation="41"/>
    <Annotation code="3" position="[203, 285, 160, 155]" saturation="44"/>
    <Annotation code="3" position="[485, 230, 165, 256]" saturation="42"/>
    <Annotation code="6" position="[34, 2, 613, 90]" saturation="32"/>
  </Result>
</ImageAnnotationTool>
```

(a) Původní snímek (snímek je získán ze zdroje I dle tabulky B.1)



(b) Anotovaný snímek

Obrázek A.1: Ukázka výstupu vytvořeného softwaru.

B Použité zdroje fotografií

číslo	zdroj
I	autor: FC Vikotria Plzeň, získáno se souhlasem autora z: http://fcviktoria.cz
II	soukromá fotogalerie, získáno se souhlasem autora
III	autor: neznámý, získáno z: http://www.thewordstyler.com/blue-satin-man.html
IV	autor: neznámý, získáno z: http://www.wallpapershd.biz/wallpaper/spring-desktop-wallpaper-0911085
V	autor: neznámý, získáno z: http://wallpapertube.com/photography/road-and-forest-background
VI	autor: neznámý, získáno z: http://www.todayswall.com/wallpaper/photography/humpback-whale
VII	autor: neznámý, získáno z: http://www.flickr.com/photos/johnandreas/6333095160/in/pool-ir
VIII	autor: neznámý, získáno z: http://obrazky.4ever.sk/stavby/historicke/hrad-132007
IX	autor: neznámý, získáno z: http://vineethsreenivasan.wordpress.com/2010/02/27/focus-on-the-out-of-focus
X	autor: neznámý, získáno z: http://www.flickr.com/photos/gordaen/4044468356/in/photostream
XI	autor: neznámý, získáno z: http://www.antpitta.com/images/photos/furnariids/gallery_furnariids2.htm
XII	autor: neznámý, získáno z: http://www.cultorweb.com/sharp/TS.html
XIII	autor: neznámý, získáno z: http://www.3megapixel.it/images/parrot-grande.jpg
XIV	autor: neznámý, získáno z: http://topicworld.net/low-key-messing-around-1813183.htm
XV	autor: neznámý, získáno z: http://www.turistika.cz/mista/resort-pivon
XVI	autor: neznámý, získáno z: http://1.bp.blogspot.com/-OtBartJKWMs/TeTf3JRHRWI/AAAAAAAAA2E/IXPaP3ckIeg/s1600/Sky+Pictures.jpg

Tabulka B.1: Použité zdroje fotografií. Snímky obsažené v celé práci jsou získány z uvedených zdrojů dne 1.5. 2013.