VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PODOBNOST OBRAZU NA ZÁKLADĚ BARVY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

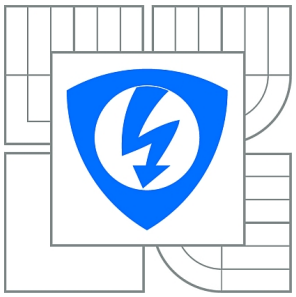
AUTOR PRÁCE
AUTHOR

Bc. FILIP HAMPL

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

PODOBNOST OBRAZU NA ZÁKLADĚ BARVY

IMAGE SIMILARITY BASED ON COLOUR

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

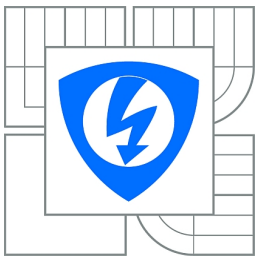
AUTOR PRÁCE — Bc. FILIP HAMPL

AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV UHER

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Filip Hampl	***ID:***	140227
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Podobnost obrazu na základě barvy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je nastudovat si současný stav vědy v oblasti porovnávání obrazu na základě barvy. Následně vybrané algoritmy implementovat v jazyce Java a ověřit jejich funkčnost na databázi fotek.

DOPORUČENÁ LITERATURA:

[1] PATERNAIN, Daniel, et al. A new algorithm for color image comparison based on similarity measures. In: EUSFLAT Conf. 2013.
[2] SU, Ching-Hung; CHIU, Huang-Sen; HSIEH, Tsai-Ming. An efficient image retrieval based on HSV color space. In: Electrical and Control Engineering (ICECE), 2011 International Conference on. IEEE, 2011. p. 5746-5749.
[3] VAVILIN, Andrey; JO, Kang-Hyun. Automatic context analysis for image classification and retrieval based on optimal feature subset selection. Neurocomputing, 2013, 116: 201-207.

Termín zadání:	9.2.2015	***Termín odevzdání:***	26.5.2015

Vedoucí práce: Ing. Václav Uher
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce pojednává o podobnosti v obraze na základě barvy. Jsou zde diskutovány potřebné teoretické základy pro pochopení dané tématiky. Těmito základy jsou barevné modely, které jsou v této práci implementovány, princip tvorby histogramu a jeho porovnávání. Dalším tématem je shrnutí současného pokroku v oblasti porovnávání obrazů a přehled několika používaných metod. V praktické části práce je představena trénovací databáze obrazů, na které jsou získávány výsledky úspěšnosti vytvořených metod. Tyto metody jsou jednotlivě popsány, včetně jejich principu a dosažených výsledků. V samotném závěru práce je popsáno vytvořené uživatelské rozhraní, které poskytuje přehlednou prezentaci výsledků pro zvolenou metodu.

KLÍČOVÁ SLOVA

Podobnost obrazu, barevný model, histogram, barevná hloubka, interpolace

ABSTRACT

This diploma thesis deals with image similarity based on colour. There are discussed necessary theoretical basis for better understanding of this topic. These basis are color models, that are implemented in work, principle of creating the histogram and its comparing. Next chapter deals with summary of recent progress in the field of image comparison and overview of several most used methods. Practical part introduces training image database, which gives results of success for each created method. These methods are separately described, including their principles and achieved results. In the very end of this work, user interface is described. This interface provides a transparent presentation of the results for the chosen method.

KEYWORDS

Image similarity, color model, histogram, color depth, interpolation

HAMPL, Filip *Podobnost v obraze na základě barvy*: diplomová práce. Místo: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, Rok. 51 s. Vedoucí práce byl Ing. Václav Uher,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Podobnost v obraze na základě barvy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Místo ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Václavu Uhrovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Místo ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Podobnost obrazu na základě barvy spočívá v rozřazení množiny obrazů na základě jejich barevných vlastností do odpovídajících kategorií. Klasifikace scény představuje poměrně jednoduchý úkol pro lidské vnímání, nicméně toto hodnocení je velice subjektivní a může být výrazně ovlivněno schopnostmi a momentálním rozpoložením hodnotícího jedince.

Naproti tomu klasifikace scény pomocí počítače vrací pro stejné vstupní obrazy vždy stejné hodnocení a eliminuje tak nežádoucí subjektivnost. Při posuzování scény pomocí počítače jsou brány v potaz parametry jako barva, tvar, textura a rozložení prvků. Tato klasifikace ovšem může narážet na problém, kterým je nerozpoznání dominantního prvku obrazu a následné chybné zařazení. Díky této skutečnosti se pro zpřesnění výsledků využívá genetických algoritmů nebo zásahu člověka.

V této práci jsou rozebrány nejdůležitější teoretické poznatky potřebné k pochopení principu určování podobnosti na základě barvy. Jsou zde diskutovány výhody, stejně tak jak nevýhody systémů využívajících slovního hodnocení pro získání podobných obrazů. Větší část je věnována systémům Content Based Image Retrieval (CBIR), které pro určování podobnosti využívají obsah obrazu, tedy jeho nízkoúrovňové příznaky. Tyto systémy jsou zde obecně zhodnoceny a stručně diskutovány jejich nejběžnější modely. Další částí práce je barevný model HSB, který se pro svoje vlastnosti a podobnost s lidským vnímáním barev často využívá v CBIR systémech. Je zde také vysvětlen princip Fuzzy množin, které jsou díky svému pojetí náležitosti k dané množině také často využívány v systémech pro hledání podobných obrazů. Další teoretickou částí jsou histogramy, u kterých je vysvětlen princip jejich vzniku a výhody tohoto vyjádření obrazu. V posledních kapitolách teoretické části práce jsou rozebrány interpolační metody a problematika barevné hloubky.

V praktické části je představena trénovací databáze fotografií a jsou zde popsány její kategorie a další vlastnosti. Velká pozornost je věnována metodám porovnávání histogramů, která jsou ústřední pro tuto diplomovou práci. Dále jsou popsány implementace vytvořených metod využívajících pro porovnávání rozdílné barevné modely a pomocné algoritmy. U každé této části lze nalézt podrobný postup včetně ukázek kódu jazyka Java, který byl pro implementaci metod použit. V samotném závěru práce jsou uvedeny procentuální úspěšnosti vytvořených metod.

1 PODOBNOST OBRAZU

Díky obrovskému rozmachu digitálních fotografií a jejich databázím vznikla potřeba efektivně vyhledávat v tomto velkém množství fotografie dle požadovaného klíče. Tento problém lze řešit textovou indexací každého obrazu. Takto jsou řešeny například mnohé nemocniční systémy, kde je k jednotlivým snímkům přiřazeno identifikační číslo pacienta a následně i další nezbytné informace, potřebné k efektivnímu vyhledávání těchto snímků. Systémy založené na tomto principu se souhrnně nazývají TBIR (Text-Based Image Retrieval). Problémem této metody je fakt, že počet obrazů rapidně narůstá a je tedy nereálné opatřit všechny potřebným hodnocením a vhodným popisem. Dalším problémem této metody je subjektivnost hodnocení scény, která může představovat přiřazení obrazu do jiné kategorie, než by byla zvolena jiným hodnotícím. Tuto skutečnost díky lidské individualitě vnímání nelze nikdy zcela eliminovat a zaručit tak objektivní hodnocení obrazu.

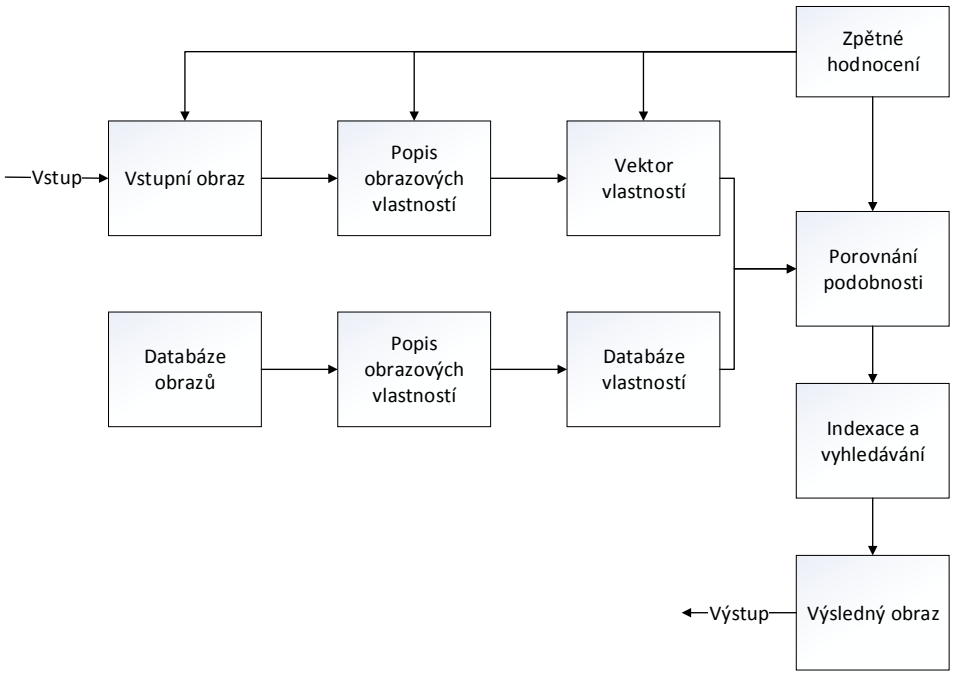
1.1 CBIR systémy

Systémy, které automaticky hodnotí obrazy na základě jejich obsahu, se nazývají CBIR (Content-Based Image Retrieval). Zájem o tyto systémy výrazně vzrostl společně s příchodem výkonnějších procesorů, pamětí a také díky jejich klesajícím cenám.

Tyto systémy fungují na principu rozřazení databáze obrazů do kategorie relevantních, či irelevantních vůči vstupnímu obrazu. Při vložení nového obrazu do databáze se pomocí příslušného algoritmu extrahují jeho obrazové vlastnosti, které jsou poté uloženy do vektoru, díky němuž je případné vyhledávání obrazů podobných dotazovému mnohem rychlejší.

Tyto systémy se potýkají se dvěma hlavními problémy. Prvním z nich je fakt, že počet vhodně označených obrazů, které mohou sloužit jako trénovací množina pro daný algoritmus, je většinou velmi omezený. Díky této skutečnosti je bohužel výrazně snižována výsledná klasifikační přesnost systému. Druhým problémem je dimenze dat určených k učení těchto algoritmů. Při obrovském množství těchto dat se nám objevují praktické problémy spojené s časovou náročností, zvolením vhodných obrazových vlastností a odfiltrováním irelevantních dat.

Obecně lze říci, že pro klasifikaci obrazu tyto systémy v různé míře využívají atributy, jakými jsou barva, textura, objekty a prostorové rozložení. Váha a způsob, kterým jsou tyto atributy v jednotlivých systémech využívány se výrazně liší. Nicméně trendem poslední doby je kombinace těchto parametrů, což vede k účinnější a přesnější klasifikaci obrazů. Princip fungování systémů CBIR je zřetelně vidět v



Obr. 1.1: Blokový diagram CBIR systému.

diagramu 1.1. V následujícím textu jsou stručně rozebrány některé techniky a jejich hlavní principy dle [1].

1.1.1 Metoda podpůrných vektorů

Tento algoritmus se snaží najít optimální řešení pro rozdělení obrazů z databáze do relevantní nebo irelevantní kategorie. Další vlastností tohoto algoritmu je snaha o maximalizaci vzdálenosti těchto dvou kategorií. Počáteční úvaha zde předpokládá, že celá databáze je lineárně rozdělitelná do těchto dvou tříd. Tato metoda počítá i s neklasifikovanými obrazy, které jsou na základě jejich relevantnosti přiřazeny do odpovídající třídy.[2]

1.1.2 Metody zpětné vazby

Protože většina obrazových databází se skládá z nepopsaných obrazů, je často potřeba pomoci učícím se algoritmům lidským hodnocením podobnosti. Jedním z hlavních cílů těchto algoritmů je postupně minimalizovat potřebu zásahu člověka do hodnotícího procesu. Hlavním principem je zjistit ideální vztahy mezi jednotlivými obrazovými atributy, které následně definují členství v dané kategorii. Pro zjištění

těchto vztahů se využívají obrazy, které obdrželi podobné hodnocení od člověka. Následně je vytvořena pravděpodobnostní funkce, která přiřazuje získané atributy určitým třídám. Tento algoritmus lze po dostatečném zásahu považovat za zcela samostatný a velmi účinný.[3]

1.1.3 Metoda aktivního učení

Metoda aktivního učení patří mezi další metody, které využívají hodnocení obrazu člověkem. Na rozdíl však od pasivního učení, kdy jsou k hodnocení předkládány obrazy zcela náhodně, využívá tato metoda systém, který dle zvoleného klíče předkládá ke klasifikaci předem určené obrazy a tím urychluje celkovou konvergenci systému k cíli. Celý tento proces funguje na základě úrovně pozitivního či negativního hodnocení podobnosti zadaného obrazu a obrazu z databáze. Díky tomuto uživatelskému hodnocení je následně k hodnocení předložen takový snímek, který by měl pomoci k rychlejšímu ukončení celého učícího se procesu. Tyto systémy tedy pracují s relevancí předložených obrazů, a to jak s pozitivní, tak i negativní.[4]

1.1.4 Bootstrapping SVM aktivní učení

Je zde využito aktivního SVM (Support Vector Machines) učení, po jehož ukončení jsou obrazy databáze ohodnoceny v sestupném pořadí na základě jejich podobnosti k vstupnímu obrazu. Vrchní obrazy jsou použity jako nejvíce podobné a tedy i výstupní. Důležitým předpokladem pro tuto metodu je prezence alespoň jednoho pozitivně a jednoho negativně hodnoceného obrazu. Následně je použit počáteční klasifikátor SVM, který je výrazně zpřesněn pomocí ještě neklasifikovaných obrazů. Díky využití těchto obrazů databáze odpadá potřeba klasifikovat další obrazy pomocí uživatele, což je v CBIR systémech vysoce žádoucí.[5]

1.1.5 Semantic Kernel Learning

Za sémantické jsou zde považovány veškeré vstupy uživatele, které jsou ukládány v průběhu všech iterací učícího se procesu. Pokud si tedy označíme Kernel matici po jednom průchodu t: $(K_{ij})_t = k((X_i)_t, (X_j)_t)$, můžeme hodnocení provedené v tomto kroku t uložit do vektoru y_t, kde na základě podobnosti 1 značí relevantní obraz, -1 irelevantní a 0 nehodnocený.[6] Pokud tedy tento systém použijeme při více průchodech, vznikne nám dobře značený a tím i přesný systém. Princip této matice je ukázán v tabulce 1.1.

	Y_1	Y_2	Y_3	...
X_1	1	0	-1	...
X_2	-1	1	0	...
X_3	0	-1	1	...
...	...	...	...	

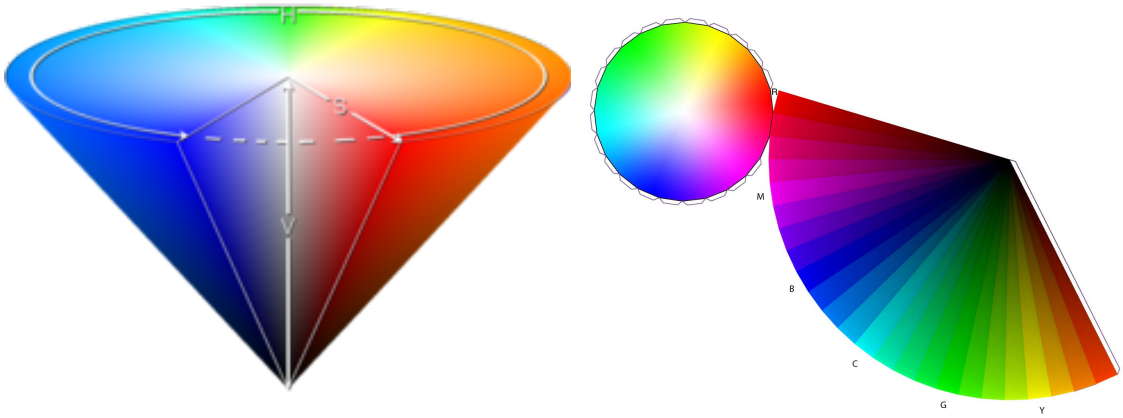
Tab. 1.1: Podoba kernel matice

1.2 Barevný model HSB

Barevný model reprezentovaný kanály Hue, Saturation, Brightness (HSB) představuje pro člověka snadněji představitelný model, než-li například model RGB (kanály Red, Green a Blue) nebo CMY (Cyan, Magenta, Yellow). Tyto modely se vyznačují velkou přehledností, ovšem pro lidské vnímání je nepřirozené přiřadit k vnímané barvě podíl tří barevných složek RGB, které jsou v ní v různém množství zastoupeny.

Zkratka HSB zastupuje Hue - odstín barvy, což je hlavní spektrální složka, Saturation - sytost, neboli také „čistotu“ a živost barvy, Brightness - jas, nebo také světlost [7]. Tento barevný model bývá zobrazován jako kužel nebo šestiboký jehlan. Jeho hlavní předností je intuitivní zobrazení barevného spektra pro lidské vnímání. Pokud chceme získat určitou barvu z tohoto barevného modelu, postupuje se nejdříve volbou barevného odstínu H, dalším krokem je nasycení S a jako poslední světlost B. Toto rozložení je výhodné tím, že i když měníme hodnoty nasycení a světlosti, zamýšlený barevný odstín zůstává stejný. V tomto ohledu překonává barevné modely RGB i CMY. Pro lepší ilustraci je možné si tuto vlastnost představit jako kreslení vodovými barvami na bílý papír, kde nejdříve zvolíme barevný pigment a následně ovlivňujeme jeho světlost přidáváním černé nebo běloby. Nasycení výsledné barvy lze ovlivňovat přimícháváním další barvy.

Tento barevný model je ukázán na obrázku 1.2, kde lze vidět souvislost mezi jednotlivými složkami barvy. Tento model bývá někdy také označován jako HSV (V...value).



Obr. 1.2: Úplný barevný model HSB (vlevo) a jeho 2D zobrazení (vpravo, podstava a plášť. [7])

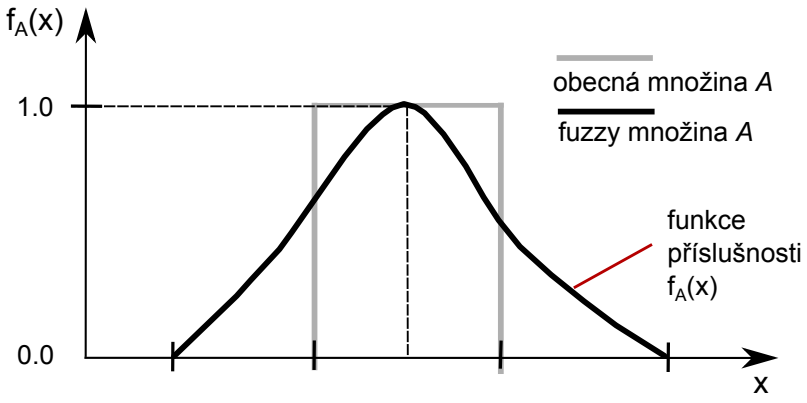
1.3 Fuzzy množiny

V reálném životě lidé často řeší otázku přiřazení různých objektů do odpovídajících tříd (množin). Některé tyto objekty však díky jejich povaze nelze zcela jednoznačně přiřadit do dané množiny. Důvodem je nejednoznačná definice některých těchto tříd, čímž se liší od definice množin ve smyslu, který známe z matematiky. Těmito třídami může být například množina čísel mnohem větších než je číslo 1. Pokud máme rozhodnout o členství například čísla deset v této množině, logicky vyvstává problém s jednoznačností tohoto zařazení. Dalším příkladem podobně nejednoznačné třídy může být množina krásných žen nebo vysokých mužů. I když tyto množiny nesplňují klasický charakter, hrají důležitou roli v každodenním životě lidí, jejich myšlení, sdělování informací, abstrakci a i při klasifikaci obrazů do patřičných skupin.

1.3.1 Definice

Mějme definovaný prostor prvků X a obecný prvek tohoto prostoru x, tedy $X = \{x\}$. Fuzzy množina A v X je charakterizována funkcí příslušnosti $f_A(x)$, která je vztažena ke každému prvku v X a nabývá reálné hodnoty z intervalu $[0, 1]$. Hodnota funkce $f_A(x)$ udává „stupeň členství“ prvku x v množině A. Čím blíže je tedy hodnota této funkce jednotce, tím vyšší stupeň členství.

Pro ilustraci si lze představit řadu čísel X, u kterých rozhodujeme o jejich členství ve fuzzy množině A, čísel mnohem větších než 1. Přesné, přesto subjektivní specifikace množiny A, lze dosáhnout určením funkce $f_A(x)$ na X. Tyto hodnoty tedy mohou vypadat například takto: $f_A(0) = 0$; $f_A(1) = 0$; $f_A(5) = 0,01$; $f_A(10) = 0,2$; $f_A(100) = 0,95$; $f_A(500) = 1$.[8]

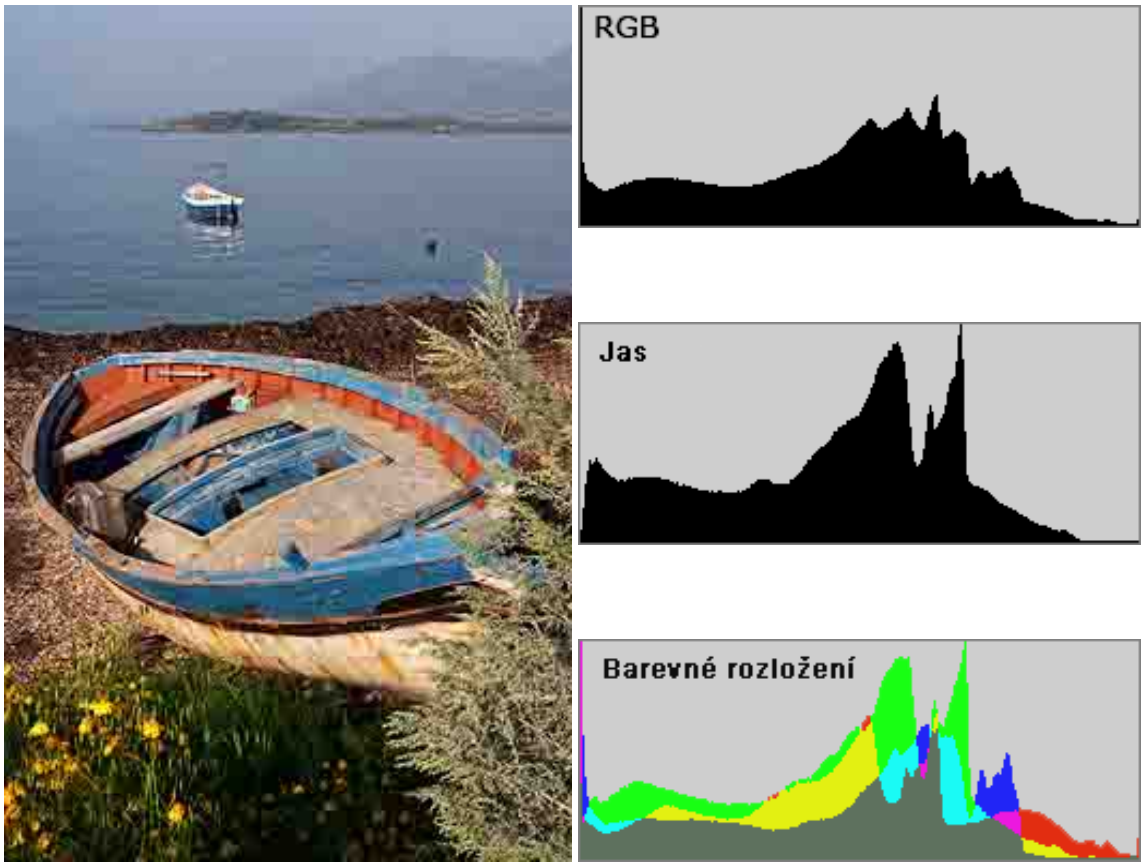


Obr. 1.3: Porovnání obecné množiny s fuzzy množinou.

Na obrázku 1.3 lze jasně vidět rozdíl mezi fuzzy množinou a obecnou množinou, která je charakterizována hodnotou 1 pro čísla náležící do této množiny nebo hodnotou 0 pro čísla mimo tuto množinu. Vlastností fuzzy množin lze s výhodou využívat při klasifikaci podobnosti obrazu, kdy díky funkci příslušnosti lze sestavit výsledné pořadí podobných obrazů a určit hranici podobnosti, která je ještě brána jako relevantní. Podobně lze zacházet také s jednotlivými atributy obrazu.

1.4 Histogram

Histogram je jedním ze sloupcových grafů, který zachycuje na vodorovné ose dané intervaly a na svislé ose četnost jejich výskytu ve vstupních datech. Barevný histogram tedy analogicky zobrazuje četnost zastoupení určité barvy v obraze. Barevný histogram lze zobrazit jak pro jednotlivé kanály barevného modelu, kterým je obraz reprezentován, tak i pro jas obrazu. Názorná ukázka obrazu a různých druhů jeho histogramů je ukázána na obrázku 1.4.

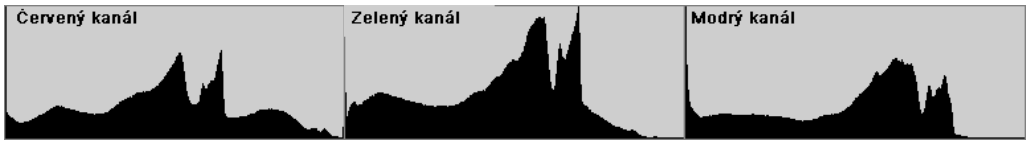


Obr. 1.4: Histogram RGB složek, jasu a složený histogram obrázku vlevo.[9]

1.4.1 Jasový histogram

Jasový histogram je mnohem vhodnějším prostředkem pro reprezentaci rozložení světla v obraze. Jasový histogram bere v potaz fakt, že lidské oko je více citlivé na zelenou složku světelného spektra, než na červenou a modrou. To je také hlavní důvod, proč se histogram zeleného kanálu nejvíce ze všech podobá jasovému, což je jasně patrné z obrázku 1.5.

Jasový histogram je vytvořen tím, že je nejdříve převeden každý pixel obrazu tak, aby reprezentoval jasovou složku. Ta je tvořena váženým průměrem všech tří



Obr. 1.5: Porovnání histogramů pro jednotlivé kanály barevného modelu RGB.[9]

barevných kanálů, ve kterém je zelená složka zastoupena z 59%, červenému kanálu odpovídá 30% a modrému zbylých 11%. Pokud tedy budeme počítat hodnotu jasu pixelu, který má hodnoty RGB kanálů 100R, 150G a 200B, musíme provést vynásobení příslušnými koeficienty a výsledný součet odpovídá hodnotě jasu pixelu, což je v tomto případě: $(100 \times 0,3) + (150 \times 0,59) + (200 \times 0,11) = 140$.[9]

Jasový histogram nám tedy přehledně poskytuje informaci o stavu osvětlení scény, což je jedním z klíčových faktorů profesionálního fotografování. I rychlým pohledem jde poté zjistit, jestli je rozložení jasu rovnoměrné, využívající celý rozsah čipu a zda se příliš neblížíme limitním hodnotám, kterými jsou čistá černá a bílá barva.

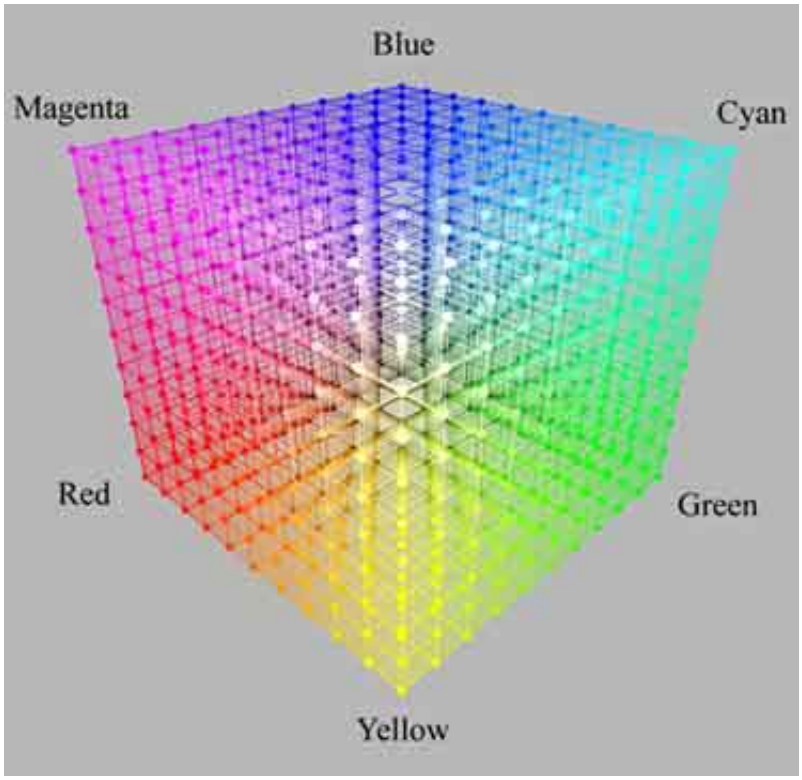
Obr. 1.6: Zobrazení jednotlivých barevných kanálů původního obrázku (uprostřed) ve stupních šedi. Červený kanál je levý horní obrázek, levý spodní zobrazuje modrý kanál, pravý horní odpovídá zelenému kanálu a pravý spodní je jasový.[9]

Histogram jednotlivých kanálů, který je ukázán na obrázku 1.5 nám dává informaci o četnosti zastoupení odstínů dané barvy. Dále z něj lze poměrně jednoduše vyčíst kvalitu expozice scény. Pokud je v histogramu vidět, že krajní hodnoty jsou zde výrazně zastoupeny, existuje velká pravděpodobnost toho, že je to způsobeno nedostatečným rozsahem použitého čipu, což vede k výrazné degradaci kresby (textury) v oblastech s těmito pixely. Tento jev se nazývá přeexponování scény a logicky jeho opakem je podexponování scény. Přeexponované hodnoty jsou obdrženy například při pořízení histogramu červeného kanálu obrázku 1.6. Celkový histogram RGB nám tedy může dát informaci o přeexponování, respektive podexponování scény, ale

již díky němu nezjistíme, zda je to zaviněno pouze jednou či více složkami. Všechny typy histogramů hrají důležitou roli v následné úpravě digitálních obrazů. Další využití histogramu je v CBIR systémech, které využívají odolnost histogramu proti transformacím obrazu jako je například rotace. Velkou nevýhodou ovšem je ignorování textury histogramem, což může způsobit, že systém vyhodnotí například obrázek červeného jablka v koruně stromu jako velmi podobný červenému automobilu parkujícímu na trávě.

1.5 Barevná hloubka

Barevnou hloubkou rozumíme počet bitů, který je určený pro reprezentaci barvy každého pixelu obrazu. Při vyšší hodnotě barevné hloubky je možné pixel vyjádřit rozdělením tohoto počtu mezi jednotlivé kanály barevného modelu, kterým je daný obraz vyjádřen. Pokud tedy pracujeme například s modelem RGB a barevnou hloubkou rovnou čtyřiadvaceti bitům, což je označováno jako „true color“, máme možnost vyjádřit 2^{24} tedy 16.777.216 barev. Těchto 24 bitů je rovnoměrně rozděleno mezi tři kanály barevného modelu RGB, tedy 8 bitů na každý kanál. Díky tomuto rozdělení máme možnost vyjádřit na každé ose krychle RGB 256 diskrétních hodnot jednotlivého kanálu. Pro lepší představu tohoto rozložení slouží obrázek 1.7, kde je každá osa rozdělena na 11 diskrétních hodnot, což dává celkový počet 1331 barevných možností.



Obr. 1.7: Diskrétní vyjádření RGB krychle. Převzato z [10].

Ve většině digitálních obrazů se setkáváme se situací, kdy je poměrně značná část barevných možností nevyužita a naopak více pixelů jednoho obrazu má totožnou hodnotu své barvy. Tento fakt byl hlavní motivací pro vytvoření mnoha algoritmů na kvantizaci barvy digitálního obrazu, díky které by bylo možné tento

obraz vyjádřit pomocí menšího počtu bitů, při minimální ztrátě zobrazované kvality. Tento problém je řešen vytvořením barevné palety, která obsahuje předem daný počet barev, odpovídající velikosti barevné hloubky. Jednotlivé pixely pak obsahují místo barevné informace pouze odkaz na pozici do této palety. Pokud tedy redukujeme barevnou hloubku na 8 bitů, je možné takto vyjádřit 256 barev.

Velkou nevýhodou tohoto přístupu je potřeba přenášet společně s obrazem i jeho barevnou paletu (tabulku). Tento princip při požadavku na vyšší počet barev obrazu již díky obrovské velikosti přidružené barevné palety ztrácí na efektivitě a využívá se opět přímého vyjádření barvy pomocí jednotlivých kanálů barevného modelu.

Za účelem vytvoření co nejlepší barevné palety bylo vymyšleno mnoho algoritmů, které se všechny snaží o minimalizaci rozdílu mezi barvami původního obrazu a obrazu výstupního. Samozřejmě se nabízí i možnost spočítat pro danou výstupní barevnou hloubku všechny možné variace barevných palet a z nich následně vybrat tu, která bude mít nejmenší hodnotu barevné odchylky od vstupního obrazu. Toto řešení ovšem vyžaduje enormní množství výpočetního času a je značně neefektivní. Pro účely této diplomové práce byl vybrán algoritmus median cut, kterému se věnuje následující kapitola.

1.5.1 Median cut

Tento algoritmus vytvořil Paul S. Heckbert roku 1979 v M.I.T (Massachusetts Institute of Technology). V době vzniku tohoto algoritmu existovaly poměrně značné technologické limity pro zpracování digitálních obrazů pomocí tehdejších procesorů a jejich následné uložení na kapacitně skromná paměťová média. Přesto tento algoritmus zůstává jedním z nejpoužívanějších i v dnešních systémech pro zpracování digitálního obrazu. Pro kvantizaci obrazu je potřeba vykonat tyto kroky:

- navzorkovat obraz, za účelem získání barevného rozložení pixelů
- na základě tohoto rozložení vybrat vhodnou barevnou paletu
- vytvořit barevnou tabulku, pomocí které je možné mapovat vybrané barvy na hodnoty jednotlivých pixelů
- přepočítat hodnoty všech pixelů na kvantizované.

Nejnáročnějším z těchto kroků je vybrat barevnou paletu, která bude mít nejmenší hodnotu vzdálenosti od původního barevného vyjádření obrazu. Další kroky jsou již poměrně snadné a jasné.[11]

Pro jednorozměrné, monochromatické obrazy, nejčastěji vyjádřeny ve stupních šedi, je při barevné hloubce 8 bitů možné kvantizovat v rozmezí $< 0, 255 >$ na požadovaných k výstupních barev. Konce nově vytvořených intervalů, sdružujících jednotlivé hodnoty, se nazývají rozhodovací úrovně. Při uniformní kvantizaci jsou tyto úrovně rovnoměrně rozmístěny po celém intervalu, naopak při neuniformní kvan-

tizaci jsou jednotlivé úrovně prodlužovány, respektive zkracovány v závislosti na pravděpodobnosti každé části daného intervalu.

Samotný algoritmus pracuje vždy s jednou rozhodovací úrovní a postupuje vždy zleva doprava (od nízkých k k větším). Zároveň jsou shromažďovány informace o kvalitě různého umístění $x[k]$ na intervalu od 0 do 254, kde se snažíme najít pozici $x[k-1]$, pro kterou je prozatímní úroveň odchylky nejmenší. Tuto odchylku můžeme spočítat jako: $\sum D[0] + D[1] + D[2] + ... + D[k-1]$. Další průchody tohoto algoritmu využívají ukládání těchto dílčích hodnot do tabulky, ve které je pro všechna k a jejich možné pozice uložena hodnota s minimální odchylkou, společně s pozicí $x[k-1]$, na které jí bylo dosaženo. Když je tedy dosaženo $k = K - 1$, kde K značí počet kvantizačních úrovní, je možné pomocí vytvořené tabulky zpětně získat hodnoty dalších mezních hodnot.

Pokud však přesuneme kvantizaci do dvoj nebo trojrozměrné dimenze, setkáváme se s mnohem složitější implementací. Je to způsobeno větší vzájemnou závislostí mezi sousedními intervaly, kdy jednotlivé buňky mohou nabývat tvaru polygonů nebo mnohostěnů. Mohou tedy nabývat tří a více společných rozhodovacích úrovní, a proto již nelze ke kvantizaci přistupovat stejně jako v jednodimenzionálním systému.

Samotný tvůrce tohoto algoritmu připodobňuje systém vypořádání se s kvantizací v trojrozměrném barevném prostoru k americkému senátu, kde je počet zástupců dané oblasti přímo úměrný její rozloze. Algoritmus tedy v každé iteraci najde minimální a maximální hodnoty červeného, zeleného a modrého kanálu pro každou z buněk. Nejvyšší hodnota rozdílu mezi tímto minimem a maximem určí v každé buňce dominantní dimenzi, dle které by se měla buňka dělit. Buňka, která má ze všech nejvyšší hodnotu tohoto rozdílu se následně rozdělí na půlky dle dominantní dimenze a algoritmus pokračuje další iterací, dokud není dosažen daný počet barev. Výsledná hodnota barevné buňky je dosažena zprůměrováním hodnot, které obsahuje. Tyto výsledné hodnoty jsou jednotlivými prvky barevné palety, dle které je daný obraz překreslen. Pro zhodnocení výsledků tohoto algoritmu následuje kolekce stejného vstupního obrazu s různými hodnotami barevné hloubky a tedy i počtem výstupních barev, viz 1.8. Původní obrázek má barevnou hloubku 24 bitů, další se pohybují v barevném rozmezí od dvou do 256ti barev.

Obr. 1.8: Ukázka výstupu implementovaného kvantizačního algoritmu. V levém horním rohu se nachází původní obraz s barevnou hloubkou 24 bitů, další obraz má barevnou paletu omezenou pouze na dvě barvy, levý prostřední má paletu s pěti barvami a vedlejší obraz disponuje deseti barvami. Ve spodní řadě vlevo je paleta omezena na šestnáct barev a pravý spodní obraz má 256 barev.

1.6 Interpolační metody

Interpolací rozumíme proces vytváření přibližných hodnot spojité funkce, u které jsou známy jen některé její diskrétní hodnoty. Interpolační algoritmy jsou poměrně často využívány při zpracování digitálních obrazů, u kterých se provádí změna jejich velikosti.[12] V následujících podkapitolách jsou přiblíženy tři tyto algoritmy, které jsou zároveň použity pro účely této diplomové práce.

1.6.1 Nejbližší soused

Tento interpolační algoritmus, který je anglicky označován jako Nearest neighbour, je jedním z nejjednodušších algoritmů pro nalezení mezilehlých hodnot na daném intervalu. K určení těchto hodnot používá pouze vzdálenost od známých hodnot prvků, respektive pixelů. Jakmile je tedy některý z těchto prvků vyhodnocen jako nejbližší od hledaného bodu, je jeho hodnota použita i pro hledaný bod, kterým může být pixel či libovolný prvek.

Platí zde principy zaokrouhlování a pokud nastane situace, kdy hledaný bod leží přesně v polovině vzdálenosti mezi dvěma známými hodnotami, je mu přiřazena hodnota toho prvku, který má vyšší hodnotu v odpovídající souřadnicové ose. Díky těmto vlastnostem je tento algoritmus poměrně jednoduchý na implementaci a je schopný zachovat ostré hrany, pokud jsou rovnoběžné se souřadnicovými osami. V ostatních případech bohužel vzniká poměrně silný aliasing. [13]

Obr. 1.9: Osmkrát zmenšený vstupní obraz pomocí algoritmu nejbližšího souseda. Vlevo bez aplikovaného vyhlazení výsledku průměrováním, vpravo s aplikovaným vyhlazením. Oba tyto obrázky jsou získány z totožného původního obrazu jako v 1.8.

V této diplomové práci byl implementován algoritmus, který využívá několik interpolačních metod, mezi které patří i metoda nejbližšího souseda. Tyto metody

jsou aplikovány pro účely zmenšení velikosti vstupních obrazů, ve které jsou následně porovnávány. Celý algoritmus bude probrán v praktické části této diplomové práce. Dílčí výstup z této metody s vstupním obrazem upraveným pomocí interpolace nejbližšího souseda je vidět na obrázku 1.9.

1.6.2 Bilineární interpolace

Tato metoda již zohledňuje pro vytvoření nové hodnoty větší počet okolních známých bodů. V případě dvojrozměrného obrazu to jsou čtyři okolní pixely, které váženým průměrem určí výslednou hodnotu nového pixelu. Základní myšlenkou je zde provedení lineární interpolace nejdříve v jednom směru souřadných os a až poté i na druhé souřadné ose. Pořadí těchto dílčích interpolací nemá na výsledek žádný vliv a je tedy na nás, v jaké ose se rozhodneme interpolovat jako první. Výsledná hodnota je získána váženým průměrem ze známých hodnot nejbližších pixelů, kdy o váze těchto známých hodnot na výslednou hodnotu rozhoduje poměr jejich vzdáleností od hledaného bodu.[14]

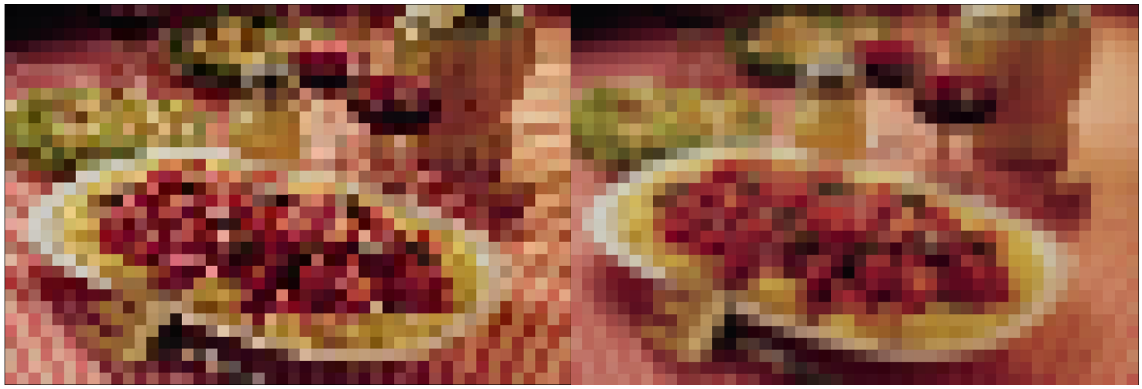
Pro lepší představu funkčnosti tohoto algoritmu je uveden obrázek 1.10, který pro stejný vstupní obraz jako v předchozích dvou případech představuje míru zásahu bilineární interpolace při osminásobném zmenšení vstupního obrazu.

Obr. 1.10: Výstupní obrazy po aplikování bilineární interpolace při osminásobném zmenšení vstupního obrazu. Vlevo bez použití průměrování výsledných hodnot, na pravý obrázek bylo použito průměrování výsledných hodnot.

1.6.3 Bikubická interpolace

Bikubická interpolace představuje nejsložitější prvek z uvedených interpolačních metod. Pro výpočet nové hodnoty ve dvourozměrném poli je zde uvažováno hned 16 nejbližších známých hodnot. Tyto hodnoty jsou do výsledku započítány různou měrou dle koeficientu, který je opět určen jejich vzdáleností od hledaného bodu. Tento přístup vede k odstranění mnoha nechtěných interpolačních artefaktů a poskytuje poměrně kvalitní výstupy. Velkou nevýhodou tohoto algoritmu je ovšem jeho výpočetní náročnost, která je znatelně vyšší než u předchozích algoritmů. Pokud však pracujeme s aplikacemi, ve kterých rychlost není rozhodujícím faktorem, je tento algoritmus vhodnou volbou pro zvětšování i zmenšování velikosti obrazů.[15]

Následující obrázek 1.11 představuje dílčí výstup algoritmu, který využívá bikubickou interpolaci pro získání osmkrát zmenšené verze vstupního obrazu.



Obr. 1.11: Obrazy získané pomocí bikubické interpolace při osminásobném zmenšení vstupního obrazu v obou souřadných osách. Na pravý obrázek byla navíc použita metoda průměrování výsledných hodnot.

2 ŘEŠENÍ

Praktická část této diplomové práce byla implementována v jazyce Java, který poskytuje poměrně široké spektrum knihoven určených pro práci s obrazy. Jednotlivé metody, které byly použity k nalezení podobnosti v obraze jsou rozebrány v následujícím textu a jsou zde komentovány výsledky a úspěšnost jednotlivých metod.

2.1 Databáze obrazů

Trénovací množina obrazů byla použita z [16]. Tato databáze obsahuje tisíc obrazů rozčleněných do deseti kategorií, kde každá má sto položek. Těmito kategoriemi jsou obrázky domorodců, pláží, květin, pokrmů, autobusů, dinosaurů, historických budov, slonů, koní a horských štítů. Všechny tyto obrázky disponují rozlišením 256x384 pixelů, respektive 384x256 při orientaci na šířku. Tato databáze obsahuje dostatečné množství různorodých vlastností obrazů, aby kvalitně ověřila účinnost daného algoritmu. Pár zástupců těchto kategorií obrázků lze vidět na obrázku 2.1.

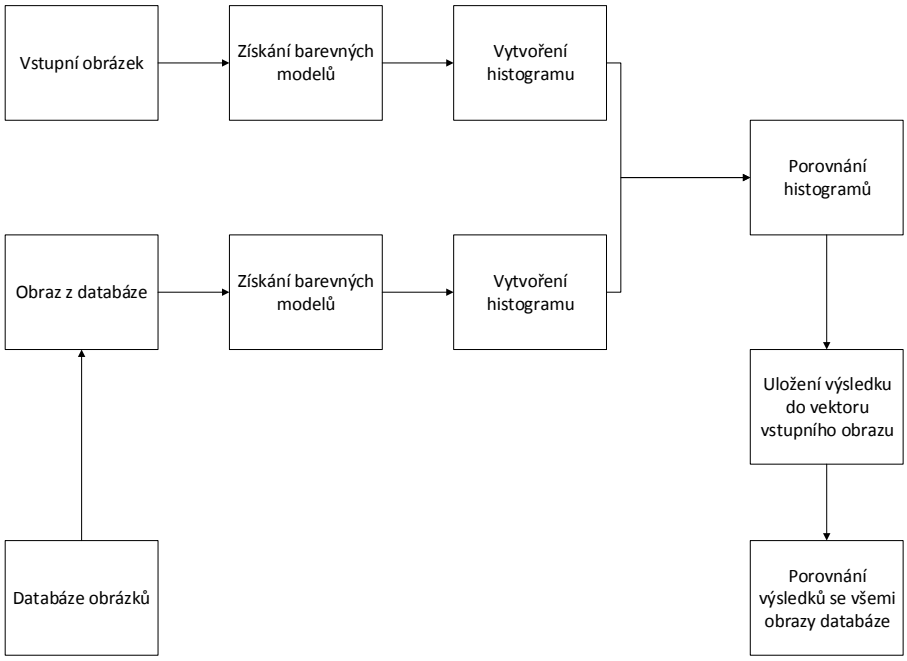
Obr. 2.1: Ukázka některých kategorií trénovací množiny obrázků.[16]

2.2 Metoda porovnávání histogramů

Tato metoda byla zvolena jako první, kvůli své poměrně jednoduché implementaci, díky které lze dosáhnout lepšího pochopení problematiky podobnosti obrazu. Její princip spočívá v extrakci histogramu obrazu, který představuje četnost zastoupení jednotlivých barev. Tento histogram lze zapsat jako pole datového typu Integer, jehož pozice v poli odpovídá barevnému odstínu daného kanálu. Jeho hodnota poté ukazuje, kolikrát se v obraze tento barevný odstín vyskytuje. Protože barevný model RGB je zpravidla reprezentován hodnotami 0 až 255 pro jeho jednotlivé kanály, i velikost pole histogramu má tedy 256 položek.

Celkový rozdíl mezi obrazy tedy lze získat pomocí postupného porovnávání hodnot na stejných pozicích histogramů obou porovnávaných obrazů. Součet vzdáleností těchto hodnot poté udává celkový rozdíl mezi zadanými obrazy. Názorně lze tento princip vidět na diagramu 2.2.

Pro zjištění nejlepšího barevného modelu pro porovnání histogramů byly implementovány metody využívající barevného modelu RGB a jeho jasového vyjádření a dále pak i modelu HSB. Jednotlivé metody jsou v následujícím textu popsány a okomentovány. Hlavním kritériem hodnocení byla výsledná přesnost a rychlost, se kterou dokázali zadanou databázi klasifikovat.



Obr. 2.2: Diagram procesu porovnávání podobnosti dvou obrazů pomocí histogramu.

2.2.1 Jasový histogram

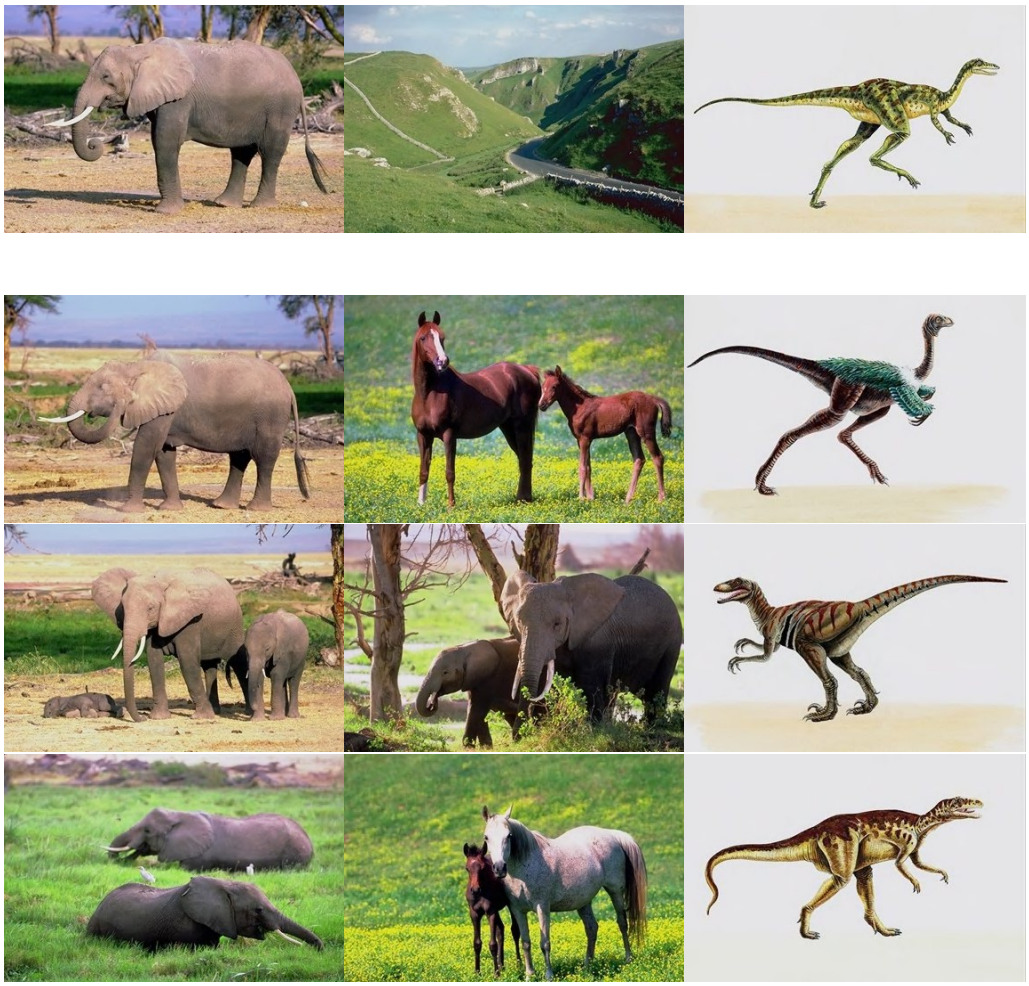
Tento typ histogramu je úzce spojen s barevným modelem RGB, ze kterého přímo vychází jeho hodnoty. Jak již bylo diskutováno v kapitole 1.4.1, je z hlediska vnímání obrazu lidským okem potřeba, aby byly jednotlivým kanálům přiřazeny koeficienty, které ovlivňují míru jejich zastoupení v jasovém histogramu. Míra zastoupení v základním nastavení třídy ColorProcessor, která je pro získání histogramu využita, je rozložena rovnoměrným dílem pro všechny kanály, tedy na třetiny. Proto bylo potřeba upravit toto rozložení na požadovaných 30% pro červený kanál, 59% pro kanál zelený a zbylých 11% pro modrý kanál. Již zmíněná třída ColorProcessor nabízí metodu `ColorProcessor.setWeightingFactors(0.3, 0.59, 0.11)`, která nám zajistí přidělení váhovacích koeficientů jednotlivým kanálům dle požadovaných kritérií.

Dalším krokem je získání samotného histogramu, což je jedna z dalších metod, kterou obsahuje třída ColorProcessor. Pokud tedy je dána cesta k souboru, díky které lze vytvořit objekt třídy ColorProcessor, je možné získat histogram tohoto objektu zápisem kódu `cp.getHistogram()`, kde `cp` představuje vytvořený objekt. Výstupem této metody je pole integerů o velikosti 256, což odpovídá počtu odstínů jednotlivých kanálů barevného modelu RGB.

Pro porovnání těchto polí byla vytvořena metoda, jejíž zápis je uveden v ukázce kódu níže. Zde je vidět, že pro vstupní parametry jsou postupně porovnávány jednotlivé hodnoty jejich korespondujících polí a jejich rozdíl následně přičítán k celkovému rozdílu, který zastupuje hodnota `diff`. Ta je po dosažení posledního pole vrácena jako výstup této metody. Právě výsledná hodnota `diff` je používána jako hlavní kritérium pro finální seřazení obrazů dle podobnosti.

```
private static int arrayCompare(int[] histogram1, int[]
   histogram2) {
               int diff = 0;
               for (int i = 0; i < histogram1.length;
                  i++) {
                     diff += Math.abs(histogram1[i] -
                        histogram2[i]);
               }
               return diff;
      }
```

Pro ukládání informací o velikosti rozdílu mezi obrazy, jejich názvem a uživatelským hodnocením podobnosti byla vytvořena třída ImageFeatures, která obsahuje metody pro zápis i extrakci těchto informací. Po získání hodnoty rozdílu mezi ob-



Obr. 2.3: Výstup metody porovnávání obrazů pomocí jasového histogramu.

razy je tedy tato informace společně s úplnou cestou k obrazu použita pro konstruktor objektu třídy ImageFeatures. Tyto objekty jsou následně vkládány do kolekce `ArrayList<ImageFeatures>`, která umožňuje jejich seřazení na základě zvoleného prvku. Zvoleným prvkem v tomto případě byla hodnota rozdílu dvou obrazů, která byla použita tak, aby se tyto objekty seřazovali vzestupně od nejmenší hodnoty rozdílu. Tento postup umožňuje jednoduchý přístup k nejvíce podobným obrazům databáze s obrazem vstupním a jejich následné zobrazení. Pro ilustraci funkčnosti této metody jsou na obrázku 2.3 ukázány pro tři náhodné vstupní obrazy (vrchní), vždy tři nejpodobnější obrazy z databáze ve sloupcích pod nimi.

2.2.2 RGB histogram

Tato metoda pracuje přímo s barevným modelem RGB a rovným dílem využívá všechny kanály tohoto modelu. S tímto faktem je ovšem spojena i nutnost zpracovat trojnásobné množství dat. Je tedy potřeba počítat s větší časovou i paměťovou

náročností. Dalším faktorem, který ovlivňuje rychlost této metody je fakt, že třída ColorProcessor vrací hodnoty kanálů RGB v datové jednotce byte. Ta je v jazyce Java reprezentována osmi bity, z nichž první určuje polaritu čísla. Teoretický rozsah 256 čísel je tedy rozdělen na -128 až 127. To představuje problém pro získání histogramu, jehož pole je adresováno od hodnoty 0. Proto je potřeba nejdříve převést tento rozsah do požadovaného tvaru. K tomuto účelu se dá využít metoda, kterou nabízí třída Byte, a jejíž zápis je `Byte.toUnsignedInt(x)`, která převede daný byte `x` na integer v již požadovaném rozsahu.

Takto upravené histogramy již lze stejně jako v předchozím případě předat metodě arrayCompare(). Téměř všechny dříve zmíněné úkony zajišťuje vytvořená metoda RGBhistogram, která jako vstupní parametry přijímá dvě trojice histogramů v poli bajtů a vrací celkový rozdíl mezi histogramy obrazů viz kód níže.

```
public static int RGBhistogram(byte[] red1,
        byte[] green1, byte[] blue1,
        byte[] red2, byte[] green2, byte[] blue2) {
        int[] redHistogram = new int[256];
        int[] greenHistogram = new int[256];
        int[] blueHistogram = new int[256];
        int[] redHistogram2 = new int[256];
        int[] greenHistogram2 = new int[256];
        int[] blueHistogram2 = new int[256];
        for (int x = 0; x < red1.length; x++) {
        redHistogram[Byte.toUnsignedInt(red1[x])]++;
        greenHistogram[Byte.toUnsignedInt(green1[x])]++;
        blueHistogram[Byte.toUnsignedInt(blue1[x])]++;
        redHistogram2[Byte.toUnsignedInt(red2[x])]++;
        greenHistogram2[Byte.toUnsignedInt(green2[x])]++;
        blueHistogram2[Byte.toUnsignedInt(blue2[x])]++;
                }
                int diff = arrayCompare(redHistogram,
                   redHistogram2) +
                arrayCompare(greenHistogram,
                   greenHistogram2) +
                arrayCompare(blueHistogram,
                   blueHistogram2);
                return diff;
        }
```

Na tomto kódu je zřetelně vidět paměťová náročnost této metody. Výstupy této metody jsou prezentovány na následujícím obrázku 2.4, který opět zobrazuje nahoře tři náhodně zvolené obrazy a ve sloupcích pod nimi tři obrazy, které byly touto metodou vyhodnoceny jako nejpodobnější.

Obr. 2.4: Výstup metody porovnávání obrazů pomocí RGB histogramu.

2.2.3 HSB histogram

Tato metoda využívá barevný model HSB a všem třem jeho kanálům přikládá stejnou váhu. Prvním krokem je tedy získání těchto kanálů. K tomu je využita metoda třídy ColorProcessor `getHSBStack()`, která vrací objekt třídy ImageStack. Ten si lze představit jako kolekci podobnou například třídě Vector, jehož je rozšířením. Indexace zde ovšem začíná na čísle 1, což je oproti ostatním kolekcím nezvyklé.

Z objektu ImageStack již tedy lze správnou indexací získat histogramy jednotlivých kanálů modelu HSB v poli integerů. Tento formát umožňuje předat tato pole metodě `arrayCompare()` a po sečtení výsledků všech kanálů modelu již lze tuto

hodnotu použít společně s cestou k obrázku jako konstruktor třídy ImageFeatures. Takto vytvořené objekty jsou opět uloženy do kolekce `ArrayList<ImageFeatures>`, která slouží k seřazení těchto objektů a jejich následné extrakci. Níže je kód metody, která zajišťuje porovnání dvou obrázků na základě jejich HSB histogramů. Tato metoda je volána pro porovnání každého obrazu databáze se vstupním obrazem. S rostoucí velikostí databáze tedy logicky roste i časová náročnost získání výsledků.

```
private static int getHSBhistogramComparison(
        ColorProcessor cp1,ColorProcessor cp2)
        {
        ImageStack is = cp1.getHSBStack();
        ImageStack is2 = cp2.getHSBStack();
        int[] histH = is.getProcessor(1).getHistogram();
        int[] histS = is.getProcessor(2).getHistogram();
        int[] histB = is.getProcessor(3).getHistogram();
        int[] histH2 =
           is2.getProcessor(1).getHistogram();
        int[] histS2 =
           is2.getProcessor(2).getHistogram();
        int[] histB2 =
           is2.getProcessor(3).getHistogram();
        int diff = arrayCompare(histH, histH2) +
           arrayCompare(histS, histS2)
        + arrayCompare(histB, histB2);

                return diff;
        }
```

Výstup této metody lze vidět na obrázku 2.5, kde jsou pro tři náhodné vstupní obrazy zobrazeny vždy tři jejich nejpodobnější obrazy z databáze, dle použité metody.

Obr. 2.5: Výstup metody porovnávání obrazů pomocí HSB histogramu.

2.3 Metody využívající interpolaci

Díky teoretickým poznatkům z kapitol 1.5 a 1.6 byli vytvořeny dva algoritmy, které umožňují poměrně kvalitní porovnávání digitálních obrazů na základě jejich barevného rozložení. Tyto algoritmy již berou v potaz prostorové rozložení jednotlivých pixelů a jsou tedy výpočetně značně náročnější. Tuto náročnost se snaží snížit použitím již zmíněných metod pro omezení barevné hloubky a velikosti digitálního obrazu.

2.3.1 Omezení barevné hloubky

Tato metoda pracuje s barevným modelem RGB, který je omezen na určený počet barev. Toto omezení je aplikováno především proto, aby bylo eliminováno postupné hromadění malých odchylek mezi jemnými odlišnostmi barev mezi podobnými obrazy. Pokud tedy vhodně zvolíme velikost barevné palety, se kterou budeme pracovat, můžeme výrazně vylepšit výsledek porovnávání v dané databázi digitálních obrazů.

Pro účely tohoto algoritmu byla vytvořena třída Method, která nabízí mimo jiné i metodu, která je hlavním tělem pro tento algoritmus. Tato metoda je statická, což znamená, že je možné ji volat i bez potřeby vytvářet instanci této třídy. První ze tří vstupních parametrů je typu File, a představuje adresář, ve kterém se nachází obrazy k porovnávání. Tento adresář je stejně jako u předchozích metod vybírán pomocí dialogového okna, které je nastaveno na akceptování pouze adresářů a nikoli konkrétních souborů.

Druhým a třetím vstupním parametrem je hodnota cílové šířky, respektive výšky obrazu, který je určen k porovnávání. V této metodě jsou porovnávány všechny tři kanály barevného modelu a to přes všechny odpovídající si pixely. Díky této skutečnosti je žádoucí urychlit algoritmus pomocí zmenšení velikosti všech obrazů na jednotnou velikost. Je potřeba nalézt vhodný kompromis mezi rychlostí algoritmu a přesností, kdy pro opravdu velké hodnoty zmenšení je sice dosahováno poměrně rychle výsledků, ovšem tyto výsledky jsou silně ovlivněny nepřesnostmi způsobenými tímto vysokým zmenšením porovnávaných obrazů.

Posledním vstupním parametrem je velikost barevné palety, která má být použita pro všechny obrazy. Jak již bylo zmíněno, můžeme tímto krokem eliminovat kumulování malých rozdílů, které mohou ve svém součtu výrazně ovlivnit určení míry podobnosti mezi obrazy. Stejně jako u předchozích dvou parametrů je třeba vhodně zvolit tuto velikost palety, kdy je třeba brát v úvahu fakt, že tato paleta je tvořena pro každý obraz zvlášť a algoritmus, který byl popsán v kapitole 1.5, má stejný počet iterací jako je velikost barevné palety. Na druhou stranu je potřeba zachovat dostatečné množství barev k tomu, aby jsme neeliminovali již relevantní

barevné rozdíly mezi obrazy.

Po určení těchto vstupních parametrů je prozkoumán zvolený adresář a položky, které odpovídají formátu JPEG, jsou uloženy do patřičného pole, mezi jehož položkami probíhá následné porovnávání. Nejdříve se díky funkci `Math.random()` vytvoří čtyři indexy pro pole obrazů, díky nimž jsou vybrány čtyři vstupní obrazy. Každý z těchto obrazů je předán další statické metodě, která přijímá stejné vstupní parametry jako předchozí. Vstupní část metody má podobu:

```
private static ArrayList<ImageFeatures>
   resizedQuantizedConcreteImageComparison
(File file, int dstWidth, int dstHeight, int maxcubes)
```

a je z ní patrné, že vrací kolekci objektů ImageFeatures, která již reprezentuje seřazený výsledek porovnávání databáze s konkrétním obrazem.

Na začátku této metody je vytvořena šestice bytových polí, do kterých jsou později ukládány hodnoty barevných kanálů vstupního a právě porovnávaného obrazu. Dalším krokem je vytvoření instance třídy ColorProcessor, která nabízí možnost nastavení jedné ze tří interpolačních metod, které byli probrány v teoretické části 1.6. Pro tuto metodu byla vybrána bikubická interpolace s požadovanými rozměry 50 x 50 pixelů. Jak již bylo zmíněno, tato metoda porovnává obrazy po odpovídajících si pixelech a je tedy třeba zajistit shodnou velikost všech porovnávaných obrazů.

Dalším krokem je omezení barevné palety na zadaný počet barev. Toho je dosaženo pomocí třídy MedianCut, která pomocí algoritmu popsaného v kapitole 1.5 upraví barevnou paletu vstupního obrazu na požadovaný počet barev. Tento počet poměrně silně ovlivňuje výsledek porovnávání a je poměrně náročné určit nejvhodnější hodnotu tohoto parametru tak, aby kvalitně reprezentovala všechny kategorie obrazů. Pro již zmíněné vstupní parametry a barevnou hloubku rovnou také padesáti, vypadají obrazy, které jsou následně porovnávány jako na obrázku 2.6. Tato hodnota barevné hloubky je ještě schopna vracet poměrně validní výsledky. Pokud však pokračujeme na nižší hodnotu barevné palety, výrazně se zhoršuje porovnávací schopnost algoritmu navzdory tomu, že pro lidské oko tento rozdíl není až tak markantní.

Z takto upraveného vstupního a právě porovnávaného obrazu jsou extrahovány hodnoty jednotlivých barevných kanálů do bytového pole, které bylo již dříve inicializováno na potřebnou velikost. Tato extrakce probíhá pomocí metody třídy ColorProcessor `public void getRGB(byte[] R, byte[] G, byte[] B)`, která z objektu této třídy, která metodu volá, získá jednotlivé barevné kanály a uloží je do polí uvedených ve vstupních parametrech metody.

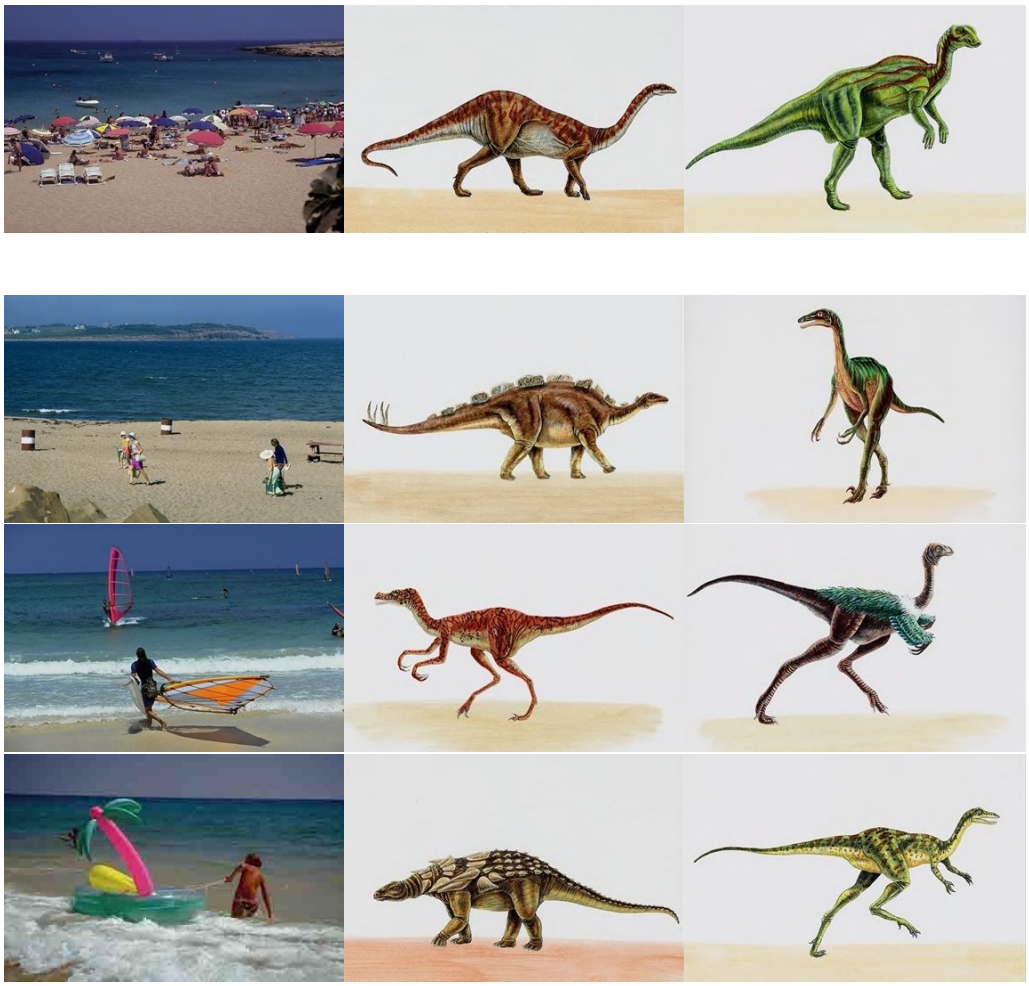
Pro porovnání odpovídajících polí byla vytvořena třída Comparator, která nabízí

Obr. 2.6: Dílčí výstup metody 2.3.1 pro zvolenou šířku i výšku obrazu rovnou padesáti a barevnou hloubku taktéž omezenou na padesát hodnot.

statickou metodu, která vrátí součet vzdáleností mezi prvky stejně velkých polí. Pro získání celkové hodnoty rozdílu mezi dvěma obrazy, která je rozhodující pro finální pořadí nejpodobnějších obrazů, lze tedy zapsat kód například takto:

```
int diff = Comparator.compareByteArray(red1, red2)
        + Comparator.compareByteArray(green1, green2)
        + Comparator.compareByteArray(blue1, blue2);
```

Takto získané hodnoty rozdílu jsou použity společně s úplnou cestou k porovnávanému souboru pro konstrukci objektu ImageFeatures, který je následně umístěn do kolekce těchto objektů a předán k prezentaci na uživatelském rozhraní. Výstup tohoto algoritmu je vidět na obrázku 2.7, kdy pro tři náhodně vybrané vstupní obrázky jsou prezentovány vždy tři s nejmenší hodnotou rozdílu.



Obr. 2.7: Výstup metody 2.3.1 pro vstupní parametry rovny padesáti pro výšku i šířku obrazu. Barevná paleta byla omezena na 100 barev.

2.3.2 Porovnání HSB kanálů

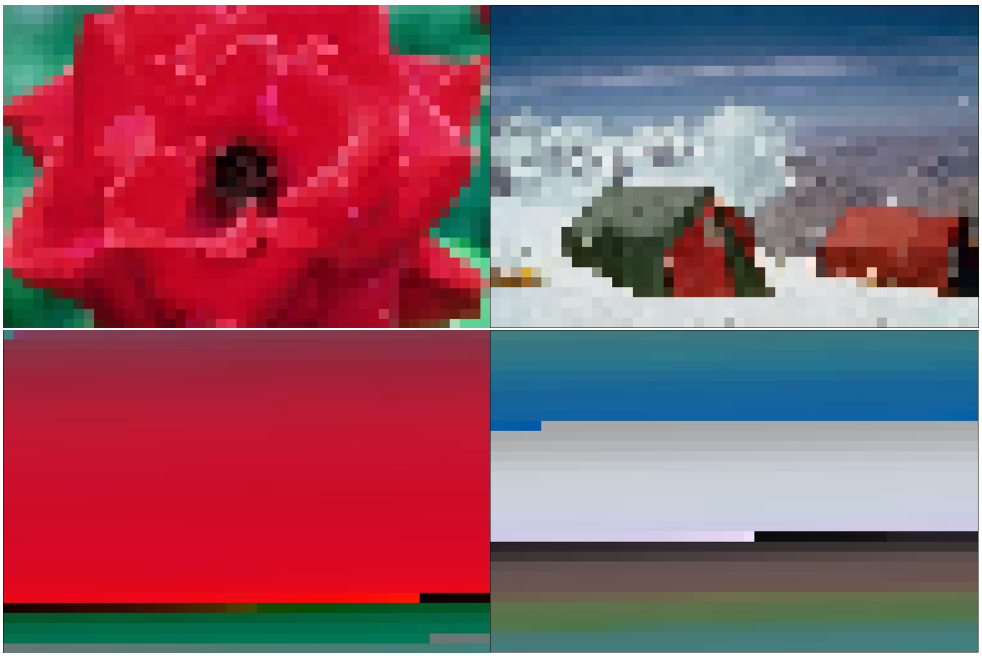
Tato metoda, jak již název napovídá pracuje s barevným modelem HSB, respektive s jeho kanály. Díky tomu je omezen rozdíl v hodnotě jasu pouze do jednoho ze tří kanálů, a obrazy, které se liší převážně jen jasovou složkou, budou vyhodnoceny jako více podobné. Další výhodou této metody je invariance proti natočení porovnávaných obrazů.

Samotný nápad na vytvoření této metody vznikl díky vedoucímu této diplomové práce, za což mu patří velké uznání. Implementace tohoto algoritmu je do značné míry podobná s předchozí metodou, popsanou v části 2.3.1. Hlavní metoda této funkce je umístěna v již zmíněné třídě Method. Jedná se opět o statickou metodu, která ovšem jako vstupní parametr přijímá pouze jeden objekt typu File. Ten představuje cestu k adresáři s databází fotek určených k porovnávání. V tomto adresáři jsou již probraným způsobem vybrány čtyři vstupní soubory, které jsou následně

porovnávány se zbytkem databáze.

Po načtení všech souborů z daného adresáře, které jsou formátu JPEG jsou tyto soubory porovnávány s každým ze 4 vstupních obrazů tak, že je vytvořen ze známé cesty objekt třídy ImageProcessor. Tento objekt je pomocí metody nejbližšího souseda následně zmenšen na velikost 50 pixelů pro výšku i šířku obrazu. Toto zmenšení je provedeno jak kvůli vyšší rychlosti algoritmu, tak kvůli udržení použitelnosti metody i na jinou databázi obrazů, než je použita pro tuto diplomovou práci. Pokud by totiž databáze obsahovala obrazy s variabilními rozměry a nebyla by použita unifikace na jednotný rozměr, vznikla by při porovnávání barevných kanálů potřeba doplnit jedno z polí na stejnou velikost druhého fiktivními hodnotami. Druhé řešení by bylo ignorovat hodnoty, jejichž pozice je již za velikostí prvního porovnávaného pole. Obě řešení nejsou příliš elegantní a vedou ke vzniku markantních chyb.

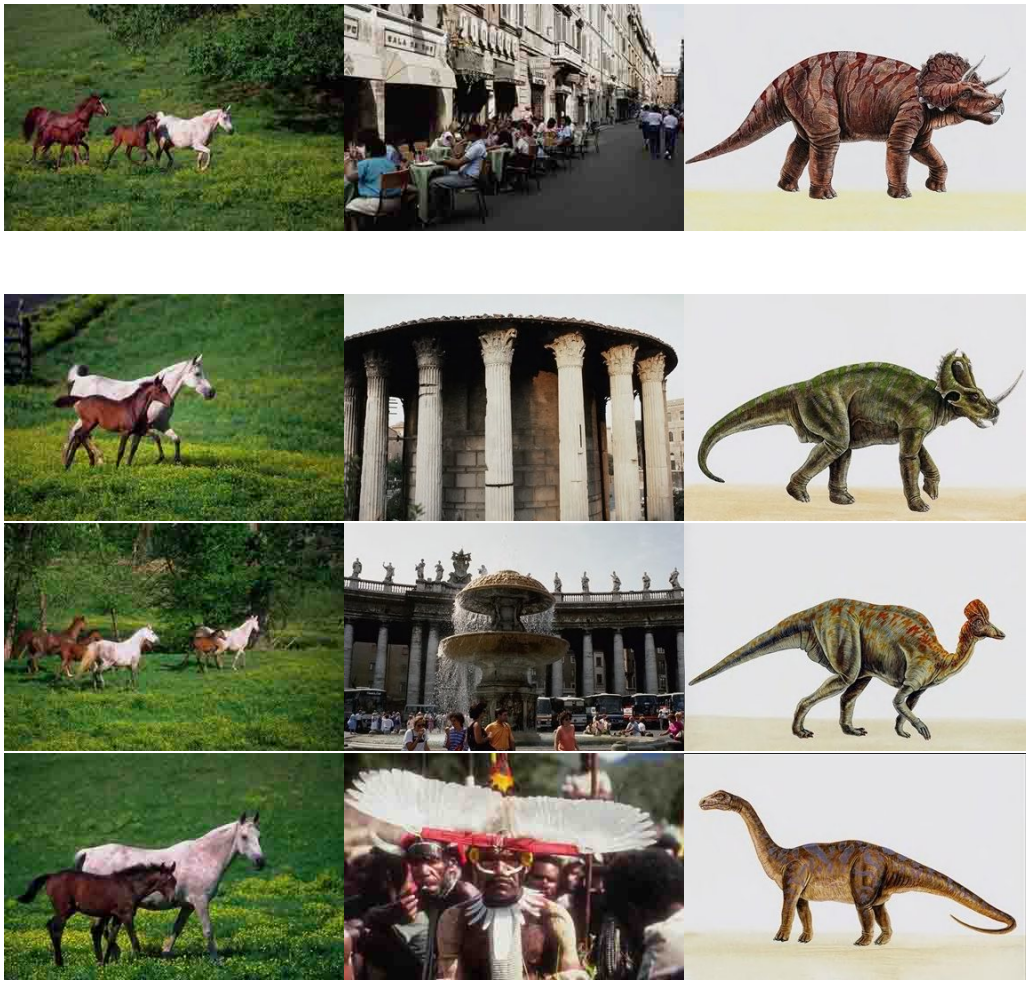
Po zmenšení porovnávaných obrazů na stejnou velikost je do již inicializovaných polí uložena hodnota barevných kanálů HSB těchto obrazů. Tato pole jsou následně pomocí funkce třídy Arrays vzestupně seřazena a předána k porovnání metodě `compareByteArray`. To nám zajistí velkou míru shody pro obrázky s podobným barevným charakterem, bez ohledu na prostorové uspořádání jednotlivých pixelů. Pro lepší představu tohoto procesu je na obrázku 2.8 vidět, jak toto seřazení změní podobu již zmenšeného vstupního obrazu.



Obr. 2.8: Dílčí výstup metody porovnávání HSB kanálů. Nahoře zmenšená verze původního obrázku na 50 pixelů pro šířku i výšku, dole stejný obraz se seřazenými hodnotami barevných kanálů.

Rozdíl získaný tímto porovnáním je opět použit pro vytvoření objektů třídy Image-

Features, které slouží pro prezentaci výsledků na uživatelském rozhraní. Na obrázku 2.9 je zobrazen výstup metody pro tři náhodně vybrané vstupní obrazy, kdy pro každý z nich jsou demonstrovány tři nejpodobnější dle vytvořeného algoritmu.



Obr. 2.9: Výstup metody 2.3.2, pro tři horní obrazy, které jsou vstupními pro tuto metodu a ve sloupcích pod nimi tři nejpodobnější dle vytvořeného algoritmu.

2.4 Uživatelské rozhraní

Pro potřeby lepší přehlednosti dosažených výsledků bylo vytvořeno pomocí programu JavaFX Scene Builder uživatelské rozhraní, ve kterém jsou implementovány prostředky pro zobrazení dosaženým výsledků a pro uživatelské hodnocení kvality těchto výsledků. Důležitým požadavkem na toto rozhraní bylo, aby přehledně zobrazovalo více vstupních obrazů a k nim jejich nejpodobnější obrazy ze zadané databáze.

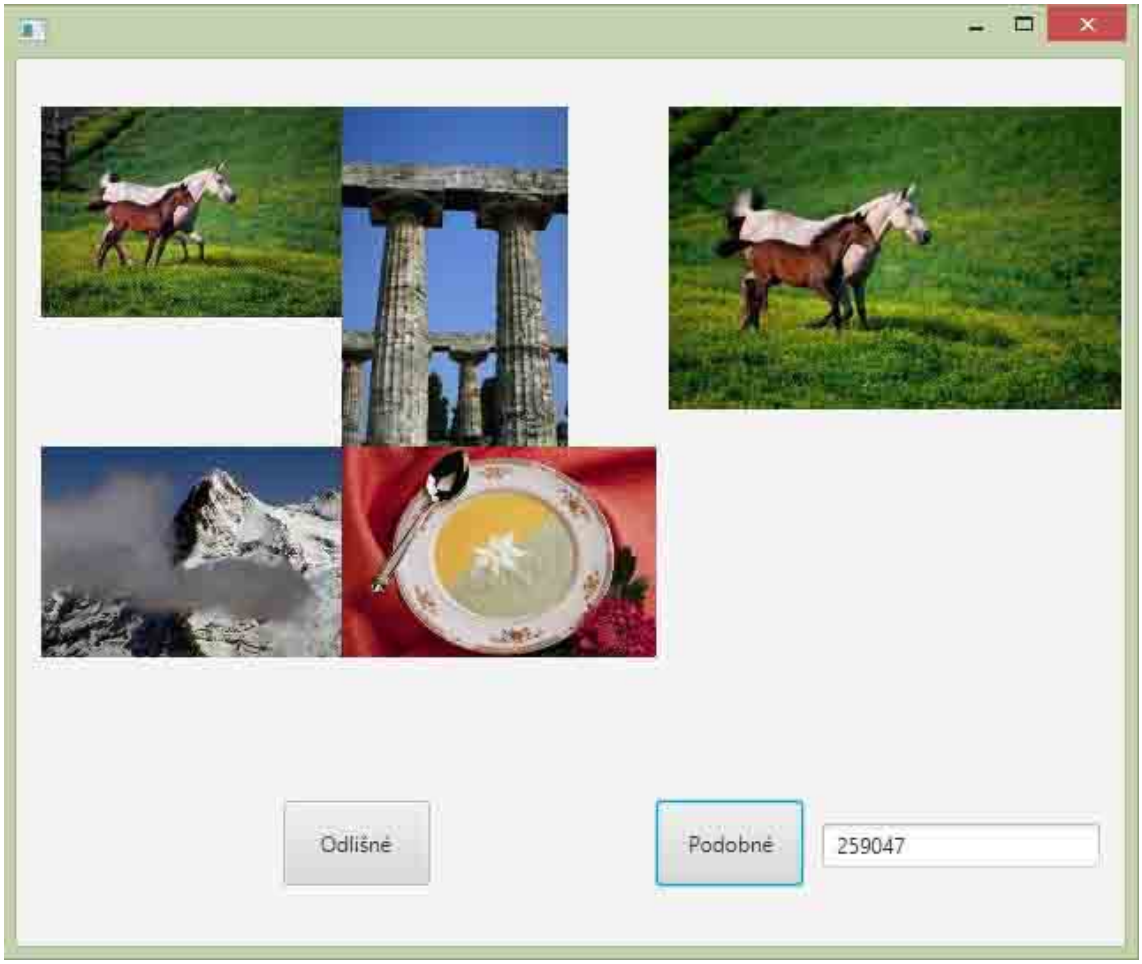
Aby bylo možné zobrazovat výsledky i vstupní obrazy, je v uživatelském prostředí zařazen objekt ImageView z knihovny JavaFX. Tento prvek je zařazen do rozhraní pětkrát, aby zobrazoval náhledy čtyř vstupních obrázků a aktuálního snímku z vybraných nejpodobnějších obrázků databáze. Rozložení těchto prvků je vidět na obrázku 2.10. Velikost čtyř prvků ImageView vlevo, odpovídá polovině rozlišení obrázků z trénovací databáze, což je 178x192 pixelů. ImageView vpravo, který je použit pro zobrazení nejpodobnějších obrázků disponuje rozlišením 256x384 a zobrazuje tedy originální (nezmenšené) rozlišení obrázků orientovaných na výšku.

Možný postup potřebný pro zobrazení obrázku v komponentě ImageView je ukázán v kódu níže, kde je nejdříve vytvořen objekt třídy File díky cestě k obrázku, která je uložena v kolekci náhodných obrazů, z tohoto objektu je vytvořen objekt třídy Image, který již slouží přímo jako vstupní parametr komponenty ImageView.

```
File f = new File(randomImages.get(0).getName());
Image im1 = new Image(f.toURI().toString());
this.imageView1.setImage(im1);
```

Pro zajištění objektivnosti výsledků je do zhotovené metody určující podobnost implementován náhodný výběr vstupních obrazů. Trénovací množina obsahuje kategorie, které disponují poměrně jednoduchými znaky a tím tedy velmi usnadňují určení podobných obrázků. Jako příklad lze uvést kategorii dinosaurů, kteří jsou všichni zasazeni do jednotného pozadí, čímž výrazně ovlivňují svůj histogram. Pokud by se tedy pro určení kvality aplikované metody záměrně vybírala například pouze tato kategorie, nedosáhlo by se objektivního výsledku. Proto je vložen do algoritmu pomocí funkce třídy `Math.random()` náhodný výběr vstupních obrázků, čímž je zajištěna vyšší míra objektivnosti dosažených výsledků.

Celý proces po spuštění vytvořené aplikace se tedy skládá z výběru metody, kterou chceme použít pro porovnávání, následuje výběr složky s databází obrázků, u kterých chceme zjistit podobnost. To je zajištěno pomocí dialogového okna, které je obsluhováno třídou JFileChooser. Po zvolení požadovaného adresáře je pomocí třídy File získáno pole všech souborů ze zvoleného adresáře, které jsou zároveň soubory obrazovými. Tyto soubory jsou postupně převáděny do histogramů, či barevných



Obr. 2.10: Podoba uživatelského rozhraní pro vytvořenou aplikaci.

kanálů dle zvolené metody a porovnávány s náhodně vybranými vstupními obrazy. Výsledný rozdíl je použit společně s úplnou cestou k porovnávanému souboru jako konstruktor třídy ImageFeatures a uložen do kolekce, která je na konci porovnávacího procesu seřazena právě dle hodnoty tohoto rozdílu. Takto vytvořené kolekce jsou využity pro zobrazení nejpodobnějších obrazů obrazu vstupnímu.

Již zmíněné objekty ImageView mají implementovánu metodu, která po kliknutí na jednu z těchto komponent uživatelského rozhraní zobrazí právě jeho nejpodobnější obrazy. Tato funkce vyžaduje prezenci všech kolekcí pro čtyři náhodné vstupní obrazy, díky kterým je výsledek uživateli prezentován okamžitě bez nutnosti dalších výpočtů. Těmito kolekcemi lze postupně procházet a prohlédnout si tak kompletní pořadí podobnosti databázových obrazů.

2.4.1 Hodnocení uživatele

Pro hodnocení kvality dosažených výsledků danou metodou jsou do uživatelského rozhraní zařazeny tlačítka ze třídy Button, díky kterým lze procházet seřazenou databází nejpodobnějších obrazů obrazu vstupnímu. Jak je vidět na obrázku 2.10, jedná se o dvojici tlačítek, které mají za úkol hodnotit úspěšnost zadané metody ve vybrání nejpodobnějších obrazů. Pokud vstupní obraz náleží do stejné kategorie jako výstupní, kliknutím na tlačítko „Podobné“ jsou o jednotku inkrementovány proměnné vytvořené třídy Controller `valuated, goodRetrieval`. Podílem těchto dvou proměnných je následně počítána hodnota úspěšnosti zvolené metody, kdy po dosažení hodnoty 99 u proměnné `valuated` je touto hodnotou vydělena proměnná `goodReatrieval`. Tímto podílem tedy získáme procento úspěšně zvolených podobných obrazů databáze.

Druhým tlačítkem je „Odlišné“, které inkrementuje o jednotku pouze proměnnou `valuated`, což ve výsledném poměru mění velikost procentuální úspěšnosti zvolené metody. Další funkcí obslužné metody tohoto tlačítka je výrazné zvýšení hodnoty `valuation` objektu třídy ImageFeatures, která se stejnou měrou jako hodnota rozdílu porovnávaných obrázků podílí na řazení v kolekci. Pokud tedy klikneme u obrazu na tlačítko „Odlišné“, posuneme tím tento obraz na mnohem zadnější pozici výsledného pořadí. Hodnota, která se přiřazuje proměnné `valuation` byla experimentálně stanovena na 1.000.000 a nelze tedy předem říci o kolik pozic v pořadí nám hodnocený obraz posune. Vzdálenosti mezi hodnotami rozdílu dvou obrazů jsou totiž velmi různorodé a je tedy potřeba zvolit hodnotu tak, aby vždy ovlivnila výsledné pořadí obrazů. Po novém seřazení kolekce je tedy zobrazen obrázek, který nově zaujal právě ohodnocenou pozici.

Posledním prvkem uživatelského rozhraní je komponenta TextField, která zobrazuje hodnotu rozdílu mezi vstupním obrazem a aktuálně zobrazeným obrazem z

výsledné kolekce. Tato hodnota roste s postupným procházením výslednou kolekcí obrazů a je tedy díky ní možné získat představu o reálné velikosti vzdálenosti porovnávaných obrazů.

2.5 Úspěšnost implementovaných metod

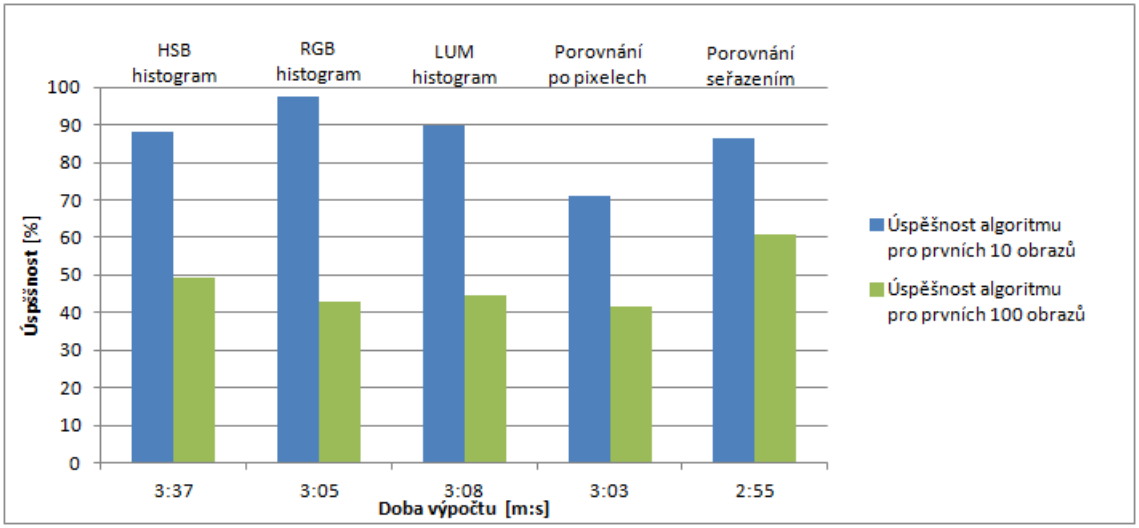
Pro zjištění procentuální úspěšnosti byla zvolena metoda, při které se zjišťuje počet správně zvolených obrazů databáze v prvních 99ti obrazech, vyhodnocených jako nejpodobnějších. Jak již bylo zmíněno, trénovací databáze se skládá z tisíce obrázků, rozdělených do deseti kategorií po sto obrazech. Díky tomu, že vstupní obrazy jsou vybírány náhodně a ze stejné databáze, ve které se následně hledají nejpodobnější obrazy, je možné pro dokonalý vyhledávací systém nalézt pouze 99 nejpodobnějších obrazů. Z toho vyplývá, že vstupní obraz je při vyhledávání ignorován, neboť jeho hodnota rozdílu by byla nulová, což by ho řadilo na první pozici kolekce.

Pro metodu porovnávající histogramy jednotlivých obrazů a využívající barevný model HSB byla zjištěna úspěšnost 49,25%, což znamená že do prvních 99 nejlepších obrazů celé databáze bylo vybráno 49 obrazů ze správné kategorie. Metoda využívající histogramu všech kanálů barevného modelu RGB dosahuje úspěšnosti 43% a poslední metoda, která využívá jasový histogram dosahuje úspěšnosti 44.75%.

Metoda 2.3.1, které porovnává odpovídající si pixely obrazů a využívá k tomu barevný model RGB dokáže do první stovky vybraných obrazů zařadit průměrně 42 ze správné kategorie. Toto číslo není nikterak závratné a bylo dosaženo pro relativně malé vstupní parametry, kdy rozměry všech obrazů byly upraveny na 50 na 50 pixelů a barevná hloubka byla omezena na 100 barev. Tyto hodnoty jsou schopny zajistit poměrně rychlý běh algoritmu, ovšem už nejsou schopny kvalitně roztřídit všechny kategorie obrazů databáze. Pokud tedy rychlost není hlavním kritériem, je možné dosáhnout i lepších výsledků. Na druhou stranu je třeba zmínit, že obecně porovnávání obrazů po jednotlivých pixelech, či jejich seskupeních, nepatří mezi nejpřesnější metody.

Metoda 2.3.2, která pro určení podobnosti mezi obrazy využívá vzestupně seřazených hodnot kanálu barevného modelu HSB, je díky poměrně dobře optimalizované implementaci nejrychlejší z vytvořených metod. Zároveň dosahuje i při nízkých vstupních parametrech relativně kvalitních výsledků. Tyto obrazy jsou opět zmenšeny na velikost 50 na 50 pixelů, ovšem při zachování původní barevné hloubky. Při těchto parametrech dosahuje tato metoda průměrné úspěšnosti téměř 61%, což ji řadí na první místo mezi vytvořenými algoritmy.

Je potřeba říci že obrazy trénovací množiny jsou opravdu různorodé a občas je členství některých obrazů v dané kategorii minimálně diskutabilní. Proto pro



Obr. 2.11: Porovnání úspěšnosti a výpočetní náročnosti vytvořených metod.

srovnání s předchozími výsledky byla zkoumána úspěšnost pro prvních deset nejpodobnějších obrazů. V tomto testu dosahuje metoda využívající barevného modelu HSB výsledku 88%, RGB 97,5% a jasový histogram v tomto testu obdržel 90%. Pro metodu 2.3.1 je tato úspěšnost 71,25% a pro metodu 2.3.2 je tato hodnota 86,25%. Přehled všech vytvořených metod, včetně jejich úspěšností a doby potřebné k výpočtu je zobrazen na grafu 2.11, kde je pro jednotlivé metody zobrazena na ose x průměrná doba, potřebná k výpočtu a na ose y je procentuální úspěšnost metod.

3 ZÁVĚR

V této práci byly rozebrány teoretické základy podobnosti obrazů na základě barvy. Bylo zde vysvětleno, proč je s rostoucí velikostí obrazových databází neudržitelné hodnotit všechny obrazy pomocí lidského zásahu. To je hlavní motivací k velkému vývoji a zkoumání v oblasti vytváření systémů, které hodnotí podobnost v obraze na základě jeho nízkoúrovňových příznaků, jako jsou jeho barva, textura, rozložení prvků a mnoho dalších.

V teoretické části práce byly diskutovány různé druhy systémů zabývajících se získáváním podobných obrazů jako je obraz vstupní. Dvěma hlavními kategoriemi těchto systémů jsou textově založené systémy, jejichž scéna je hodnocena pomocí lidského vnímání. Jako druhé byly představeny CBIR systémy, které podobnost vyhodnocují na základě nízkoúrovňových příznaků a pracují zcela automaticky. Nevýhodou je výpočetní náročnost a přesnost těchto systémů.

Dalšími tématem této práce byl barevný model HSB, který je díky své blízkosti k lidskému vnímání často využíván v CBIR systémech, stejně tak jako Fuzzy množiny, které představují další kapitolu této práce. Velká pozornost byla věnována principu a vlastnostem histogramu. Jeho využití v oblasti digitální fotografie a také při porovnávání podobnosti obrazu. Dalšími významnými kapitolami teoretické části byly interpolační metody a problematika barevné hloubky obrazu.

V praktické části byla představena vybraná trénovací databáze fotografií, její kategorie a důležité vlastnosti. Hlavním bodem této práce byla metoda porovnávání histogramů, která zde byla implementována ve třech variantách. První z nich byla metoda využívající jasového histogramu, druhá metoda byla založená na všech barevných kanálech modelu RGB a poslední využívala barevný model HSB. Neméně důležitou částí byla implementace metod využívajících interpolačních algoritmů, popřípadě omezení barevné hloubky pro porovnávání obrazů.

Předposlední kapitola této diplomové práce pojednávala o vytvořeném uživatelském rozhraní, které přehledně prezentuje dosažené výsledky jednotlivých metod. Toto rozhraní umožňuje procházení seřazených výsledků a také dovoluje hodnotit správnost vyhodnocení podobnosti jednotlivých obrazů.

V poslední kapitole byla diskutována procentuální úspěšnost vytvořených metod. Jejich výpočetní náročnost a vzájemné srovnání ve vyhodnocení deseti a sta nejpodobnějších obrazů.

LITERATURA

[1] DHARANI, T. a I. L. AROQUIARAJ. *A survey on content based image retrieval.* 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering. IEEE, 2013, s. 485-490. DOI: 10.1109/ICPRIME.2013.6496719. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6496719>.

[2] WANG, Guosheng. *A Survey on Training Algorithms for Support Vector Machine Classifiers.* 2008 Fourth International Conference on Networked Computing and Advanced Information Management. IEEE, 2008, s. 123-128. DOI: 10.1109/NCM.2008.103. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4623990>.

[3] HOI, S.C.H., M.R. LYU a R. JIN. *A unified log-based relevance feedback scheme for image retrieval.* IEEE Transactions on Knowledge and Data Engineering. IEEE, 2006, vol. 18, issue 4, s. 509-524. DOI: 10.1109/TKDE.2006.1599389. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1599389>.

[4] YANFANG GAO, S.C.H., M.R. JIWEI WANG a R. JIN. *Active learning method of bayesian networks classifier based on cost-sensitive sampling.* 2011 IEEE International Conference on Computer Science and Automation Engineering. IEEE, 2011, vol. 18, issue 4, s. 233-236. DOI: 10.1109/CSAE.2011.5952671. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5952671>.

[5] *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003.* Proceedings. IEEE Comput. Soc, 2003, vol. 18, issue 4. ISSN 1041-4347. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1211412>.

[6] GOSSELIN, P.H., M. CORD a R. ZHIHUA ZHANG. *Semantic kernel learning for interactive image retrieval.* IEEE International Conference on Image Processing 2005. IEEE, 2005, vol. 18, issue 4, I-1177. DOI: 10.1109/ICIP.2005.1529966. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1529966>.

[7] RAJMIC, P.; SCHIMMEL, J. *Moderní počítačová grafika.* Brno: Vysoké učení technické v Brně, 2013. ISBN: 978-80-214-4906- 0.

[8] L.A. Zadeh, Fuzzy sets, *Information Control*, 8:338–353, 1965.

[9] Cambridge in colour [online]. [cit. 2014-12-02]. Dostupné z: <http://www.cambridgeincolour.com/>.

[10] Info cellar [online]. [cit. 2015-05-09]. Dostupné z: <http://www.infocellar.com/Graphics/color-theory.htm>.

[11] P. Heckbert, *Color image quantization for frame buffer display,* Computer Graphics, **16**(3), pp. 297-307 (1982).

[12] KEYS, R. *Cubic convolution interpolation for digital image processing.* IEEE Transactions on Acoustics, Speech, and Signal Processing [online]. 1981, vol. 29, issue 6, s. 1153-1160 [cit. 2015-05-16]. DOI: 10.1109/tassp.1981.1163711.

[13] HU, L.P., C. WANG a H.C. YIN. (2D)2k-NNDA: *Two-directional two-dimensional k-nearest neighbour discriminant analysis for target recognition.* In: Proceedings of 2011 IEEE CIE International Conference on Radar [online]. 2011 [cit. 2015-05-16]. DOI: 10.1109/cie-radar.2011.6159878.

[14] SA, Yang. *Improved Bilinear Interpolation Method for Image Fast Processing.* In: 2014 7th International Conference on Intelligent Computation Technology and Automation [online]. 2014 [cit. 2015-05-16]. DOI: 10.1109/icicta.2014.82.

[15] SEKAR, K., V. DURAISAMY a A. M. REMIMOL. *An approach of image scaling using DWT and bicubic interpolation.* In: 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE) [online]. 2014 [cit. 2015-05-16]. DOI: 10.1109/icgccee.2014.6922406.

[16] James Z. Wange Research group. [online] <http://wang.ist.psu.edu/docs/home.shtml>. [cit. 2014-12-10].

[17] JING LI. *The application of CBIR-based system for the product in electronic retailing.* 2010 IEEE 11th International Conference on Computer-Aided Industrial Design. IEEE, 2010, s. 1327-1330. DOI: 10.1109/CAIDCD.2010.5681962. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5681962>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

CBIR Content Based Image Retrieval

CMY Barevný prostor reprezentovaný barvami Cyan, Magenta, Yellow

$f_A(x)$ funkce příslušnosti fuzzy množiny

HSB Barevný model reprezentovaný kanály Hue, Saturation, Brightness

JPEG Joint Photographic Experts Group

RGB Barevný prostor reprezentovaný barvami Red, Green, Blue

SVM Support Vector Machines