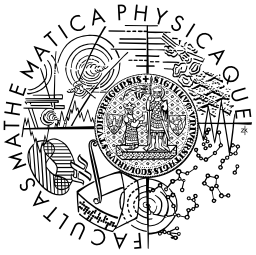


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Peter Dräxler

Hľadanie známeho obrázku pomocou jednoduchých farebných nákresov

Katedra softwarového inženýrství

Vedoucí bakalářské práce:	doc. RNDr. Jakub Lokoč, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové a datové inženýrství Bc. R8

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Týmto by som chcel poďakovať doc. RNDr. Jakub Lokoč, Ph.D. vedúcemu tejto bakalárskej práce. Cením si čas, ktorý ste mi venovali, Vašu odbornú pomoc, rýchle odpovede na e-maily, konštruktívnu kritiku a povzbudenia.

Ďakujem Arturovi Fingerovi za vytvorenie nástroja na zber dát od užívateľov.

Ďakujem RNDr. Josef Pelikán, za kontrolu kapitoly 2.

Ďalej ďakujem svojím rodičom za podporu počas celého môjho štúdia a Eve Semanovej za podporu a pomoc s korektúrou práce.

Název práce: Hľadanie známeho obrázku pomocou jednoduchých farebných nákresov

Autor: Peter Dräxler

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Jakub Lokoč, Ph.D., Katedra softwarového inženýrství

Abstrakt: S narastajúcim množstvom multimediálneho obsahu narastá dôležitosť dostupnosti kvalitných nástrojov pre vyhľadávanie. Bez vhodných vzorových dotazov je náročné vyhodnotiť kvalitu akéhokoľvek vyhľadávacieho algoritmu. V tejto práci sa venujeme vyhľadávaniu známeho obrázku v databáze. Popisujeme experiment, pri ktorom bolo zozbieraných 2 500 jednoduchých farebných nákresov od reálnych užívateľov. Pomocou týchto nákresov vyhodnocujeme presnosť, s akou si je užívateľ schopný zapamätať farby na obrázku. Získané dáta použijeme k vyhodnoteniu presnosti rôznych vyhľadávacích modelov. Súčasťou práce je aj webová aplikácia, ktorá umožňuje vyhľadávať v obrázkoch.

Klíčová slova: Vyhľadávanie pomocou nákresu, Vyhľadávanie vo videu, Hľadanie obrázku

Title: Known-item search in image datasets using simple color sketches

Author: Peter Dräxler

Department: Department of Software Engineering

Supervisor: doc. RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: With growing amount of multimedia content, availability of high quality search tools is becoming more important than ever. It's difficult to evaluate quality of any search algorithm without sample queries. In this paper, we present an experiment in which 2 500 sample queries were gathered. We use these queries to evaluate accuracy of users for remembering and entering colors. We use these data to evaluate effectiveness of various ranking models. A web application for image search is a part of this work.

Keywords: Known Item Search, Sketch Search, Video retrieval, Image search

Obsah

Úvod

Stalo sa vám už niekedy, že ste si spomenuli na obrázok alebo video, ktoré ste v minulosti videli a chceli ste ho niekomu ukázať? Ak ste si nezapamätali názov videa ani text, ktorý sa pri videu vyskytuje, nájsť dané video môže byť zložité.

Tento problém, kedy užívateľ pozná scénu, ktorú hľadá, vie v ktorej kolekcií sa nachádza, ale nepozná jej umiestnenie, sa nazýva hľadanie známej scény (angl. *Known-Item Search (KIS)*). Ak je kolekcia malá, užívateľ môže scénu nájsť sekvenčne. Pri väčších kolekciách sa však nezaobíde bez efektívnych vyhľadávacích modelov. Vzniklo preto množstvo modelov a nástrojov, ktoré vyhľadávanie zefektívňujú.

V tejto bakalárskej práci uvedieme niekoľko príkladov, akou formou je možné zadať dotaz pre vyhľadávanie v obrázkoch a ukážeme princíp práce príslušných vyhľadávacích modelov. Podrobnejšie skúmame vyhľadávanie pomocou jednoduchých farebných nákresov, kde užívateľ na plátno umiestni 3 až 5 farebných krúžkov. Jednoduché nákresy sme zvolili preto, že tento spôsob vyhľadávania považujeme za užívateľsky prívetivý, keďže od užívateľa nevyžadujeme umelecké zručnosti a naviac predpokladáme, že užívateľ obrázok videl v minulosti a jeho detaily si nepamätá. Napriek svojej jednoduchosti sa tento spôsob zadávania vstupu ukazuje ako efektívny. (Lokoč a kol., 2019)

V ďalších častiach tejto práce vysvetlíme ako užívatelia vnímajú farby, z pohľadu, ktorý je pre vyhľadávanie v obrázkoch dôležitý, predstavíme náš model pre vyhľadávanie v obrázkoch pomocou farebných nákresov, potom vykonáme experiment, pomocou ktorého získame viac než 2500 nákresov od reálnych užívateľov a tieto nákresy využijeme pre optimalizáciu parametrov nášho vyhľadávacieho modelu.

0.1 Využitie vyhľadávania v obrázkoch

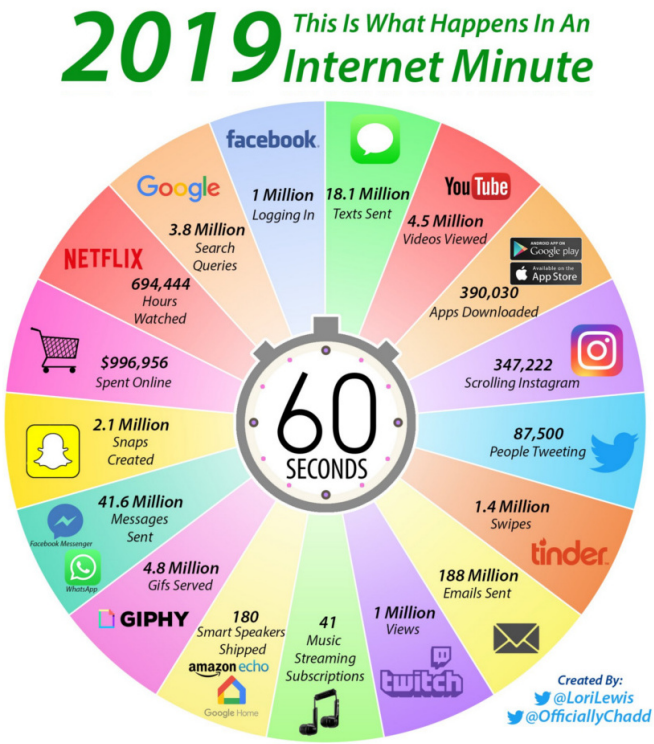
Na začiatok uvedieme niekoľko príkladov existujúcich kolekcií, v ktorých je vyhľadávanie potrebné.

0.1.1 Kolekcie multimédií dostupné na internete

Lori Lewis, viceprezidentka vydavateľstva MODERN LUXURY, každoročne publikuje infografiku "Takto vyzerá minúta na Internete"(v originále "This Is What Happens In An Internet Minute") na ktorej môžeme vidieť odhadovanú aktivitu na najpopulárnejších službách Internetu.

Spoločnosti ako Facebook, YouTube, Instagram, Twitter, Tinder, Twitch, Giphy, Messenger a Snapchat obsluhujú kolekcie obrázkov a videí, ktoré sú tvorené ich užívateľmi. Z infografiky je zrejmé, že tieto kolekcie narastajú enormným tempom. Ako užívateľ nájde video, ktoré mu ukazoval kamarát?

Tvorcovia obsahu, ktorý je publikovaný na internete, sa spoliehajú na fotobanky, aby našli vhodný obrázok ku svojmu článku alebo videu. Lepší vyhľadávací algoritmus môže byť faktorom podľa ktorého sa užívatelia budú rozhodovať, ktorú fotobanku využijú.



Obrázok 1: Čo sa udeje na internete za 60 sekúnd (Lewis, 2019).

Za zmienku stojí aj to, že na vyhľadávanie vo videu sa dajú aplikovať postupy pre vyhľadávanie v obrázkoch, pričom za obrázky sú považované jednotlivé snímky videa.

0.1.2 Kolekcie vytvorené pri výskume

Za všetky využitia vyhľadávania obrázkov vo vede uvedieme len dva príklady.

Pri vývoji samo-riadiacich áut automobilky nechávajú nepretržite jazdiť autá, ktoré kamerou zaznamenávajú cestu a jej okolie. Niekedy potrebujú v tisíckach hodín videí nájsť záznam, kde auto prechádza určitým typom križovatky.

Biológovia zasa môžu vyhľadávanie vo videu využiť pri hľadaní zaujímavých momentov v zázname delenia buniek či priebehu chemickej reakcie.

0.1.3 Osobné kolekcie multimédií

Takmer každý užívateľ chytrého mobilného telefónu dnes vlastní osobnú kolekciu fotografií.

Spoločnosť *Magisto* v roku 2015 analyzovala, koľko multimediálneho obsahu tvoria jej noví užívatelia. Kým priemerný užívateľ tejto aplikácie na editáciu videa svojim mobilným telefónom odfotí 150 fotografií mesačne a natočí 7 minút videa, žena do 25 rokov žijúca v Japonsku na svojom iPhone odfotí až 250 snímkov mesačne (Roettgers, 2015).

Pri podobnom množstve fotografií môže byť sekvenčné vyhľadávanie snímku v 5 ročnom archíve obtiažne a niektorí užívatelia by určite ocenili kvalitné nástroje, ktoré by im vyhľadávanie zjednodušili.

Obrázok 2: Príklad obrázku, ktorý vývojári samo-riadiacich áut môžu potrebovať nájsť v tisíckach hodín videozáznamu. (Manak, 2016).

0.2 Zhrnutie výskumu a výsledkov

V rámci tohto výskumu sme zozbierali viac než 2 500 vzorových nákresov, ktoré užívatelia kreslili po tom ako im bol obrázok ukázaný na 7 sekúnd. Vyhodnotili sme s akou presnosťou sú užívatelia schopní nakresliť farby z ukázaného obrázku.

Nákresy sme využili ako dotazy pre vyhľadávanie a sledovali sme pozíciu obrázku, ku ktorému nákres patrí. Na základe priemernej pozície obrázku vo výsledkoch vyhľadávania sme ohodnotili kvalitu rôznych vyhľadávacích algoritmov. Našli sme algoritmus, ktorý nám umožní pomocou veľmi jednoduchého nákresu vyhľadávanie obmedziť na malú časť databázy.

1. Možnosti vyhľadávania v obrázkoch a vo videách

Kým súkromné kolekcie bežného užívateľa môžu byť dostatočne malé na to, aby boli prehľadávané sekvenčne, veľkosť multimediálnych kolekcií dostupných na internete sekvenčné vyhľadávanie neumožňuje. Kvalitné vyhľadávacie algoritmy sú teda čím ďalej tým potrebnejšie.

Dnes sú dostupné tri metódy vstupu pri vyhľadávaní v obrázkoch a videách - pomocou textu, podobnosti obrázku alebo pomocou farebných nákresov. V praxi sú všetky tri metódy vstupu kombinované pre dosiahnutie čo najväčšej presnosti (Lokoč a kol., 2019).

1.1 Vyhľadávanie podľa textového dotazu

Jednou z možností vstupu pre vyhľadávací algoritmus je text, ktorý popisuje obsah obrázka. Pri využití tejto metódy vyhľadávania sú k obrázkom v databáze priradené texty, ktoré sú následne porovnávané so vstupom od užívateľa.

Problém vyhľadávania v obrázkoch pomocou textu sme takto previedli na problém vyhľadávania v texte. Nastávajú teda dve otázky: Ako získať text k obrázkom? Ako budeme porovnávať text od užívateľa s popisom obrázku?

Ak bol obrázok objavený na internete v rámci webovej stránky, text v okolí obrázku pomáha identifikovať, čo sa na obrázku nachádza. Ak k obrázku nie je k dispozícií relevantný text, je potrebné k nim popis vytvoriť - teda ich anotovať. Obrázky môžeme anotovať manuálne alebo automaticky.

1.1.1 Manuálna anotácia obrázkov

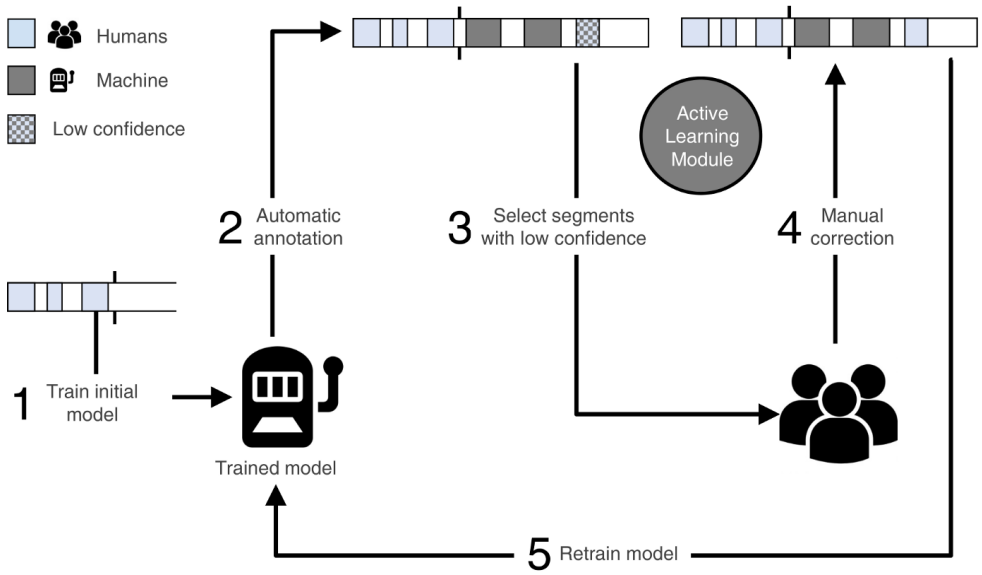
Manuálne anotácia obrázkov prebieha tak, že ľudia označia oblasť v obrázku a popíšu čo sa na nej nachádza. V prípade, že potrebujeme zabezpečiť vysokú presnosť anotácie, necháme viacerých ľudí anotovať ten istý obrázok a výsledok akceptujeme len v prípade vysokej zhody medzi anotátormi.

Obrázok 1.1: Príklad anotovaného obrázku. Obrázok z fotobanky Unsplash.

Existuje niekoľko softvérov, ktoré je možné použiť pre anotáciu. Pre príklad uvedieme LabelImg, VGG Image Annotation Tool (VIA), Computer Vision Annotation Tool (CVAT) a LabelMe. Taktiež existujú firmy, ktoré manuálnu anotáciu obrázkov zabezpečujú za poplatok - napríklad LabelBox alebo LionBridge.

1.1.2 Kooperatívna anotácia obrázkov

Manuálna anotácia obrázkov zabezpečuje vysokú presnosť. Pre svoju časovú alebo finančnú nákladnosť ju však nie je možné použiť vo veľkom rozsahu. V praxi je tak manuálna anotácia obrázkov využívaná najmä pre tvorbu dát, na ktorých sú modely pre automatickú anotáciu trénované. Efektívny spôsob využitia vo svojom článku predstavili Wagner a kol. (2018). Odporúčajú, aby boli manuálne anotované najmä obrázky, pri ktorých je spoľahlivosť automatickej anotácie najnižšia. Po anotácií obrázkov je model opätovne natrénovaný a opäť sú vyhodnotené obrázky s najnižšou spoľahlivosťou. Každá iterácia tohto procesu tak zlepšuje spoľahlivosť modelu pre automatickú anotáciu.



Obrázok 1.2: Priebeh kooperatívnej anotácie podľa Wagner a kol. (2018).

1.1.3 Automatická anotácia obrázkov

Pri využití automatickej anotácie obrázkov je na obrázky aplikovaný model, ktorý obrázkom priradí kľúčové slová alebo frázy bez asistencie človeka.

Najmodernejšie nástroje, ktoré boli prezentované na súťaži Video Browser Showndown pre automatickú anotáciu používajú konvolučné neurónové siete (Lokoč a kol., 2019). Ide o neurónovú sieť, ktorá používa konvolúciu na miesto maticového násobenia aspoň v jednej zo svojich vrstiev (Goodfellow a kol., 2016). Každá vrstva siete poskytuje abstraktnejšiu informáciu o obsahu zdrojového obrázka. Posledná vrstva siete priradí vstupnému obrázku relevantné kľúčové slová.

Konvolučné neurónové siete sú trénované na rozsiahlych databázach manuálne anotovaných obrázkov. Jednou z takých sietí je ImageNet, ktorá obsahuje viac než

14 miliónov obrázkov. Na súťaži ImageNet Large Scale Visual Recognition Challenge (ILSVRC) je každoročne vyhodnotený a porovnávaný výkon natrénovaných sietí.

Tvorcovia nástrojov pre vyhľadávanie vo videu tieto siete potom využijú ako základ, pričom zvolenú sieť pretrénujú na dataset na ktorom má dosahovať čo najlepšie výsledky. Používané sú napríklad siete AlexNet (Krizhevsky a kol., 2012) alebo GoogLeNet (Szegedy a kol., 2015). (Lokoč a kol., 2019)

1.1.4 Vyhľadávanie v texte

Keď máme k dispozícií textový dotaz a textové popisy obrázkov, môžeme pristúpiť k vyhľadávaniu v obrázkoch - teda zoraďovaniu obrázkov podľa relevancie voči dotazu. Pred samotným použitím modelu pre vyhľadávanie v texte je dotaz pred-spracovaný. Typické predspracovanie dotazu má nasledovné kroky (Long a Chang, 2014):

1. **Rozšírenie dotazu** - ku každému slovu sú pridané jeho synonymá.
2. **Normalizácia slov** - každé slovo je prevedené na základný tvar.
3. **Odstránenie nepotrebných slov** - z dotazu sú odstránené slová, ktoré nesú málo významu (napr. „a“, „alebo“).
4. **Rozdelenie na výrazy** - slová z dotazu sú spojené do výrazov (napríklad dotaz „žena pije vodu“ obsahuje výrazy „žena pije“, „pije vodu“ a „žena pije vodu“).
5. **Gramatická analýza** - identifikácia vetného základu, predmetu, podmetu, prívlastku a podobne.

Po pred-spracovaní dotazu sa realizuje vyhľadávanie pomocou vyhľadávacieho modelu. Modely môžeme podľa Sharma a Patel (2013) rozdeliť na tri skupiny:

- **logické modely** - Vyhodnocujú prítomnosť dokumentu v množine výsledkov. Umožňujú definovať dotazy pomocou operátorov AND a OR. Ich nevýhodou však je, že neumožňujú zoraďovať výsledky podľa relevancie. Patrí sem napríklad Boolovský model (Boo, 1974) alebo rozšírený Boolovský model (Salton a kol., 1983).
- **vektorové modely** - Vyhodnocujú podobnosť medzi dotazom a dokumentami. Patrí sem napríklad Vektorovo-priestorový model (Salton a kol., 1975) alebo neurónové siete Baeza-Yates a Ribeiro-Neto (1999).
- **pravdepodobnostné modely** - Vyhodnocujú pravdepodobnosť, že užívateľ hľadá daný dokument. Patrí sem napríklad model binárnej nezávislosti (Roelleke a Wang, 2007) alebo jazykové modely (Lv a Zhai, 2009).

Podrobný popis vyhľadávacích modelov je možné nájsť v dizertačnej práci Kraaij (2005).

1.2 Vyhľadávanie podľa podobného obrázku

Ak užívateľ z nejakého zdroja získal obrázok, ktorý sa podobá na hľadaný obrázok, môže ho použiť ako vstup vyhľadávacieho algoritmu. Obrázky môžu byť podobné buď vo významovej rovine (obsahujú podobné objekty) alebo vo vizuálnej rovine.

Ak vyhľadávame obrázky podobné **vo významovej rovine**, pre identifikáciu objektov použijeme rovnaké neurónové siete ako pri automatickej anotácií obrázkov (viď. kapitolu 1.1.3). Z predposlednej vrstvy siete, ktorá reprezentuje obrázky vo vysoko dimenzionálnom priestore (prípadne konceptu) sa vytvorí vektor. Vzájomnú vzdialenosť týchto vektorov (podľa Manhatanskej, kosínovej, Euklidovskej alebo inej metriky) považujeme za vzdialenosť obrázkov. (Donahue a kol., 2014)

Ak vyhľadávame obrázky podobné **vo farebnej rovine**, môžeme výsledky zoradiť podľa priemernej vzdialenosti farby pixelov obrázku z databázy a obrázku z dotazu. Vzdialenosti farieb je možné počítať ako Euklidovskú vzdialenosť v priestore RGB. Na základe ľudského vnímania farieb však predpokladáme, že porovnanie farieb v priestore Lab bude dosahovať lepšie výsledky (viď. kapitoly 2.1 a 2.2.2). Pri vyhľadávaní nad veľkými databázami by však porovnávanie každého obrázku s dotazom trvalo príliš dlho. Používajú sa preto rôzne spôsoby indexovania obrázkov podľa toho, aké farby obsahujú. Jedna z metód každému obrázku je vypočíta histogram jeho farebnosti a obrázok je zaradí do príslušného clustera. Pri vyhľadávaní je počítaná vzdialenosť len k obrázkom v clusteri, do ktorého spadá obrázok z dotazu.

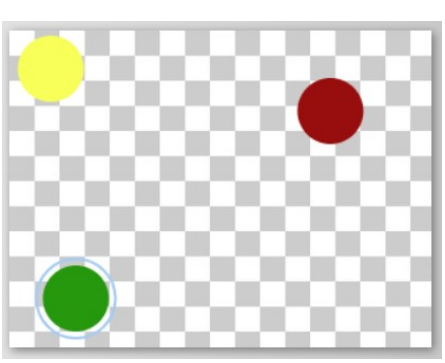
Vyhľadávanie na základe podobnosti obrázaka umožňuje aj Google umožňuje na adrese `https://www.google.com/imghp`, po kliknutí na ikonku fotoaparátu.

1.3 Vyhľadávanie podľa farebného nákresu

Vstupom pre vyhľadávanie je nákres obrázku. Detailnosť nákresu sa líši podľa použitého algoritmu. Niektoré algoritmy umožňujú nakresliť obrázok do detailov, aké umožňuje dosiahnuť program *Skicár*.

Predmetom tejto práce je vyhľadávanie v obrázkoch na základe farebnej zhody s jednoduchým nákresom od užívateľa. Užívateľ má možnosť umiestniť na plátno 3 až 5 farebných krúžkov. Pozíciu krúžkov na plátne považujeme za smerodajnú, ale tvar objektov pre vyhľadávanie nepoužívame.

Širší popis vyhľadávania podľa farebného nákresu je v kapitole 1.3.



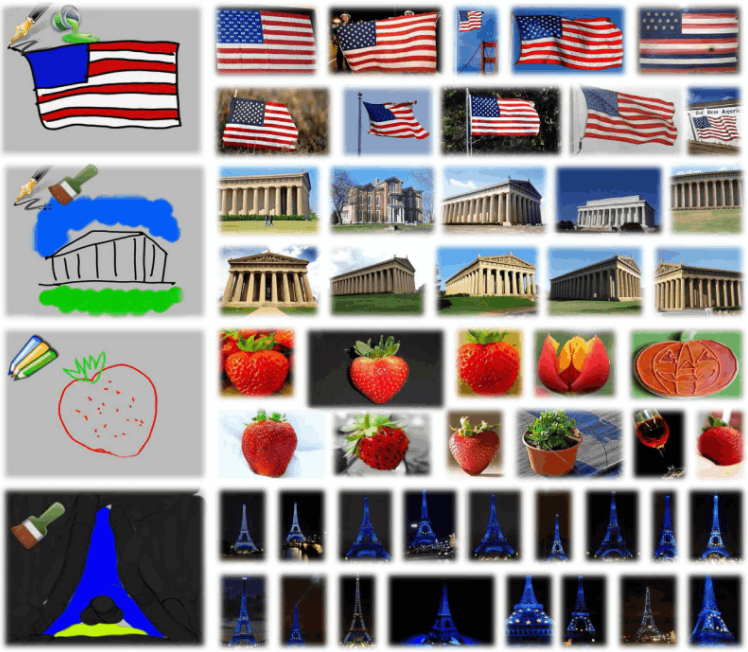
Obrázok 1.3: Krúžky, ktoré užívateľ umiestnil na plátno.

Pre urýchlenie vyhľadávania sú snímky v databáze pred vyhľadávaním pred-spracované. Z obrázkov sa vytvoria farebné signatúry, ktoré poskytujú informácie o farbách na obrázku a umožňujú tak rýchlejšie vyhodnotiť relevanciu obrázka. Jedna z jednoduchých metód pre vytvorenie signatúr je zmenšenie obrázka na veľkosť napríklad 20x10 pixelov. Radikálne sa tak znižuje počet pixelov, ktorý je potrebné pri vyhľadávaní vyhodnocovať, ale stále máme k dispozícií informáciu o dominantných farbách na obrázku a informáciu o ich umiestnení. V kapitole 3 vyhodnocujeme, ktorý algoritmus pre zmenšenie obrázka je pre tento účel najvhodnejší.

Ďalšiu možnosť pred-spracovania obrázkov popísali Leibetseder a kol. (2018). Ich systém vypočítava farebný histogram pre každú časť obrázka a výsledky poznamenáva do indexu, ktorý je použitý pri vyhľadávaní.

1.3.1 Porovnanie so súvisiacimi prácami

Sun a kol. (2013) umožnili účastníkom svojho výskumu nakresliť detailné dotazy pomocou nástrojov pero, štetec, ceruzka a výplň farbou.



Obrázok 1.4: Ukážka dotazov a výsledkov vyhľadávania pomocou nástroja MagicBrush (Sun a kol., 2013)

Z nášho pohľadu však niektorí ľudia nemajú potrebné zručnosti a trpezlivosť na kresbu detailných nákresov. Rozhodli sme sa preto skúmať vyhľadávanie pomocou jednoduchších nákresov.

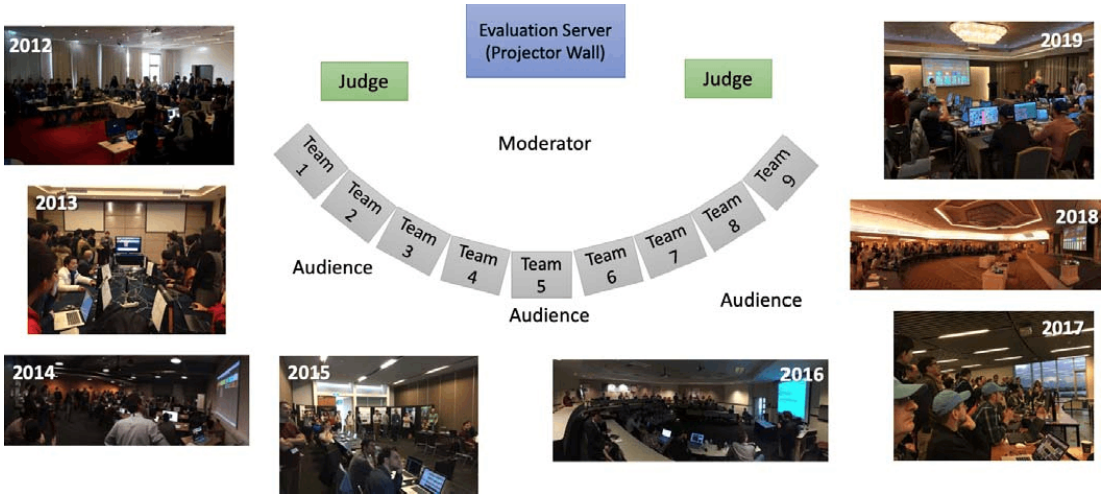
Lokoč a kol. (2017) sa vo svojej práci tiež zaoberali vyhľadávaním v obrázkoch pomocou jednoduchých farebných skíc. Vo svojej práci testovali formálny rámec pre vyhľadávanie pomocou jednoduchých farebných skíc. Pomocou rôznych typov simulovaných dotazov analyzovali efektivitu vyhľadávacieho modelu. Naša práca sa zameriava na porovnávanie rôznych parametrov vyhľadávacieho modelu s využitím dát získaných od reálnych užívateľov.

Lokoč, Kovalčík a Souček (2018) popisujú svoj nástroj na vyhľadávanie vo videu, ktorý v roku 2018 vyhral súťaž Video Browser Showdown.

1.4 Vyhodnocovanie nástrojov pre interaktívne vyhľadávanie vo videu

Medzinárodná súťaž *Video Browser Showdown*, ktorej sa zúčastňujú výskumné tímy z celého sveta, sa snaží o vyhodnotenie efektivity najmodernejších nástrojov na riešenie tohto problému.

Počas súťaže tvorcovia používajú vlastné nástroje, aby riešili úlohy zadávané od usporiadateľov. Pre hľadanie známej scény majú úlohy buď formu textového popisu (napríklad: „Žena, ktorá pije vodu zo skleného pohára“, alebo je súťažiacim premietnutý 20 sekundový záber z videa, ktoré majú nájsť.



Obrázok 1.5: Typické rozloženie na súťaži Video Browser Showdown. Každý tím používa vlastný vyhľadávací nástroj. Na projekcií sa zobrazujú zadania úloh a priebežné skóre tímov. (Schoeffmann, 2019)

Na Karlovej univerzite je vyvíjaný nástroj VIRET, ktorý súťaž VBS v roku 2018 vyhral (Lokoč, Kovalčík a Souček, 2018). Jedným zo vstupov vyhľadávacieho nástroja VIRET je jednoduchý farebný nákres.

V tejto práci sa venujeme zberu a vyhodnoteniu dát o tom ako užívatelia vnímajú farby. Zozbierané dáta následne využijeme na vyhodnotenie efektivity algoritmov, ktoré vyhľadávajú pomocou jednoduchých farebných nákresov. Tento typ vyhľadávania má na súťaži *Video Browser Showdown* využitie keď súťažiaci hľadajú 20 sekundový záber videa, ktorý im bol premietnutý.

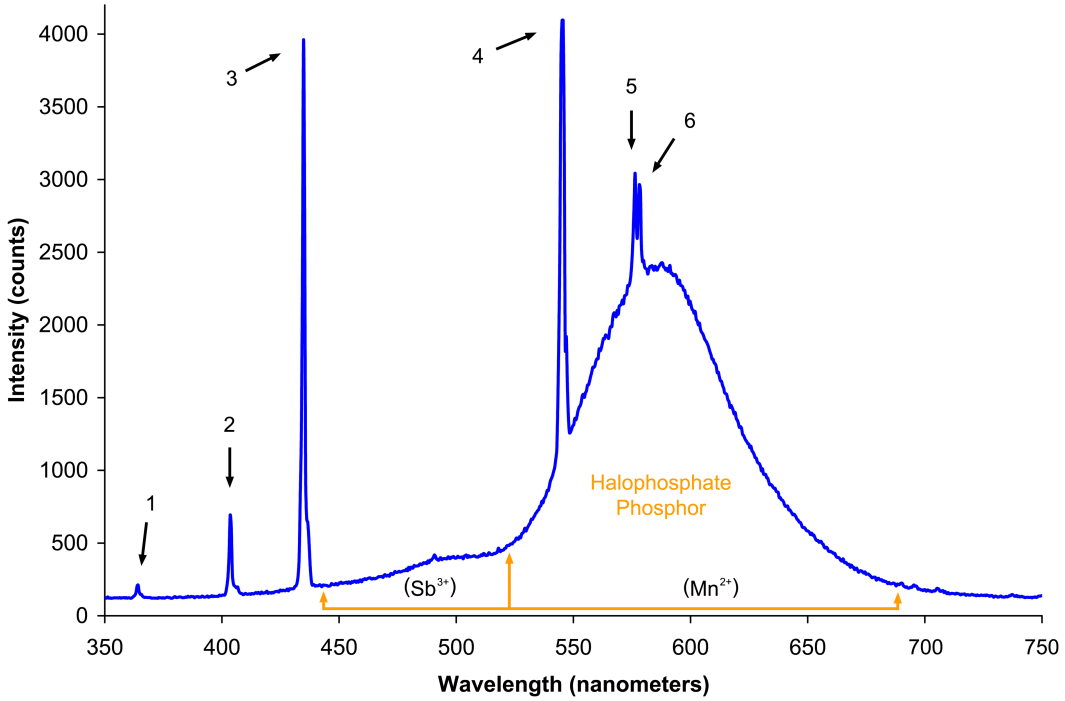
Výskum popísaný v tejto práci mal za cieľ pomôcť tvorcom nástroja VIRET navrhnúť vhodný vyhľadávací algoritmus pre vyhľadávanie pomocou farebných nákresov.

2. Vybrané pojmy z teórie farieb

Pre lepšie pochopenie problematiky si objasníme niektoré dôležité pojmy. Tieto pojmy vo svojej knihe opísal Nassau (1998).

Viditeľné svetlo je elektromagnetické žiarenie, ktoré sme schopní vnímať očami. Toto žiarenie môže mať vlnovú dĺžku od 380 do 780 nm.

V každom momente vstupuje do oka žiarenie rôznych vlnových dĺžok, ktoré vyvolá vnemy na sietnici, ktoré potom po transformácií mozog vníma ako **farbu**.



Obrázok 2.1: Svetlo rôznych vlnových dĺžok vyžarované žiarivkou. Krivka označuje, akú intenzitu žiarenia žiarivka vyžaruje v danej vlnovej dĺžke (Deglr6328, 2006).

Ako je možné, že vidíme hnedý stôl? Zdroj svetla vyžaruje elektromagnetické žiarenie rôznych vlnových dĺžok, ktoré dopadnú na objekt. Objekt niektoré z týchto vlnových dĺžok pohltí a niektoré odrazí. Vlny, ktoré boli odrazené, dopadnú do oka. Všetky vlny, ktoré dopadli na jedno miesto na sietnici, spoločne určia farbu, ktorú na danom mieste vnímame.

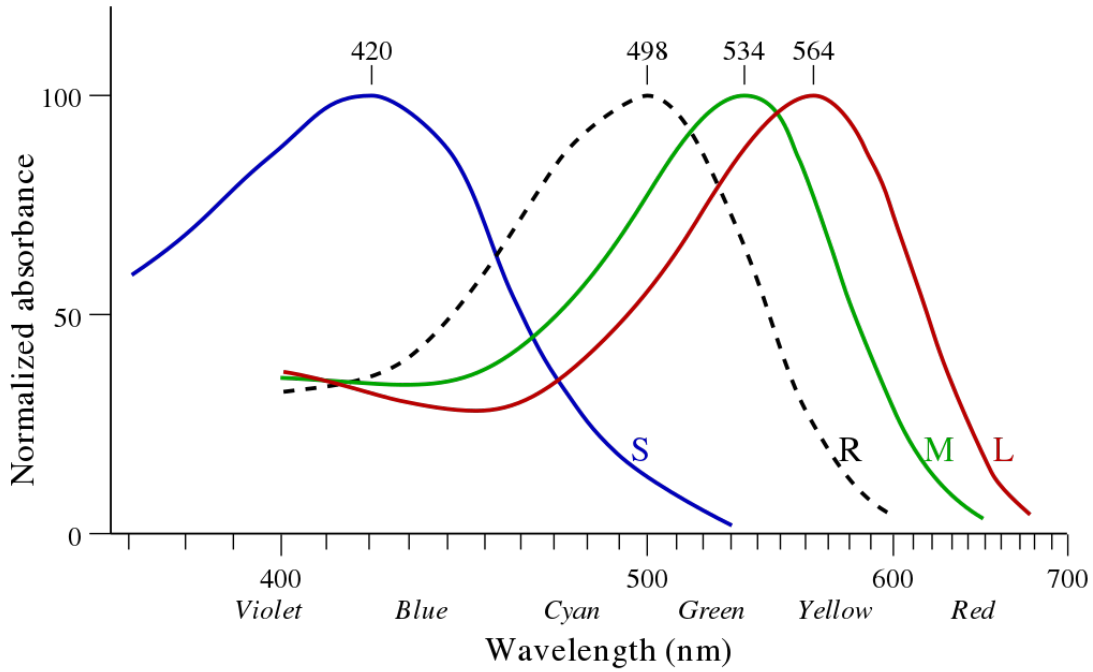
Keď sa obzrieme po miestnosti, môžeme vidieť objekty rôznych farieb - napríklad hnedú stoličku, bielu stenu alebo zelenú rastlinu. Objekty môžeme vidieť vďaka svetlu, ktoré na ne dopadá. Čierne objekty väčšinu svetelných lúčov pohlcujú, biele objekty odrážajú vlny rôznych vlnových dĺžok naprieč spektrom.

2.1 Ako vnímame farby

Elektromagnetické vlny rôznych vlnových dĺžok dopadajú na sietnicu. Sietnica obsahuje 4 typy zmyslových buniek:

- čapíky citlivé na červené svetlo (L)
- čapíky citlivé na zelené svetlo (M)
- čapíky citlivé na modré svetlo (S)
- tyčinky rozlišujúce odtiene sivej (R)

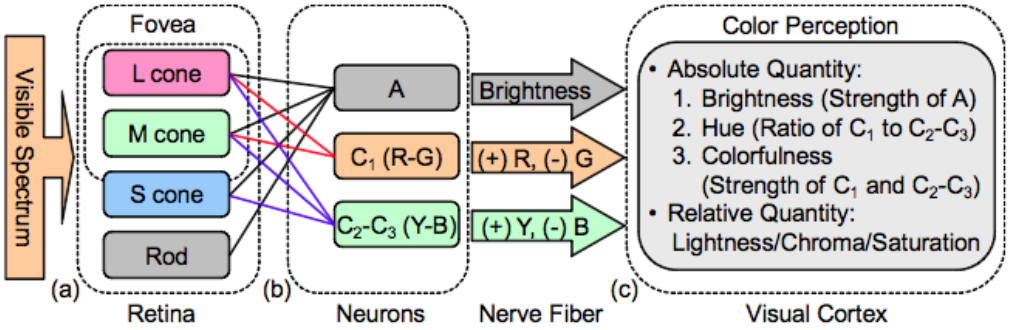
Za uskutočnenie experimentov, ktoré ukázali rozdiely v citlivosti zmyslových buniek, bola v roku 1967 výskumníkom udelená Nobelova cena. (Bowmaker a Dartnall, 1980) Výsledky experimentov môžeme vidieť na nasledujúcom grafe.



Obrázok 2.2: Graf zobrazuje citlivosť zmyslových buniek na svetlo danej vlnovej dĺžky. Krivky L, M a S označujú citlivosť troch druhov čapíkov, krivka R označuje citlivosť tyčiniek. (J.K. a H.J.A., 1980).

Ďalším dôležitým bodom pri spracovaní svetla sú gangliové bunky. Sú to neuróny, ktoré prijímajú (čiastočne spracovaný) signál z tyčiniek a čapíkov. V týchto neurónoch sú signály z rôznych typov čapíkov od seba odčítané a zvyšok zrakovej dráhy pracuje s rozdielmi vnímaných farieb namiesto samostatných signálov. Schému zapojenia môžeme vidieť na obrázku.

Vnímanie farieb vo forme rozdielov prvý krát popísal Ewald Hering, prvý rektor Karlovej univerzity, v roku 1892. (der Wissenschaften in Wien., 1872) Táto teória sa nazýva *Opponent color theory* a je dnes považovaná za pravdivú.



Obrázok 2.3: Schéma zobrazuje transformáciu svetla na chemický signál, ktorý putuje do mozgu. Signál smeruje od zmyslových buniek na sietnici (Retina) ku gangliovým bunky (Neurons) a potom vo forme rozdielov do mozgu (Visual Cortex). (Googolplexbyte, 2013)

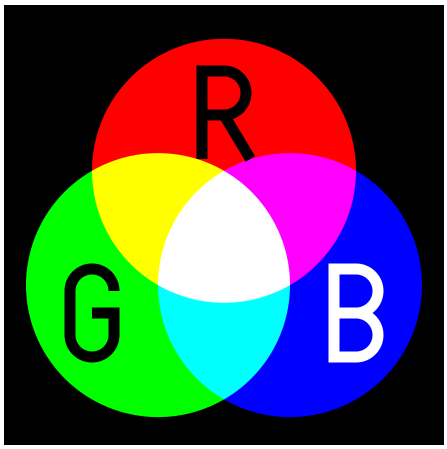
2.2 Modely pre reprezentáciu farieb v počítači

2.2.1 Farebný model RGB

Na základe vedomostí o tom, že ľudské oko vníma farby cez 3 druhy receptorov, vznikol model RGB.

Na zobrazenie farby **použijeme 3 zložky svetla - červené svetlo (Red), zelené svetlo (Green) a modré svetlo (Blue)**, každé vo vhodnej intenzite. Každý z týchto farebných zdrojov vzbudí naše receptory do takej miery, aby vyvolali vnem požadovanej farby.

RGB je aditívny farebný model - to znamená, že bielu farbu dosiahneme zmiešaním všetkých základných farieb.



Obrázok 2.4: Vizualizácia aditívneho miešania farieb. Kombinácia červeného, zeleného a modrého svetelného zdroja vytvorí bielu farbu. Kombináciou dvoch zdrojov v plnej intenzite dosiahneme žltú, fialovú alebo tyrkysovú farbu. (Horvath, 2006)

Zariadenia ako fotoaparáty, kamery, televízory a monitory dnes zachytávajú, prípadne vyžarujú červenú, zelenú a modrú farbu samostatne a preto je tento farebný model dnes najpoužívanejší.

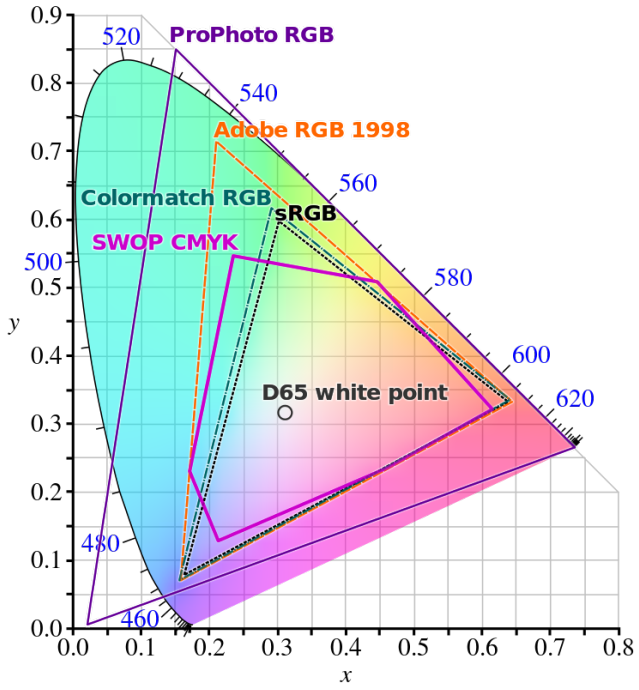
Farebné priestory RGB

Model RGB nedefinuje presnú vlnovú dĺžku červeného, zeleného a modrého svetla - tú definujú farebné priestory. Keďže všetky farby sú generované kombináciou týchto troch farieb, výber základných farieb svetla určuje, aký rozsah farieb vieme zobraziť.

V prípade, že farebný priestor nešpecifikujeme, predpokladá sa, že používame farebný priestor **sRGB**. Tento priestor vznikol v roku 1996 ako výsledok spolupráce spoločností *Microsoft* a *Hewlett Packard*. (iec, 1999)

Pre niektoré aplikácie nie je farebný priestor sRGB vhodný. Platí to napríklad pri tlači - pomocou farieb používaných v tlačiarňach je možné vytlačiť farby mimo rozsah sRGB. Aby bolo možné pracovať s čo najväčším množstvom vytlačiteľných farieb v RGB modeli, spoločnosť *Adobe Systems, Inc* v roku 1998 navrhla farebný priestor **Adobe RGB** (ado, 2005).

Ďalšie populárne RGB priestory zahŕňajú **CIE (1931) RGB**, **ISO RGB**, **ProPhoto RGB**.



Obrázok 2.5: Porovnanie RGB a CMYK (BenRG, 2014).

2.2.2 Farebný model CIE Lab

Nevýhodou modelu RGB je, že Euklidovská vzdialenosť farieb v RGB priestore sa nezhoduje s tým, aké rozdiely ľudia vo farbách vnímajú. Farebný model CIE Lab bol naopak navrhnutý tak, aby sa ľuďmi vnímaný rozdiel v dvoch farbách dal aproximovať Euklidovskou vzdialenosťou. Model zahŕňa zložky:

- L označuje celkový jas
- a označuje červeno-zelenú zložku

- b označuje žlto-modrú zložku

Typická implementácia modelu má v zložke L rozsah od 0 pre čiernu farbu až po 100 pre bielu farbu a v zložkách a,b rozsah od -128 po +127. Kladné hodnoty zložky a označujú výskyt červenej farby a záporné hodnoty označujú výskyt zelenej farby. Kladné hodnoty zložky b označujú výskyt žltej farby a záporné hodnoty označujú výskyt modrej farby.

Po prvýkrát tento model popísal Richard Sewall Hunter v roku 1948 (col, 1948). Neskôr *Medzinárodná komisia pre osvetľovanie* popísala novšiu verziu tohto modelu nazývanú *CIE 1976 L*, a*, b**, ktorá je používaná dodnes. Vo zvyšku práce budeme tento model nazývať Lab.

2.3 Záver kapitoly

V tejto kapitole sme ukázali, podstatu farebných priestorov RGB a Lab. Vysvetlili sme, že priestor RGB má základ v tom, ako sú farby snímané a vytvárané a priestor Lab má základ v tom, ako sú farby vnímané ľudským okom. Tieto skutočnosti poukazujú na to, že použitie farebného priestoru Lab je pre vyhľadávanie vo videu vhodnejšie. V experimente zmeriame, či rozdiel v presnosti týchto farebných priestorov je podstatný alebo zanedbateľný a pokúsime sa teda zodpovedať otázku, či má význam obrázky pred vyhľadávaním konvertovať z RGB do Lab.

3. Vyhľadávanie pomocou jednoduchej skici

Cieľom experimentu, ktorý je predmetom tejto práce, bolo zistiť, ako dobre sú užívatelia schopní zapamätať si farby na obrázku. V tomto experimente mali ľudia možnosť zadať farby tak, že nakreslili 3 až 5 farebných krúžkov na plátno.

3.1 Databáza obrázkov

Databáza obrázkov, s ktorými pracujeme, sú snímky z videí z databáze *Internet Archive videos (IACC.3)* od organizátorov TRECVID (tre). Táto databáza je dostupná pod licenciou Creative Commons. Ide prevažne o záznamy z televízneho vysielania.

Z videí sme exportovali snímok každých 200 sekúnd. Keďže videá majú dĺžku od 6 do 9 a pol minúty, získali sme 2 až 3 snímky z každého videa. Takýmto postupom sme získali 13 222 obrázkov.

3.2 Zber dát

Zber dát bol realizovaný pomocou webovej hry. Užívatelia boli oslovení s prosbou o pomoc s tvorbou a zberom jednoduchých farebných nákresov pre potreby tejto bakalárskej práce. Hry sa zúčastnili priatelia autora tejto bakalárskej práce a neznámi užívatelia, ku ktorým sa hra dostala prostredníctvom sociálnej siete Facebook. Užívateľom sme ukázali obrázok na dobu 7 sekúnd a potom sme ich požiadali, aby čo najpresnejšie zadali farby, ktoré videli. Autorom aplikácie pre zber dát je Artur Finger, ktorému týmto ďakujem za umožnenie použitia a upravenia jeho diela.

Pre tento experiment sme z databázy vybrali 100 obrázkov tak, aby obsahovali čo najviac farieb z RGB spektra. Zoznam použitých obrázkov je v prílohe A.1.

Použité údaje

Ku každému nákresu sme nakoniec využili nasledovné údaje:

- vygenerované ID užívateľa
- čas odoslania nákresu
- obrázok, ku ktorému nákres patrí
- rozlíšenie obrázka
- skóre zobrazené užívateľovi
- poradie kresleného obrázku - koľký obrázok užívateľ kreslí (v tejto relácií)
- farba a pozícia krúžkov, ktoré užívateľ zadal

- aké zmeny vo farbe a pozícií krúžkov užívateľ vykonával

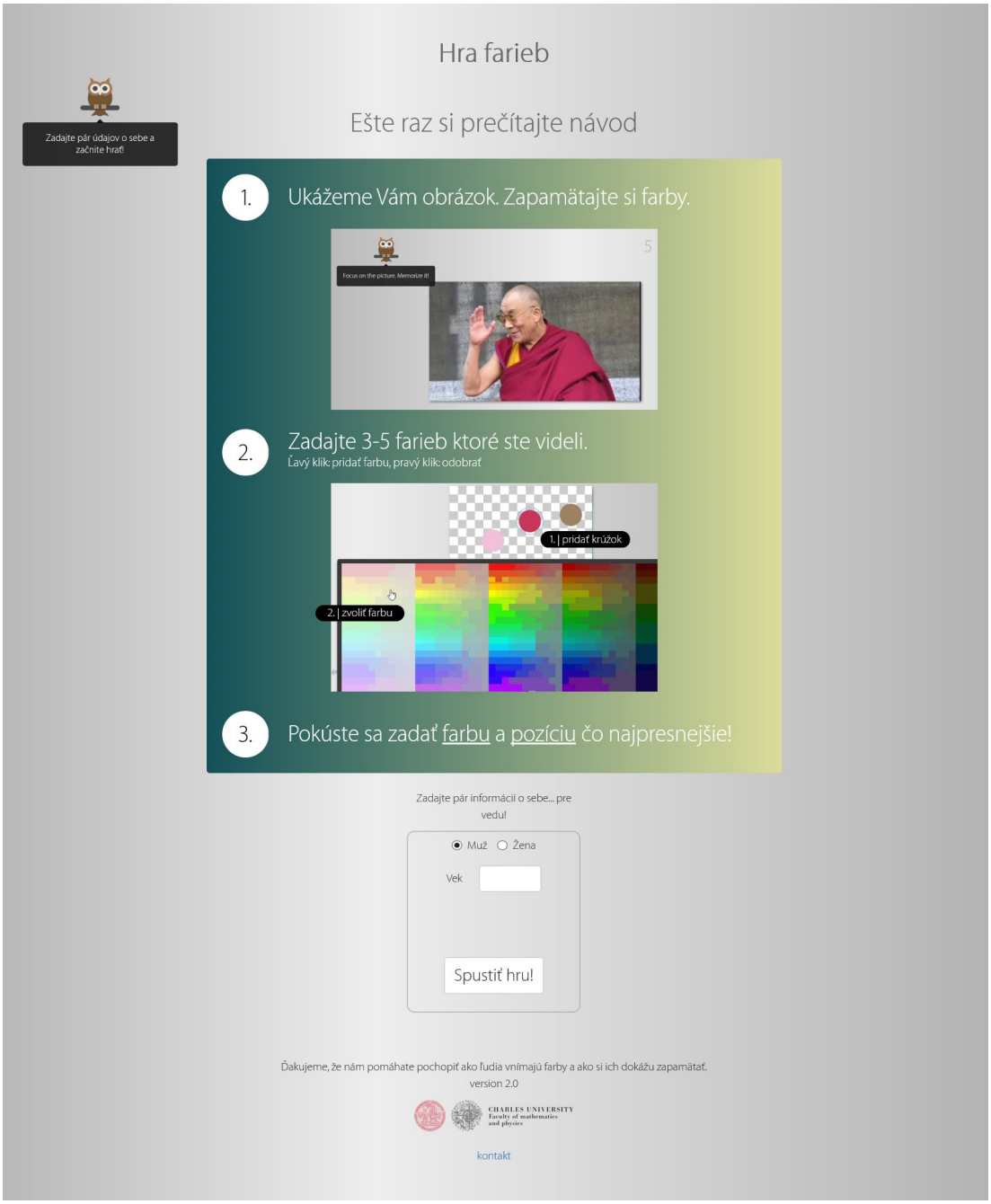
Aby sme predišli zbytočnej práci nad nerelevantnými nákresmi, pri ktorých užívatelia hru len testovali, pri vyhodnocovaní sme uvažovali len nákresy z relácií, v ktorých bol zadaný vek užívateľa a boli ohodnotené aspoň 3 obrázky.

Získali sme tak 2503 nákresov s 10138 krúžkami.

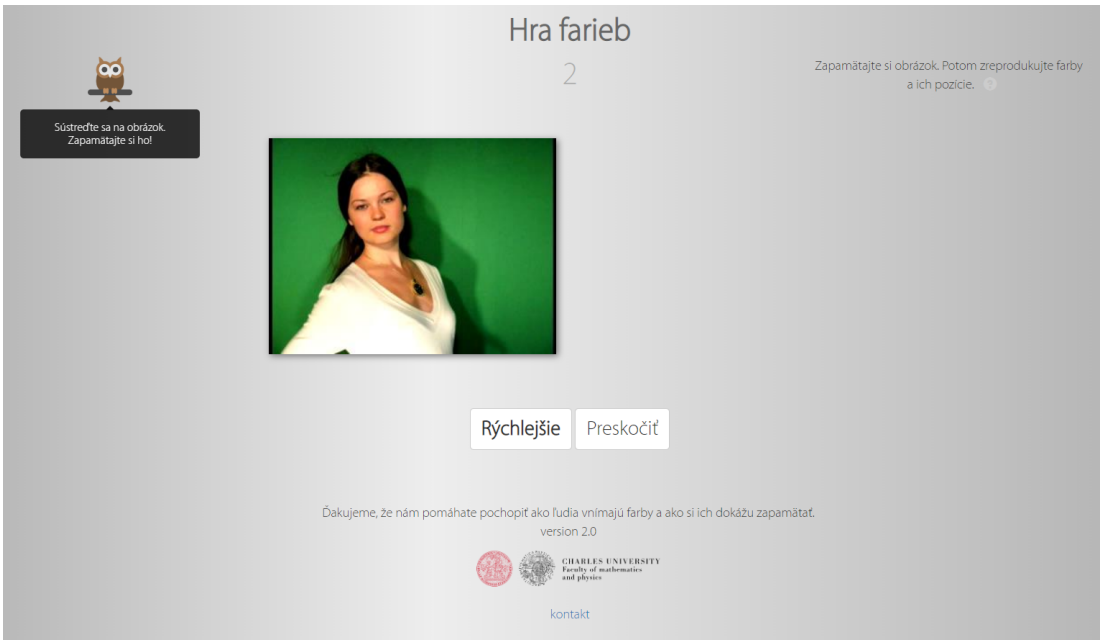
Priebeh hry

Po príchode na webovú stránku hry sa užívateľovi zobrazili dve stránky inštrukcií (obrázky 3.1 a 3.2). Potom sme užívateľovi na 7 sekúnd zobrazili obrázok a požiadali ho o vytvorenie nákresu s 3 až 5 farbami.

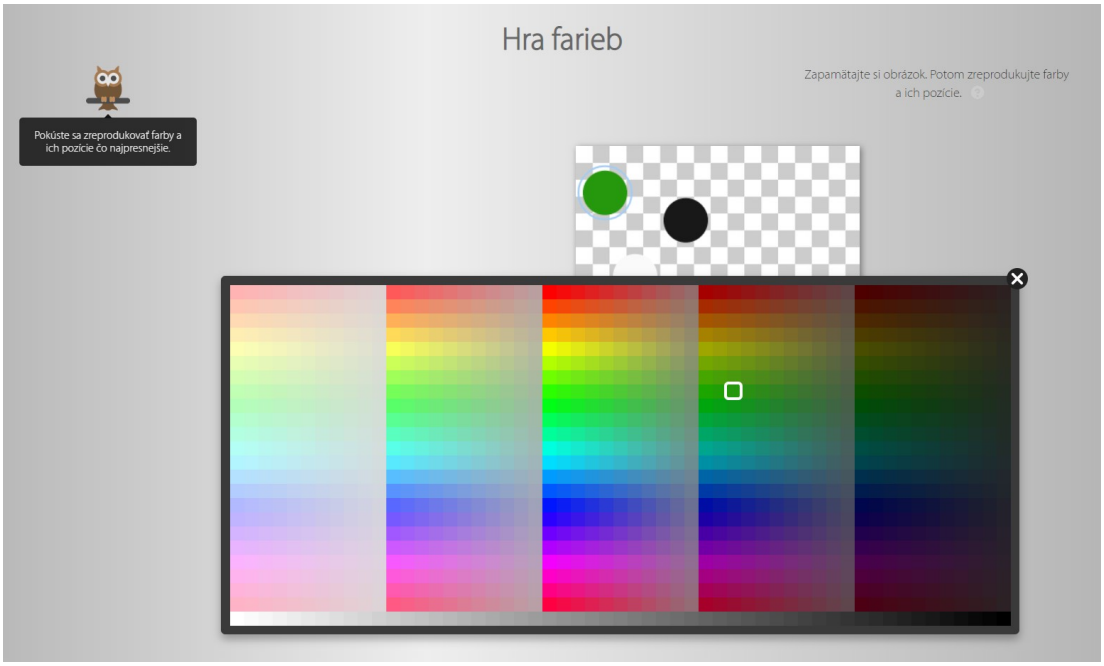
Obrázok 3.1: Prvá stránka, ktorá bola užívateľom zobrazená po príchode na webovú stránku hry. Po kliknutí na tlačítko „Spustiť hru“ im bola zobrazená stránka s detailnejšími inštrukciami (viď. obrázok 3.2).



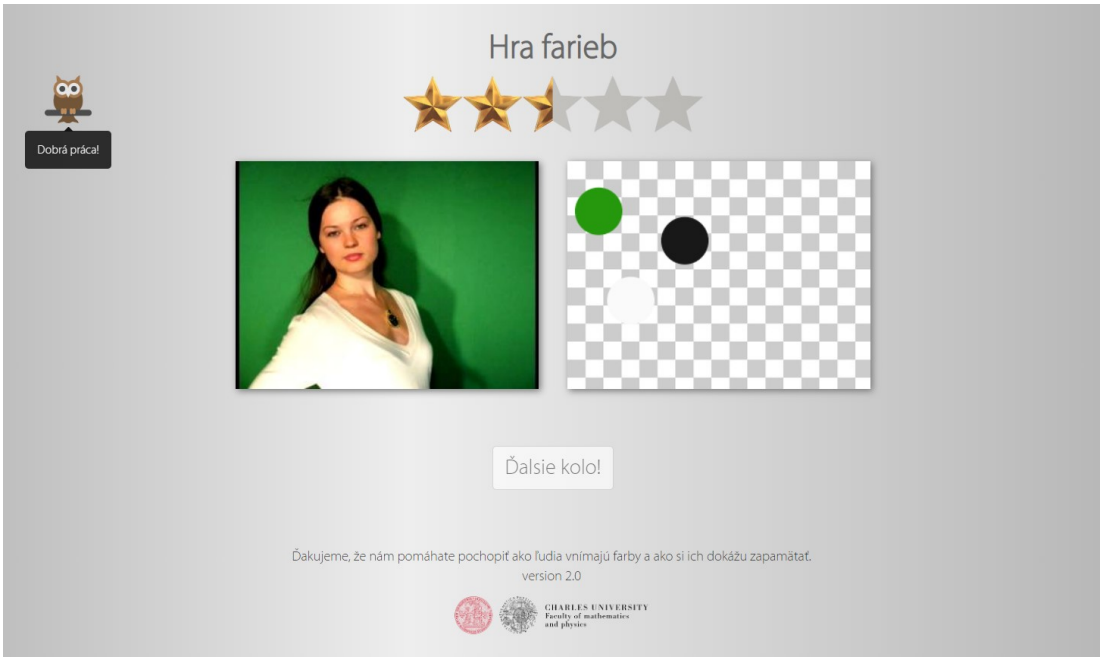
Obrázok 3.2: Stránka s detailnejšími inštrukciami. Od užívateľov sme požadovali zadanie veku.



Obrázok 3.3: Zobrazenie obrázku, ktorý si má užívateľ zapamätať. Kliknutím na tlačidlo „Rýchlejšie“ sa užívateľ dostal k plátnu. Po uplynutí 7 sekúnd sa obrázok skryl a plátno sa zobrazilo automaticky.



Obrázok 3.4: Užívateľom bolo zobrazené plátno, na ktoré mohli zadať 3 až 5 farebných krúžkov. Keď klikli na plátno, na danom mieste sa objavil krúžok a užívateľom sa ponúkla paleta na výber farby krúžku.

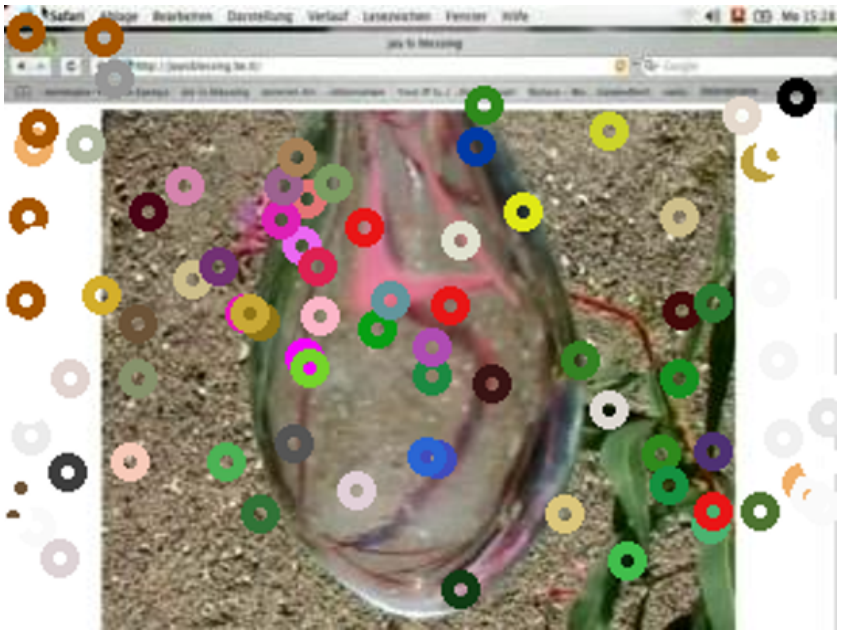


Obrázok 3.5: Po odoslaní sa užívateľom zobrazilo porovnanie ich nákresu s obrázkom. Po niekoľkých sekundách sa užívateľom zobrazilo hodnotenie.

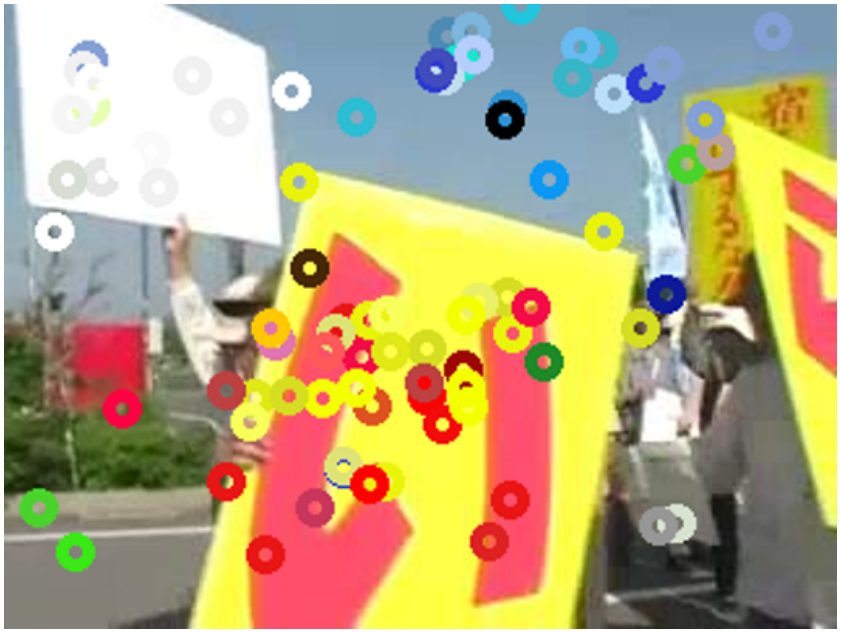
Obrázok 3.6: Hodnotenie nákresu. Obsahuje celkové skóre (zhoda farby krúžku s farbou obrázka na danom mieste) a porovnanie s ostatnými hráčmi. Kliknutím na tlačítko „Ďalšie kolo!“ sa užívateľovi zobrazil ďalší obrázok (viď. obrázok 3.3).

3.3 Vizualizácia zozbieraných dát

Na nasledujúcich obrázkoch sú zobrazené všetky krúžky zozbierané k danému obrázku. Číslo obrázka je uvedené vo formáte *Video ID-Frame ID* z databázy IACC.3 (viď. 3.1).



Obrázok 3.7: Zozbierané krúžky pre obrázok 35915-5076. Hnedé krúžky v ľavom hornom rohu naznačujú, že časť užívateľov si nezapamätala, že na obrázku je okrem pôdy a kvapky aj webový prehliadač.



Obrázok 3.8: Zozbierané krúžky pre obrázok 38342-6084.

3.4 Vyhodnotenie presnosti užívateľov

3.4.1 Určenie metriky

Pre vyhodnotenie nakoľko presne sa užívatelia trafili do farby a do jej umiestnenia potrebujeme porovnať krúžky, ktoré nakreslili, s ich správnym umiestnením. Nevieme, ktorý objekt z obrázka mal užívateľ v skutočnosti na mysli, ale môžeme to odhadnúť pomocou najbližšieho miesta kam sa krúžok „hodí“.

Budeme pracovať s metrikou f_d, ktorá vráti vzdialenosť krúžku od konkrétnemu pixelu na obrázku. f_d vracia Euklidovskú vzdialenosť v priestore s dimenziami L, a, b, X, Y, kde dimenzie L, a, b označujú zložky farby vo farebnom priestore Lab (viď. 2.2.2) a X, Y označujú umiestnenie na obrázku. Pre konverziu obrázkov z RGB do Lab budeme predpokladať, že obrázky sú uložené vo farebnom priestore sRGB. Zvažovať budeme iba pixely, ktoré sa nachádzajú do 70px od umiestnenia krúžku.

Ku každému krúžku priradíme pixel, pre ktorý f_d nadobúda minimum. Tento pixel budeme ďalej nazývať *cieľový*. Farbu a pozíciu tohto pixelu budeme považovať za správnu a voči nej budeme vyhodnocovať „chyby“.

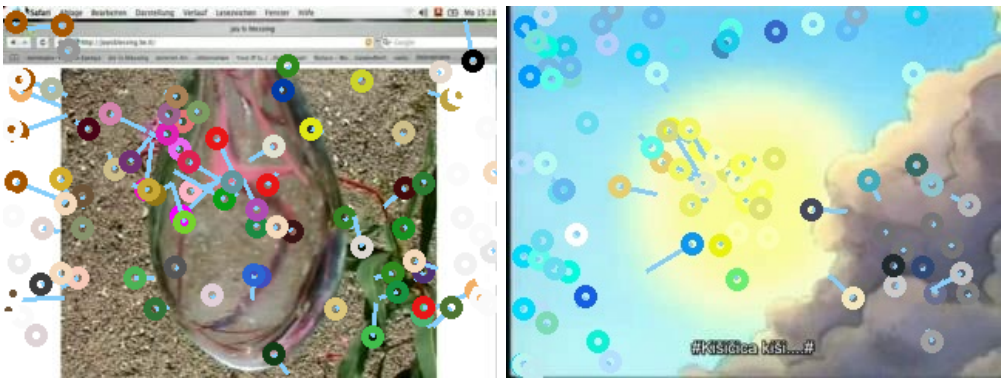
Nasledovná funkcia ukazuje princíp hľadania cieľového pixelu, kde obrázok i je reprezentovaný poľom 5-dimenzionálnych bodov typu ColorPoint, čo je štruktúra so zložkami X, Y pre polohu a zložkami L, a, b pre určenie farby. Krúžok z dotazu je tiež 5-dimenzionálny objekt typu ColorPoint. Prehľadávame pixely, ktoré sú vzdialené od krúžku z dotazu maximálne SearchRadius pixelov.

```
function FindTarget(colorPoint[] i, colorPoint q)
    closestPointDistance ← infinity
    closeImagePoints ← imagePoints within SearchRadius from [q.X, q.Y]
    for colorPoint in closeImagePoints do
        if Distance5D(q, colorPoint) < closestPointDistance then
            closestPoint ← colorPoint
```

$closestPointDistance \leftarrow Distance(q, colorPoint)$
end if
end for
return closestPoint
end function

function DISTANCE5D(ColorPoint c, ColorPoint p)
$X \leftarrow c.X$
$Y \leftarrow c.Y$
$L \leftarrow c.L$
$a \leftarrow c.a$
$b \leftarrow c.b$
return $\sqrt{(X - p.X)^2 + (Y - p.Y)^2 + (L - p.L)^2 + (a - p.a)^2 + (b - p.b)^2}$
end function

3.4.2 Ukážka cieľových pixelov priradených ku krúžkom



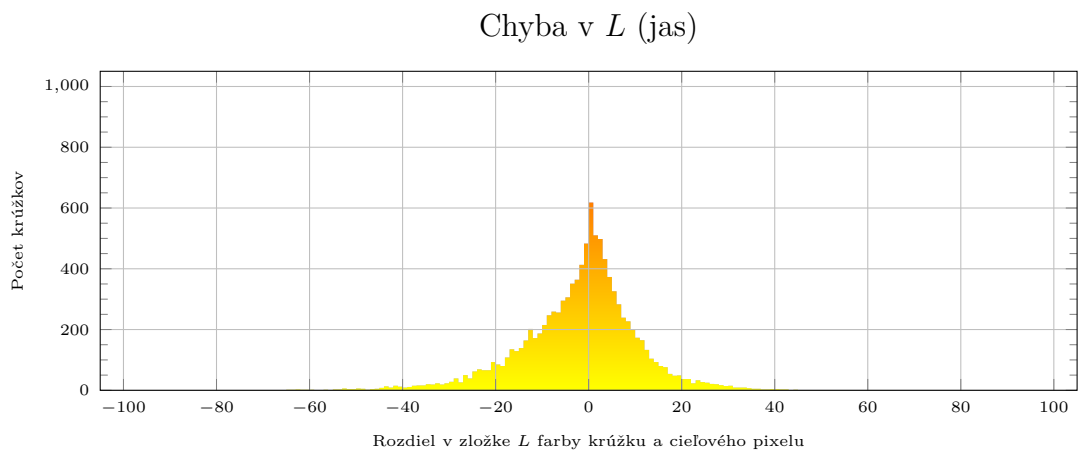
Obrázok 3.9: Na obrázku 35915-5076 a 39036-10077 sú zobrazené všetky zozbierané krúžky. Od každého krúžku vedie svetlomodrá čiara k jeho cieľovému pixelu pre SearchRadius=70px.

3.4.3 Hodnotenie výsledkov

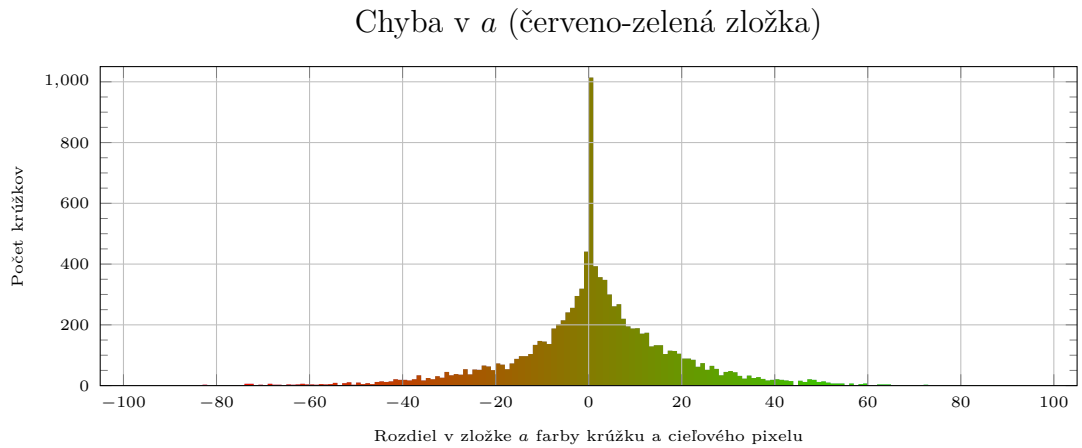
Chyba vo farbe aj v pozícií je prekvapivo malá. Až 43% krúžkov bolo umiestnených dokonale presne - s chybou 0. 40% krúžkov má nulovú chybu v červenozelenej zložke.

3.4.4 Presnosť vo farbe v Lab, RGB a v pozícii krúžku

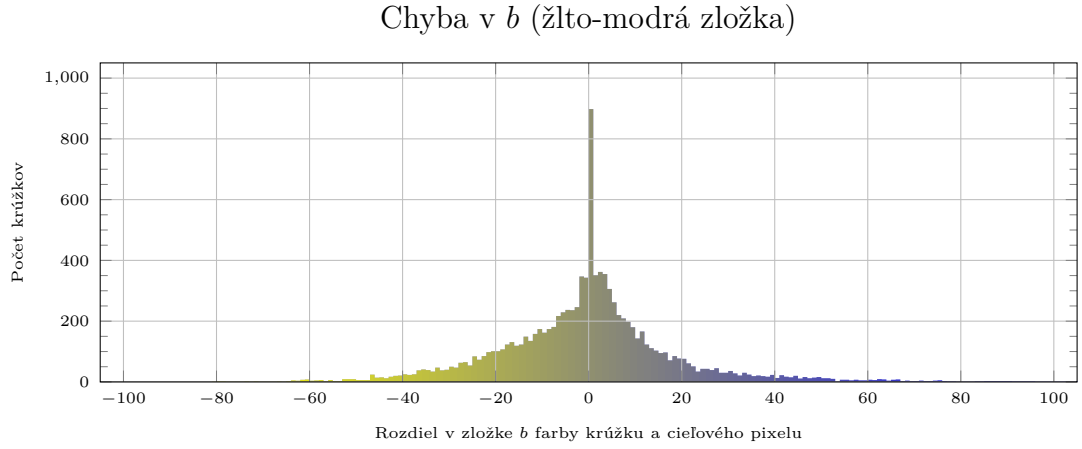
Na nasledujúcich histogramoch môžeme vidieť chybovosť užívateľov. Rozloženie je približne symetrické.



Histogram 3.10: Negatívne hodnoty znamenajú, že užívateľ zadal tmavšiu farbu ako má cieľový pixel. Pozitívne hodnoty znamenajú, že užívateľ zadal svetlejšiu farbu.

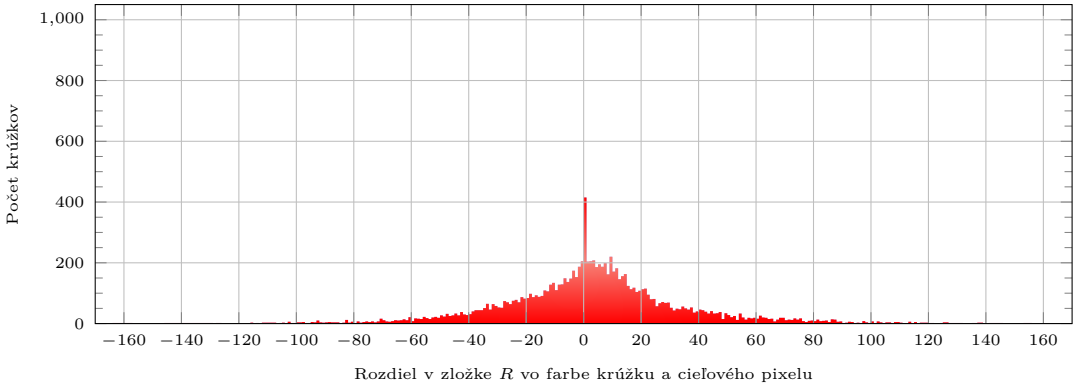


Histogram 3.11: Negatívne hodnoty znamenajú, že užívateľ zadal červenšiu farbu ako má cieľový pixel. Pozitívne hodnoty znamenajú, že užívateľ zadal zelenšiu farbu.



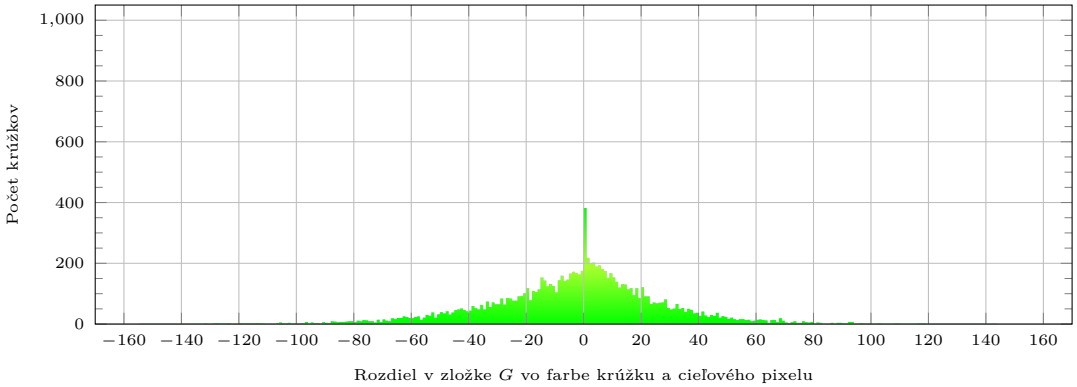
Histogram 3.12: Negatívne hodnoty znamenajú, že užívateľ zadal žltšiu farbu ako má cieľový pixel. Pozitívne hodnoty znamenajú, že užívateľ zadal modrejšiu farbu.

Chyba v červenej zložke



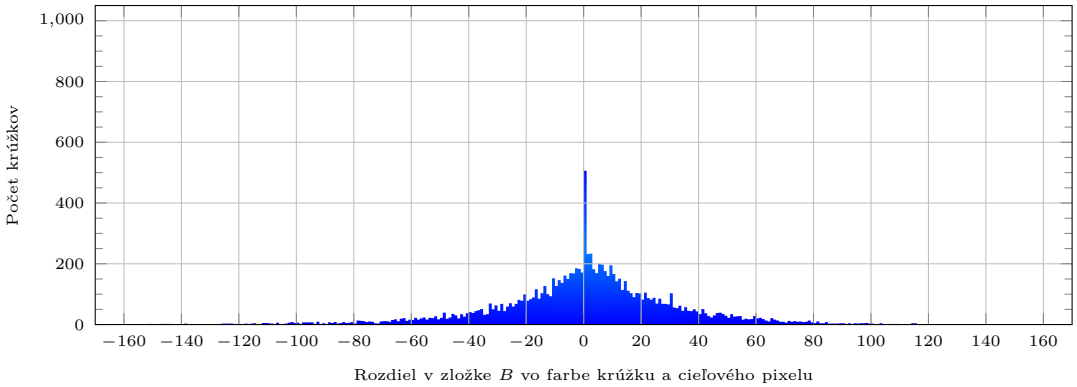
Histogram 3.13

Chyba v zelenej zložke



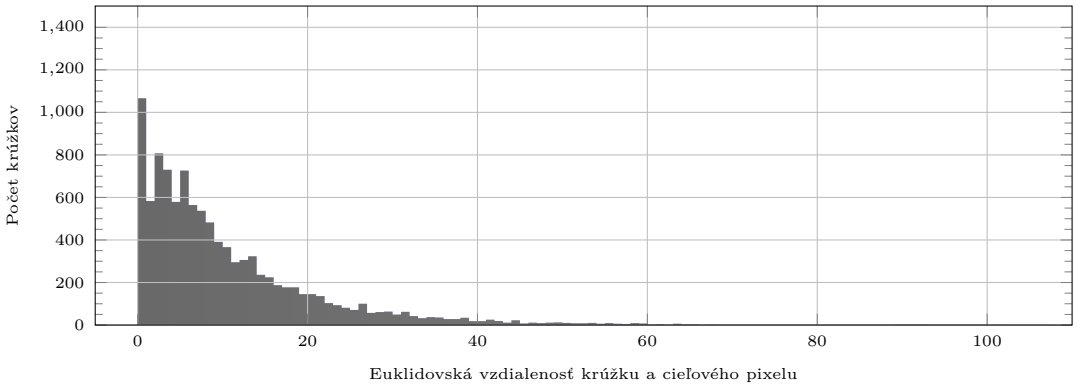
Histogram 3.14

Chyba v modrej zložke



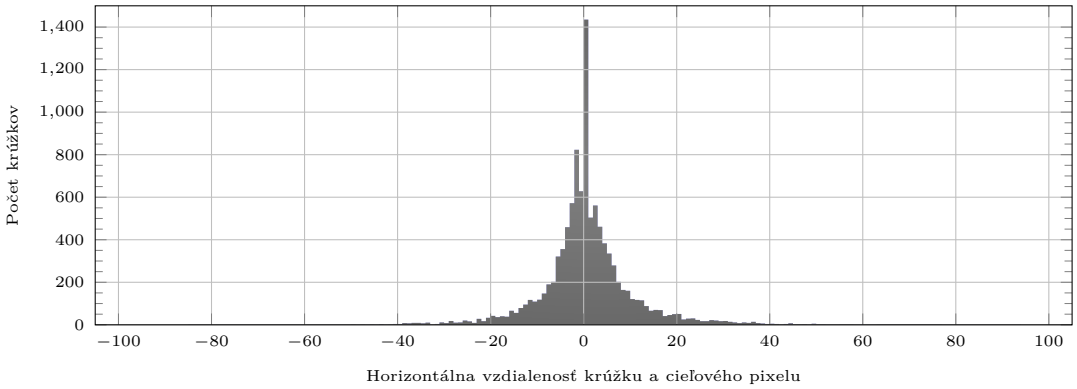
Histogram 3.15

Chyba v umiestnení krúžku (vzdialenosť)



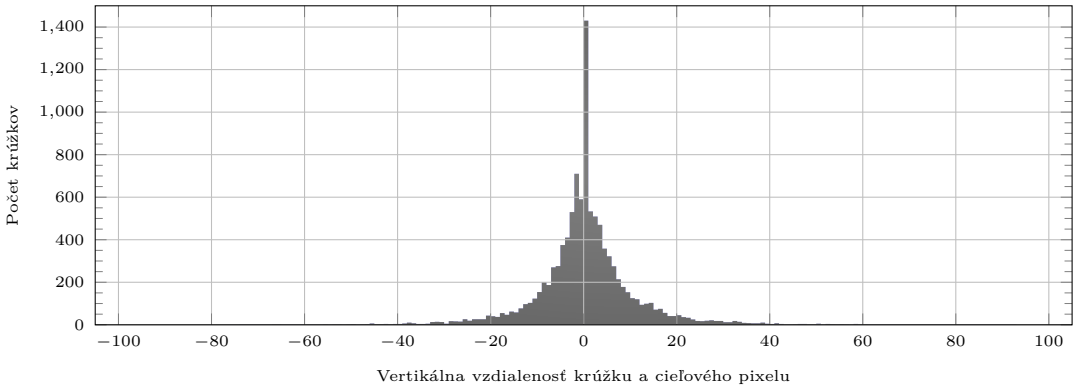
Histogram 3.16

Chyba v umiestnení krúžku v zložke X



Histogram 3.17

Chyba v umiestnení krúžku v zložke Y



Histogram 3.18

3.5 Použitie nákresov pre vyhľadávanie v obrázkoch

V tomto experimente sme získali nákresy, ktoré nám umožňujú vyhodnotiť efektivitu vyhľadávacieho systému nad databázou IACC.3 (3.1). Každý nákres nám poslúži ako dotaz pre vyhľadávanie. Vieme, ku ktorému obrázku nákres patrí, môžeme teda sledovať jeho pozíciu vo výsledkoch vyhľadávania. Priemerná pozícia vo výsledkoch vyhľadávania skrze všetky modely nám poskytuje vhodnú metriku pre zisťovanie efektivity vyhľadávacieho modelu.

3.5.1 Vyhľadávacie modely

Náš vyhľadávací systém sa skladá zo štyroch častí: predspracovanie obrázkov, výpočet podobnosti krúžka k obrázku, agregácia podobností krúžkov do podobnosti obrázka, zoraďovanie. Vyhodnocovať budeme vyhľadávanie vo farebných priestoroch RGB a Lab.

Predspracovanie obrázkov

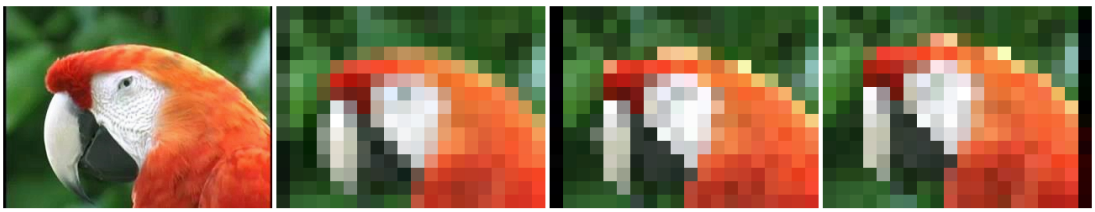
Hlavným cieľom pred-spracovania obrázkov je vyhľadávanie zrýchliť. Zrýchlenie docielime tým, že obrázky zmenšíme. Zmenšenie obrázkov je stratová operácia a preto môžeme očakávať, že bude mať negatívny dopad na presnosť vyhľadávania. Budeme porovnávať rôzne algoritmy na zmenšovanie obrázka aby sme zistili, ktorý algoritmus zhoršuje výsledky vyhľadávania najmenej:

- *Original* - Obrázok zostane bez zmeny.
- *Bicubic(šírka)* - Obrázok zmenšíme bežne známou bi-kubickou interpoláciou.
- *NearestNeighbor(šírka)* - Obrázok zmenšíme bežne známou interpoláciou nearest-neighbor.
- *GridDifferentNeighbor(šírka)* - Obrázok preložíme mriežkou a vyberieme najdominantnejšiu farbu v každej bunke. Ak niektorá susediaca bunka má už túto farbu vybranú, vyberie sa druhá najdominantnejšia farba, ktorá je od najdominantnejšej vzdialená aspoň 30.

Obrázky budú zmenšované vždy tak, aby bol zachovaný pôvodný pomer strán. Budeme vyhodnocovať efektivitu vyhľadávania pre šírky obrázka 1, 2, 4, 8, 16, 24, 32, 48, 64, 128, 256.

Ďalším krokom predspracovania bude vyhladenie obrázka. Predpokladáme, že užívateľ nevníma pixely samostatne, ale spolu s jeho okolím. Okrem nevyhladených obrázkov preto budeme vyhodnocovať aj vyhľadávanie, kde za farbu pixelu považujeme priemernú farbu jeho najbližšieho okolia - pôjde o štvorec veľkosti, ktorá závisí od šírky obrázka 3%x3%, 10%x10%, 20%x20%.

Celkovo tak bude vyhodnocovaných **136 konfigurácií preprocesorov**.



Obrázok 3.19: Ukážka zmenšených obrázkov podľa vyššie uvedených algoritmov na šírku obrázku 20px.

Vyhodnotenie vzdialenosti krúžkov k obrázku

V sekcii 3.4.1 sme použili metriku f_d, pomocou ktorej sme ku krúžku našli na obrázku pixel, ktorý má ku krúžku najbližšie čo do farby a vzdialenosti - tento, tzv. *cieľový*, pixel sme potom považovali za „správnu hodnotu krúžku“. Cieľový pixel určuje miesto a farbu, ktorú predpokladáme, že užívateľ mal na mysli. Podobne ako pri vyhodnocovaní vzdialenosti krúžku od obrázka budeme postupovať aj pri vyhľadávaní - prehľadáme okolie miesta, kam bol umiestnený krúžok a nájdeme najbližší pixel podľa jednej z týchto metrík:

1. *L2 3D RGB* - Euklidovská metrika vzdialenosti po projekcií do farebného priestoru RGB

2. *L2 3D Lab* - Euklidovská metrika vzdialenosti po projekcií do farebného priestoru Lab

3. *L2 5D RGB* - Euklidovská metrika v piatich dimenziách - 3 zložky určujú vzdialenosť vo farebnom priestore RGB a 2 zložky určujú vzdialenosť umiestnenia v X a v Y

4. *L2 5D Lab* - Euklidovská metrika v piatich dimenziách - 3 zložky určujú vzdialenosť vo farebnom priestore Lab a 2 zložky určujú vzdialenosť umiestnenia v X a v Y

Vzdialenosť najbližšieho pixelu v okolí *SearchRadius* od krúžku podľa uvedených metrík budeme považovať za vzdialenosť krúžku od obrázka. Pre obe horeuvedené metriky budeme vyhodnocovať SearchRadius 3%, 10% a 20% zo šírky obrázka. Celkovo tak budeme vyhodnocovať **12 spôsobov merania vzdialenosti**.

Agregácia krúžkov - Vyhodnotenie vzdialenosti obrázka a dotazu

Každý dotaz sa skladá z 3 až 5 farebných krúžkov. Úlohou tohto kroku je z troch až piatich hodnôt, ktoré reprezentujú vzdialenosť jednotlivých krúžkov od obrázka, vyhodnotiť celkovú vzdialenosť dotazu od obrázka. Túto agregáciu budeme vykonávať nasledovnými spôsobmi:

1. *Sum* - vzdialenosti krúžkov C sčítame

2. *SumRelativeDistance(K)* - transformované vzdialenosti krúžkov C' sčítame

Agregácia SumRelativeDistance počíta s transformovanou hodnotou krúžku $C' = max(0, \frac{C - C_{5min}}{|K - C_{5min}|})$, kde C je vzdialenosť krúžku od obrázka, C_{5min} je vzdialenosť piateho najbližšieho obrázka k danému krúžku a K je konštanta, pre ktorú sme zvolili hodnotu 50.

Celkovo tak budeme vyhodnocovať **2 typy agregácie**.

Zoradenie výsledkov

Keď máme vypočítanú vzdialenosť každého obrázka od dotazu, obrázky jednoducho zoradíme od najbližšieho k najvzdialenejšiemu a toto poradie budeme považovať za výsledok vyhľadávania.

3.5.2 Vyhodnotenie efektivity algoritmov

Efektivitu algoritmov vyhodnocujeme tak, že nákresy zozbierané nákresy použijeme ako vyhľadávacie dotazy. Priemernú pozíciu hľadaného obrázka vo výsledkoch, tzv. avgRank, považujeme za skóre algoritmu. Množstvo zozbieraných dotazov, ktorých je viac než 2500, zabezpečuje vysokú stabilitu výsledkov. Budeme vyhodnocovať efektivitu 136 spôsobov predspracovania obrázku, 12 spôsobov merania vzdialenosti krúžku od obrázka a 2 spôsoby agregácie, čo celkovo tvorí **3 264 konfigurácií**. Meranie sme vykonali programom, ktorého dokumentácia je v kapitole 4 tejto práce.

Najlepšie konfigurácie

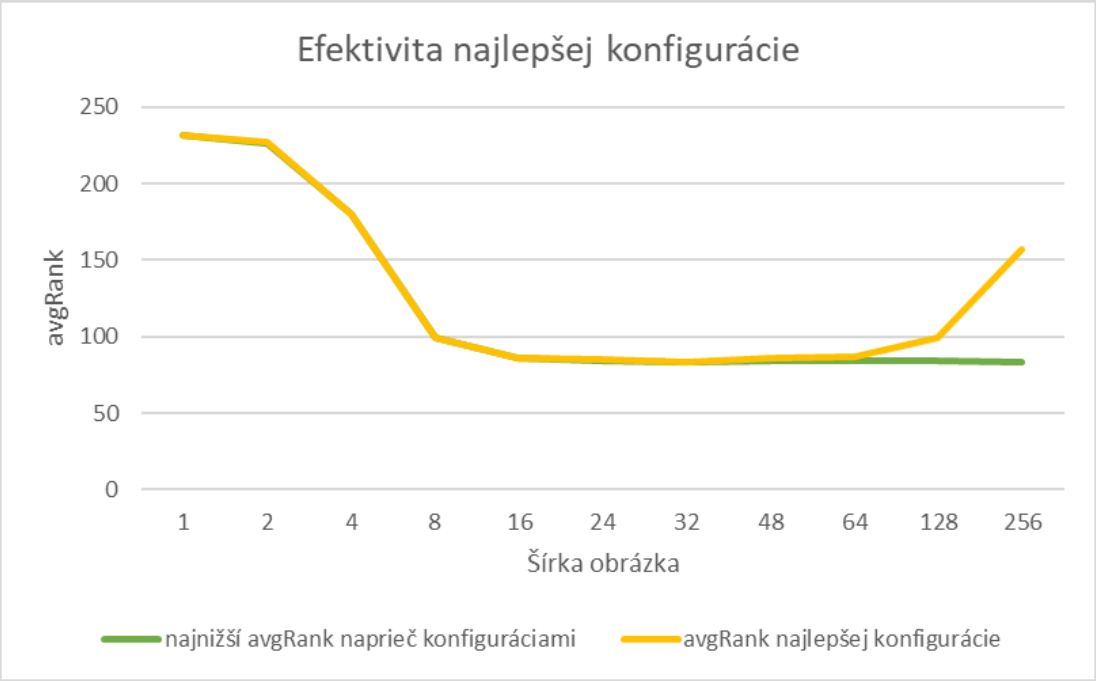
Najnižšiu priemernú pozíciu hľadaného obrázka dosiahlo nasledovných 8 konfigurácií. Všetky pracovali vo farebnom priestore Lab.

Skóre	Predspracovanie	Šírka obrázka (v px)	Search Radius (v %)	Metrika podobnosti	Agregácia	Vyhladenie (v %)
83,07	Bicubic	32	20	L2 5D	SumCir.	10
83,08	Bicubic	32	20	L2 5D	SumRel.	10
83,52	Bicubic	256	10	L2 3D	SumRel.	20
83,59	Near. N.	256	10	L2 3D	SumRel.	20
83,75	Bicubic	32	20	L2 5D	SumCir.	20
83,77	Bicubic	32	20	L2 5D	SumRel.	20
84,04	Bicubic	256	10	L2 3D	SumCir.	20

Minimálnu priemernú pozíciu vo výsledkoch nadobúda konfigurácia pri šírke obrázka 32px. Túto konfiguráciu, ktorá vyšla v rámci nášho merania ako najlepšia, budeme používať ako referenčnú. Proporčným zmenšením veľkosti obrázka zo šírky 320px na 32px sa priemerná pozícia zlepšila o 103,29. Domnievame sa, že k zlepšeniu došlo preto, že z obrázkov sa vytratili detaily, na ktoré sa užívatelia aj tak nesústredili.

Graf 3.20 nám ukazuje, že použitie najlepšej konfigurácie je vhodné najmä pre šírku obrázku 32px a nižšiu. Taktiež je možné zmenšiť obrázok až na šírku 16px, čo spôsobí úsporu pamäte až o 75%, ale spôsobí zhoršenie priemernej pozície výsledku len o 3%. Najlepšia konfigurácia však zmenšenie obrázka vyslovene

vyžaduje a pri veľkostiach obrázku 128px a vyšších dosahuje výrazne horšie výsledky ako iné konfigurácie.



Obrázok 3.20: Porovnanie avgRank konfigurácie, ktorá dosiahla celkovo najnižší avgRank (žltá čiara) a najnižšieho avgRank-u, ktorý bol dosiahnutý pri danej veľkosti (zelená čiara).

Vplyv farebného priestoru na priemernú pozíciu

Na grafe 3.21 vidíme, že konverzia obrázkov z farebného priestoru RGB do priestoru Lab výrazne zlepšuje výsledky. Pri šírke obrázka 16px až 256px sa priemerná pozícia vo výsledkoch zmenšilo o viac než 25%.

Vplyv algoritmu pre zmenšovanie obrázka na priemernú pozíciu

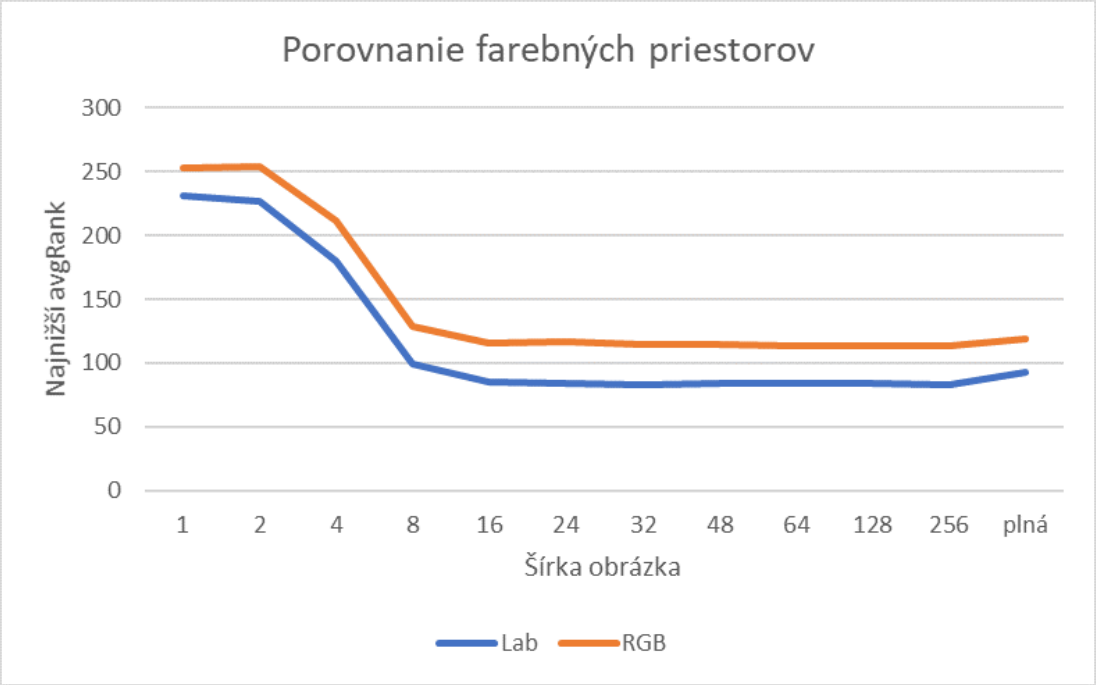
Na grafe 3.22 vidíme, že algoritmus, ktorým prebieha zmenšovanie má vplyv na výkon vyhľadávania hlavne pre obrázky do šírky 8px. Pri každej veľkosti je však vhodné použiť bikubickú interpoláciu.

Vplyv vyhladenia na priemernú pozíciu

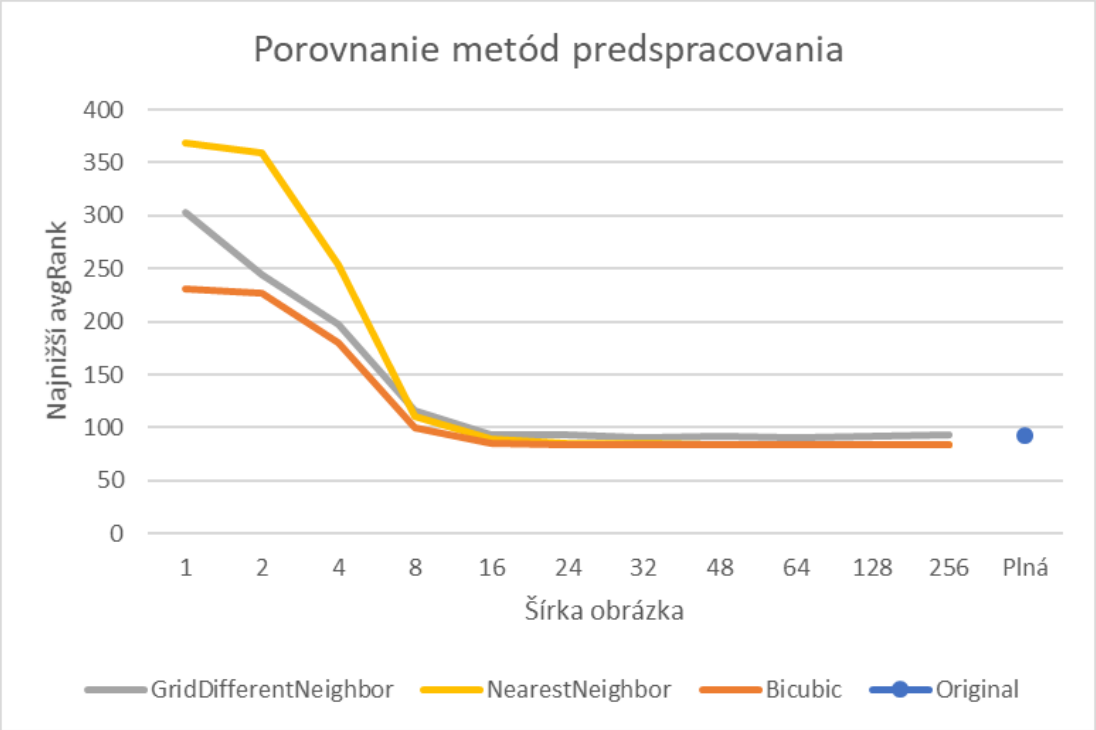
Na grafe 3.23 vidíme, že vyhladenie zlepšuje priemernú pozíciu obrázka pre väčšie veľkosti obrázkov.

Vplyv veľkosti prehľadávaného okolia na priemernú pozíciu

Na grafe 3.24 vidíme, že zväčšenie prehľadávaného okolia zlepšuje výsledky pre veľkosti obrázka do 128px.



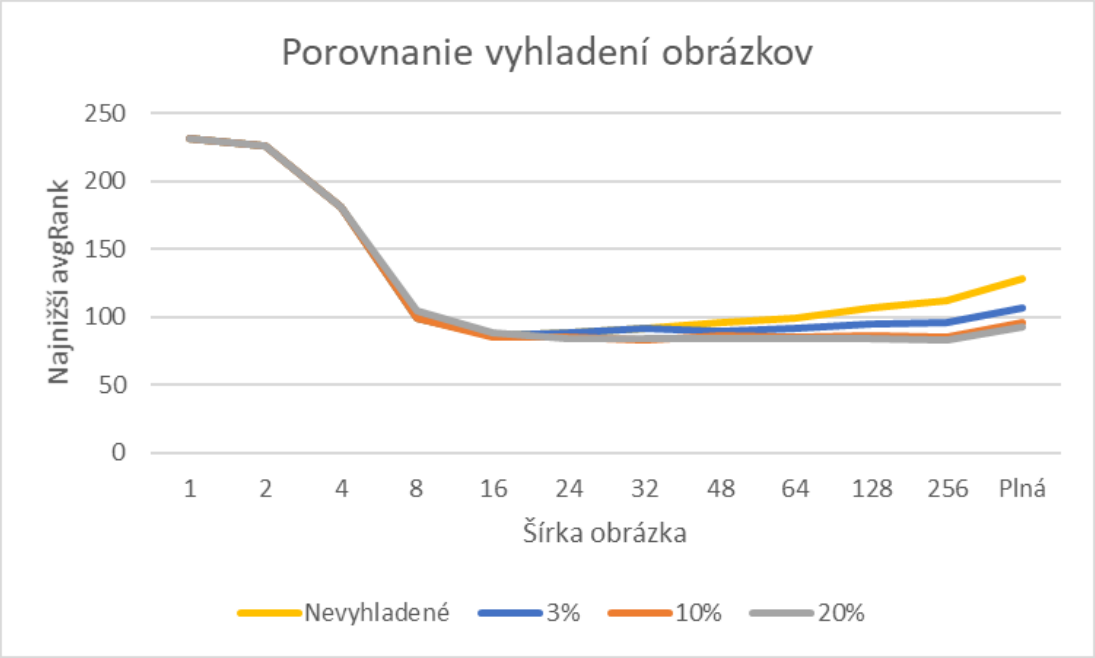
Obrázok 3.21: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedených farebných priestorov.



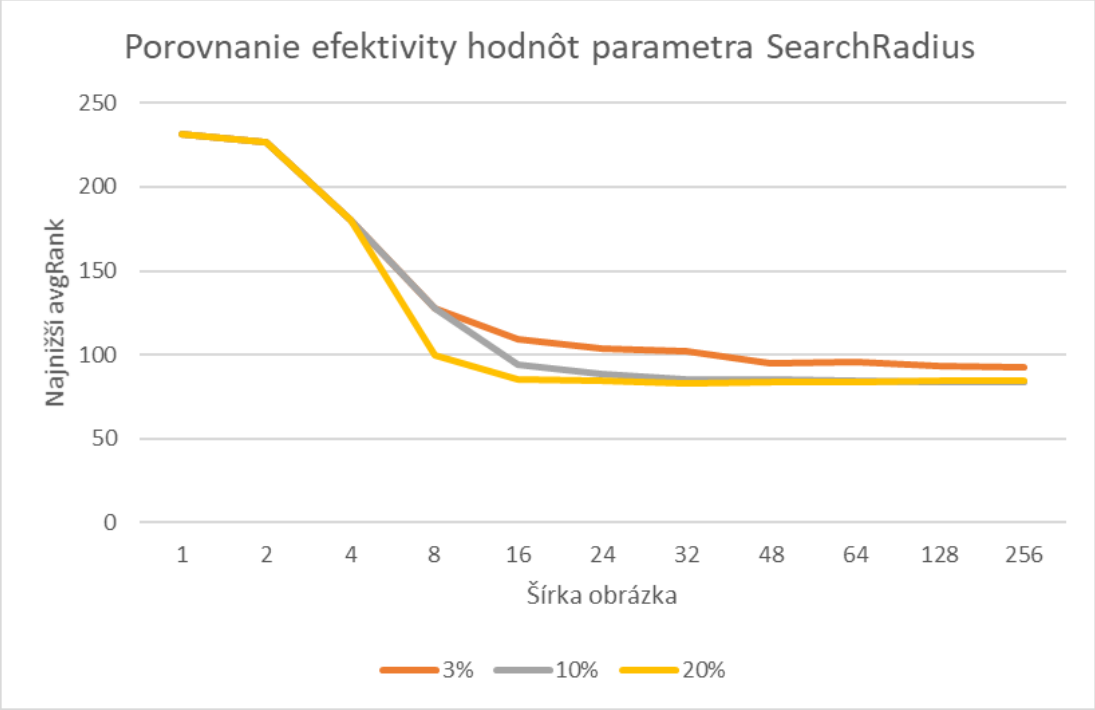
Obrázok 3.22: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedených spôsobov interpolácie.

Vplyv metriky vzdialenosti krúžku od obrázka na priemernú pozíciu

Na grafe 3.25 vidíme, že pre veľkosti do 64px je rozdiel medzi metrikou v 3D (3 zložky farby) a metrikou v 5D (2 zložky pozícia, 3 zložky farba) zanedbateľný.



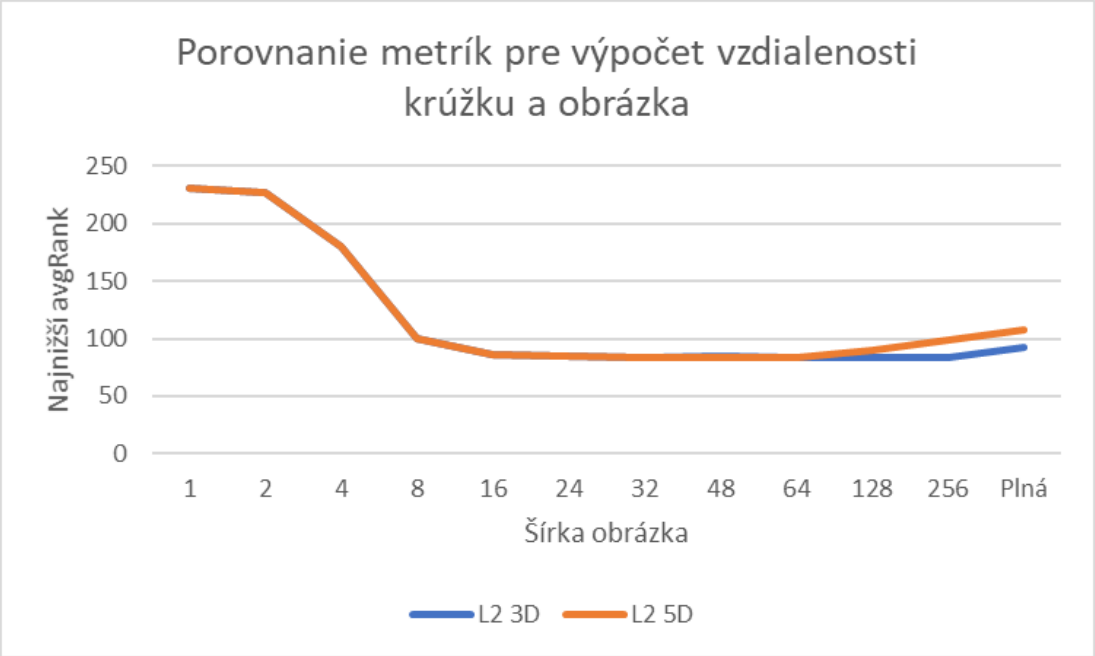
Obrázok 3.23: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedenej miery vyhladenia.



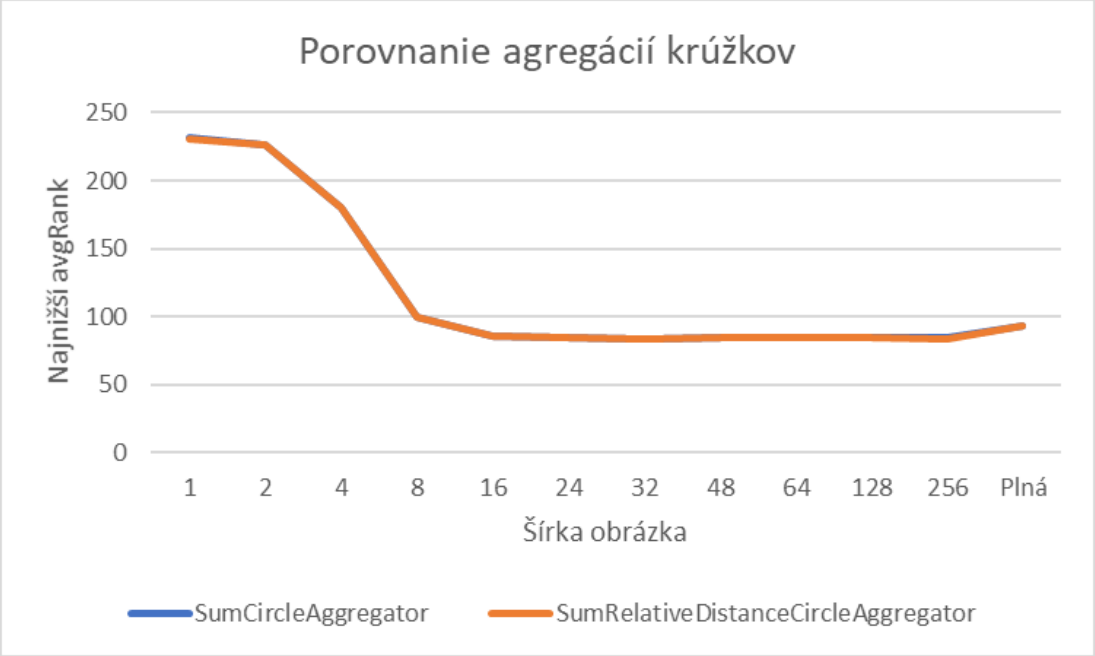
Obrázok 3.24: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedenej veľkosti prehľadávaného okolia.

Vplyv agregátorov na priemernú pozíciu

Na grafe 3.26 vidíme, že rozdiel medzi skúšanými agregátormi je zanedbateľný.



Obrázok 3.25: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedenej metriky pre výpočet vzdialenosti krúžku od obrázka.



Obrázok 3.26: Porovnanie najnižších avgRank-ov, aké boli dosiahnuté pri použití uvedeného agregátora.

3.6 Záver kapitoly

V tomto experimente sme získali viac než 2500 vyhľadávacích dotazov od užívateľov a využili sme ich pre nájdenie vhodného nastavenia parametrov vyhľadávacieho algoritmu. Rozšírili sme poznatky o schopnosti užívateľov zreprodukovať farby, našli vhodnú konfiguráciu parametrov vyhľadávacieho algoritmu a ukázali sme vplyv jednotlivých parametrov na celkovú efektivitu algoritmu. Veríme, že

naša analýza pomôže tvorcom nástrojov pre vyhľadávanie v obrázkoch a vo videu k dosahovaniu lepších výsledkov.

4. Dokumentácia k programu RankingModelEvaluator

4.1 Úvod

Program nazvaný RankingModelEvaluator, slúži na vyhodnotenie efektivity vyhľadávacích modelov a ich parametrov, ktoré boli predstavené v kapitole 3.5.1. Pripomeňme, že efektivitu algoritmu vyhodnocujeme tak, že nákresy získané od užívateľov počas experimentu (viď. kapitola 3.2) použijeme ako dotazy pre vyhľadávanie a vo výsledkoch sledujeme pozíciu obrázka, ku ktorému nákres patrí. Priemerná pozícia hľadaného obrázka naprieč všetkými vyhľadávaniami určuje efektivitu algoritmu.

4.2 Analýza

Je potrebné, aby program spĺňal nasledujúce požiadavky:

1. **Program musí byť schopný vyhodnotiť efektivitu vyhľadávacích modelov predstavených v kapitole 3.5.1**. Je teda nutné, aby mal vyhľadávací model nasledovné parametre:
   (a) **Preprocessor** - typ predspracovania obrázku
      i. **ImageWidth** - cieľová veľkosť obrázka - potrebný len ak preprocesor zmenšuje obrázok
      ii. **PixelRegionWidthPercentage** - šírka štvorca, z ktorého bude vypočítaný jeden pixel - potrebný len v prípade, že preprocesor vyhladzuje obrázok
   (b) **SimilarityMeasurer** - metrika, ktorá určí vzdialenosť krúžku od obrázka
      i. **ColorSpace** - farebný priestor, v ktorom bude rozdiel vo farbách počítaný
      ii. **SearchRadiusWidthPercentage** - maximálna vzdialenosť cieľového pixelu od stredu krúžku
   (c) **Aggregator** - typ agregátora vzdialeností krúžkov do vzdialenosti dotazu od obrázka
      i. **Constant** - konštanta pre RelativeDistanceAggregator
2. **Program musí byť ľahko rozšíriteľný.** Program RankingModelEvaluator existuje iba a jedine za účelom vyhodnotenia experimentu vykonaného v kapitole č. 3. Uvedená kapitola však je živé dielo až do momentu dokončenia celej práce, pričom sa pri jej tvorbe charakter skúmaných parametrov mení. Vývojár tak musí byť schopný program rozšíriť o vyhodnocovanie iných vyhľadávacích algoritmov len s minimálnou úpravou existujúceho kódu.

3. **Program musí výsledky zapísať do súboru vo formáte CSV.** Tabuľka bude obsahovať priemernú pozíciu hľadaného obrázka vo výsledkoch pre každú vyhodnocovanú kombináciu parametrov vyhľadávacieho modelu. Formát CSV je požadovaným výstupom programu preto, že uľahčuje následné vyhodnocovanie výsledkov pomocou programov na prácu s tabuľkami.

4. **Program musí v prípade pádu v priebehu vyhodnocovania algoritmu zapísať stack trace výnimky,** aby uľahčil nápravu chyby.

5. **Program musí zobrazovať percentá dokončenia**, aby v prípade dlhého behu umožnil užívateľovi vytvoriť odhad o dokončení.

6. **Program musí byť spustiteľný na operačnom systéme Windows 10.**

4.3 Návrh

Nad samotným vyhľadávaním uvažujeme počas celej práce v nasledujúcich fázach:

1. Predspracovanie obrázkov

2. Výpočet podobnosti krúžkov z dotazu k obrázkom

3. Výpočet podobnosti obrázka k dotazu agregáciou podobností jednotlivých krúžkov

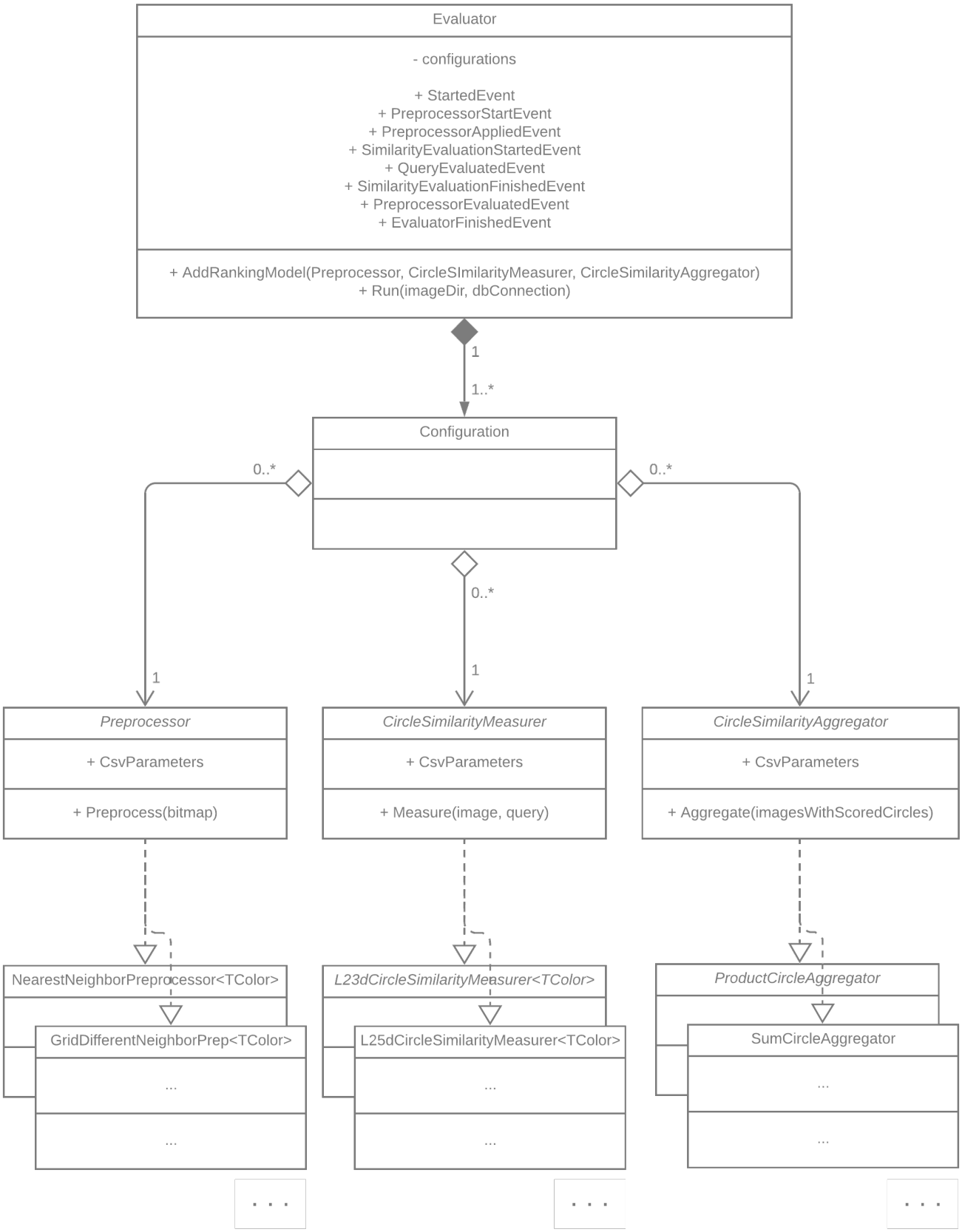
4. Zoradenie obrázkov od najpodobnejšieho k najmenej podobnému.

Podľa týchto fáz rozdelíme aj algoritmus pre vyhľadávanie, aby sme mali možnosť fázy jednotlivé časti vyhľadávacieho algoritmu vymeniť za iné. Z požiadavky č. 1 vidíme, že parametre algoritmu závisia od konkrétnej voľby častí algoritmu, pričom musíme mať na pamäti aj požiadavku č. 2 podľa ktorej je potrebné umožniť, aby parametre mohli pribúdať bez úpravy existujúceho kódu.

Požadovanú flexibilitu vyhľadávacieho algoritmu docielime pomocou návrhového vzoru *Strategy*. Pre každú fázu vyhľadávacieho algoritmu teda bude existovať abstraktná trieda alebo interface, ktorý bude akceptovaný vyhľadávacím algoritmom. Trieda Evaluator, v ktorej algoritmus beží, tak nemusí mať vedomosť o všetkých jeho parametroch. Požiadavku č. 5 o indikovaní priebehu výpočtu vyhodnocovania zabezpečíme pomocou návrhového vzoru *Observer* a konkrétne jeho implementácie v .NET Core frameworku vo forme *Events* - umožníme tak komponente, ktorá bude zabezpečovať zobrazenie indikátora priebehu aby informácie o priebehu mala k dispozícií.

Aby sme podľa požiadavky č. 3 zabezpečili výstup v tabuľkovom formáte, budeme od každej triedy, ktorá implementuje časť vyhľadávacieho algoritmu požadovať, aby definovala vlastnosť `CsvParameters`, ktorá vo forme slovníka vráti všetky svoje parametre.

Program budeme implementovať v jazyku C# na platforme .NET Core, aby bol spustiteľný na viacerých platformách, vrátane operačného systému Microsoft Windows, podľa požiadavky č. 6.



Obrázok 4.1: Diagram ukazuje vzťah triedy Evaluator a tried, ktoré tvoria vyhľadávací algoritmus.

4.4 Vývojárska dokumentácia

4.4.1 Prehľad tried

Hlavné triedy

- `Program` - inicializuje Evaluator, ProgressBarInterface a CsvOutput
- `Evaluator` - zabezpečí vyhodnotenie vyhľadávacích modelov
- `ProggressBarInterface` - zabezpečuje vykreslenie indikátorov priebehu
- `CsvOutput` - po skončení vyhodnocovania zapíše výsledok do CSV súboru

Abstraktné triedy, ktoré tvoria vyhľadávací algoritmus

Konkrétna konfigurácia vyhľadávacieho algoritmu je jednoznačne určená kombináciou troch objektov, ktoré dedia od nasledovných abstraktných typov:

- `Preprocessor` - má verejnú metódu Preprocess(Bitmap), ktorá zabezpečí predspracovanie obrázku
- `CircleSimilarityMeasurer` - má verejnú metódu Measure(image, query), ktorá príjme predspracovaný obrázok a sadu krúžkov a vráti vzdialenosť každého krúžku od obrázka
  - `TargetFinder` - má verejnú metódu FindTarget(query, circle, image), ktorá vráti súradnice pixelu, ktorý má najbližšie ku krúžku, ktorý zadal užívateľ - viď. kapitola 3.4.1.
- `CircleSmiliarityAggregator` - má verejnú metódu Aggregate(), ktorá príjme zoznam obrázkov s ohodnotenými krúžkami a vráti zoznam ohodnotených obrázkov

Práca s farbami

Podľa požiadavky č. 1(b)i potrebujeme, aby sme vedeli vyhľadávanie spustiť v rôznych farebných priestoroch. Aby sme sa vyhli písaniu samostatného kódu pre každý farebný priestor, budeme k farbám pristupovať cez abstraktnú triedu:

- `Color3d` - abstraktná trieda, ktorá zabezpečuje načítanie a konverziu farieb z trojdimenzionálnych priestoroch
- `ColorLab` - implementácia triedy Color3d pre farebný priestor Lab
- `ColorRgb` - implementácia triedy Color3d pre farebný priestor Rgb

Správa obrázkov

Nasledujúce triedy obsahujú informáce o obrázkoch:

- `ImageWarehouse` - obsahuje všetky obrázky z datasetu v ktorom vyhľadávame - instancie triedy Image

- `Image` - obsahuje konkrétny obrázok z datasetu vo forme dvojrozmerného poľa Color3d[,] a jeho metadáta
- `ExperimentPicture` - trieda, ktorá obsahuje všetky údaje o obrázku zozbierané počas experimentu, ktorý je popísaný v kapitole č. 3

Pomocné triedy

Nasledujúce triedy slúžia pre uskladnenie medzivýsledkov:

- `ImageWithRankedCircles` - obsahuje Image a zoznam ohodnotení krúžkov
- `RankedImage` - obsahuje Image a skóre obrázku
- `EvaluationResults` - obsahuje pozície hľadaných obrázkov, ktoré dosiahla konkrétna konfigurácia

4.4.2 Inicializácia programu

Trieda *Program* vytvára Evaluator a zabezpečuje správne spracovanie výnimiek. Všetky jej metódy sú súkromné:

- `Main()` - spustí Evaluator, ProgressBarInterface, CsvOutput a postará sa o zachyhtenie výnimiek
- `AddRankingModels<TColor>(evaluator)` - určí, ktoré konfigurácie budú vyhodnotené, prijíma generický parameter, ktorý určuje v ktorom farebnom priestore bude prebiehať meranie rozdielu vo farbách
- `AnnounceException(exception)` - vypíše výnimku na konzolu a uloží ju do súboru
- `GetDatasetDir()` - vráti cestu k priečinku s obrázkami, v ktorých bude prebiehať vyhľadávanie

Nastavenie vyhodnocovaných konfigurácií

Nastavenie konfigurácií, ktoré sa budú vyhodnocovať prebieha v metóde AddRankingModels<TColor>(). Metóda AddRankingModels je volaná raz pre každý farebný priestor z funkcie Main().

Pridať alebo odobrať Preprocessor, CircleSimilarityMeasurer, TargetFinder alebo CircleSmiliarityAggregator je možné jednoduchou úpravou metódy AddRankingModels<TColor>().

4.4.3 Vyhodnocovanie konfigúrácií

Trieda *Evaluator* zabezpečuje vyhodnocovanie všetkých konfigurácií. Má dve verejné metódy:

- `AddRankingModel()` - Funkcia prijíma tri parametre: Preprocessor, CircleSimilarityMeasurer, CircleSimilarityAggregator. Túto trojicu zaradí do zoznamu konfigurácií, ktoré budú vyhodnotené.

- `Run(imageDir, dbConnection)` - Funkcia v ktorej prebehne samotné vyhodnocovanie.

Postup vyhodnocovania konfigurácií

Funkcia `Run()` pracuje tak, že aplikuje Preprocessor na všetky obrázky a následne postupuje po jednotlivých spôsoboch meranie vzdialenosti spúšťa vyhľadávania pre každý dotaz, ktorý je k dispozícií. V každom momente vyhodnocujeme len jeden Preprocessor a jeden CircleSimilarityMeasurer. Samotné vyhľadávania bežia paralelne. Keď získame ohodnotenia krúžkov pre každý obrázok, aplikujeme na ne každý spôsob agregácie, ktorý má byť pre daný Preprocessor a CircleSimilarityMeasurer vyhodnotený.

Udalosti

Metóda `Run()` počas svojho behu vyvoláva udalosti, ktoré signalizujú priebeh vyhodnocovania. Pomocou udalostí sú v programe implementované indikátory priebehu a uloženie výsledkov spracovávania do súboru. Všetky udalosti sú spracovávané synchrónne a preto by mali vykonávať len krátko trvajúce operácie. Dostupné sú nasledovné udalosti:

- `StartedEvent` - po načítaní dotazov z databázy a pred aplikovaním prvého preprocesora
- `PreprocessorStartEvent` - pred aplikovaním každého preprocesora
- `PreprocessorAppliedEvent` - po aplikovaní preprocesora na všetky obrázky
- `SimilarityEvaluationStartedEvent` - pred tým ako začneme vyhodnocovať CircleSimilarityMeasurer
- `QueryEvaluatedEvent` - po vyhodnotení všetkých agregátorov pre jeden dotaz
- `SimilarityEvaluationFinishedEvent` - po vyhodnotení všetkých vyhľadávaní pre konkrétny CircleSimilarityMeasurer
- `PreprocessorEvaluatedEvent` - po vyhodnotení všetkých konfigurácií s daným preprocesorom
- `EvaluatorFinishedEvent` - po vyhodnotení všetkých konfigurácií

4.4.4 Indikátory priebehu

Trieda zabezpečuje *ProgressBarInterface* zobrazovanie indikátorov priebehu. Jej jedinou verejnou metódou je konštruktor, ktorý príjma Evaluator, na ktorom si zaregistruje handlery pre udalosti, podľa ktorých vykresľuje indikátory priebehu. Na vykresľovanie indikátorov používa knižnicu *ShellProgressBar*.

4.4.5 Výstup programu

Trieda *CsvOutput* sa stará o vytvorenie CSV súboru, ktorý obsahuje výsledky hodnotenia konfigurácií vyhľadávacích algoritmov. Jej jedinou verejnou metódou je konštruktor, ktorý príjma Evaluator, na ktorom si zaregistruje handler pre udalosť EvaluatorFinished - tzn. pre dokončenie vyhodnocovania.

Keď je vyhodnocovanie všetkých konfigurácií dokončené, CsvOutput vytvorí CSV súbor. Súboru obsahuje stĺpec AvgRankZeroBased a stĺpec pre každý parameter, ktorý vracia niektorý vyhodnocovaný Preprocessor, CircleSimilarityMeasurer alebo CircleSimilarityAggregator vo vlastnosti CsvParameters. Súbor je uložený s názvom `results-[ticks].csv`, kde `[ticks]` označuje počet 100-nanosekundových intervalov, ktoré uplynuli od 1. januára, 0001.

4.4.6 Ďalší rozvoj programu

Pre rozšírenie programu o nový farebný priestor, v ktorom prebieha meranie rozdielu vo farbách je potrebné vytvoriť triedu ColorNew, ktorá dedí od triedy Color3d. Následne je možné upraviť metódu Main(), aby zavolala AddRankingModels<ColorNew>(). Každá kombinácia parametrov potom bude vyhodnotenená aj pri meraní rozdielu vo farbách v novom farebnom priestore.

Pridať nový spôsob pred spracovania obrázku, merania vzdialenosti krúžku od obrázka alebo nový spôsob agregácie je možné jednoduchým vytvorením triedy, ktorá dedí od príslušného abstraktného predka - v prípade predspracovania je to Preprocessor, v prípade merania vzdialenosti krúžku od obrázka je to CircleSimilarityMeasurer a v prípade agregácie krúžkov to je CircleSimilarity Aggregator. Pre vyhodnotenie efektivity tejto triedy, je potrebné vytvoriť jej inštanciu v metóde Program.AddRankingModels<TColor>().

Do ďalšej verzie programu odporúčame implementovať mechanizmus, ktorý by dokázal identifikovať, ktoré konfigurácie už boli na danom datasete vyhodnotené a tak by namiesto opätovného vyhodnocovania len do výstupu zahrnul predchádzajúce výsledky. Výstup programu by tak kompletné výsledky bez potreby ručného spájania CSV súborov.

4.5 Užívateľská dokumentácia

Program je možné spustiť na operačnom systéme Microsoft Windows 10 64-bit dvojklikom na súbor *RankingModelEvaluator.exe*. Program pre svoju prácu potrebuje databázu nákresov, ktorá sa nachádza v súbore *colors.db* a je distribuovaná spolu s programom. Okrem databázy dotazov program ešte potrebuje prístup k samotným obrázkom vo formáte jpg, v ktorých má vyhľadávanie prebiehať - tie musia byť prítomné v priečinku s názvom *dataset*. Priečinok *dataset* sa môže nachádzať v rovnakom priečinku ako program, alebo o 1 až 6 levelov vyššie. Obrázky, ktoré boli použité pri zbere dát musia byť pomenované vo forme *VideoID_FrameId.jpg*. Zoznam obrázkov použitých pri zbere dát je možné nájsť v prílohe A.1.

Po spustení programu sa zobrazí okno s konzolovou aplikáciou, na ktorú sú vykresľované indikátory priebehu. Na prvých dvoch riadkoch sa zobrazuje indikátor priebehu celého procesu. Vyhodnocovanie konfigurácií prebieha sekvenčne

po preprocesoroch - konfigurácie, ktoré obsahujú druhý preprocesor sa začínajú vyhodnocovať až keď konfigurácie sú všetky konfigurácie pre prvý preprocesor vyhodnotené. Miera dokončenia celého procesu je meraná v počte preprocesorov, ktoré už boli kompletne vyhodnotené.

V podstrome hlavného indikátora priebehu sú indikátory priebehu vyhodnocovania konfigurácií, ktoré obsahujú konkrétny preprocesor alebo konkrétnu dvojicu preprocesora a metriky pre určenie vzdialenosti krúžku od obrázka.

5. Webová aplikácia pre vyhľadávanie v obrázkoch

Pre demonštráciu funkčnosti vyhľadávania v obrázkoch sme vytvorili webovú aplikáciu. Aplikácia umožňuje nakresliť farebné krúžky na plátno rovnakým spôsobom ako to bolo vyžadované od užívateľov, od ktorých sme zbierali dáta počas experimentu v kapitole 3.2.

5.1 Vyhľadávací model

Webová aplikácia vyhľadáva v obrázkoch pomocou modelu, ktorý bol v kapitole 3.5.2 vyhodnotený ako najlepší.

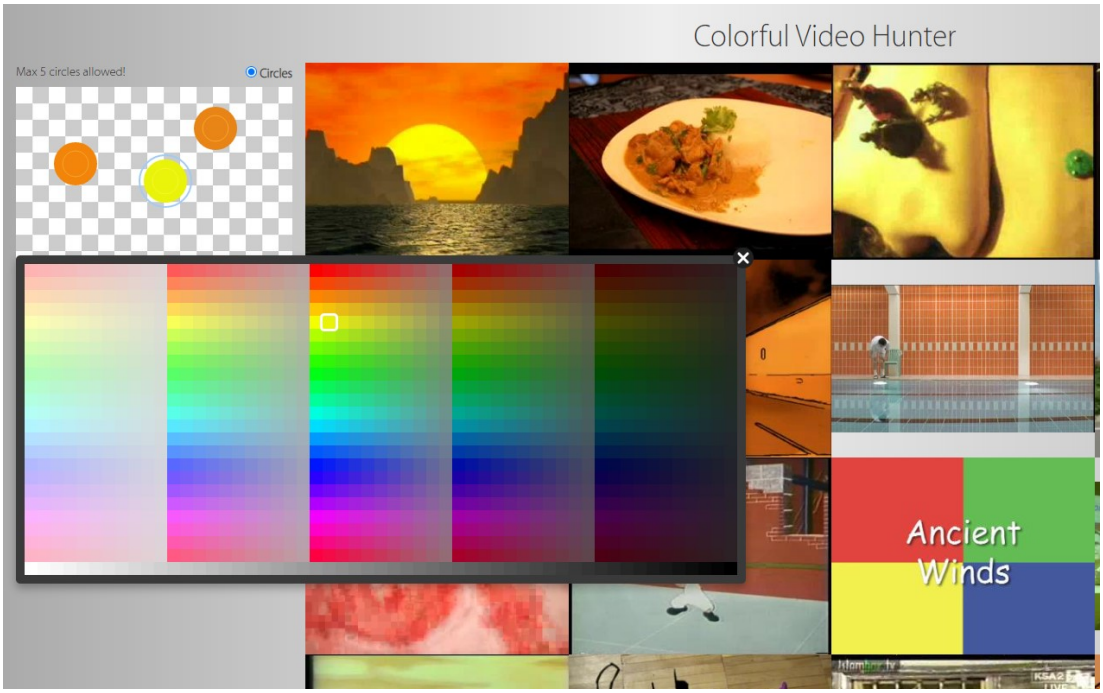
- **Predspracovanie:** bikubická interpolácia
  - **Šírka obrázka:** 32
  - **Vyhladenie:** 10%
- **Metrika podobnosti**: L2 5D
  - **Farebný priestor:** Lab
  - **Vyhľadávací polomer**: 20%
- **Agregácia:** SumCircleAggregator (jednoduché sčítanie krúžkov)

5.2 Použitie webovej aplikácie

Webová aplikácia sa spúšťa dvojklikom na súbor ImageSearchWebApplication.exe. Podporovaný je operačný systém Microsoft Windows 10 64-bit. Po spustení je aplikácia dostupná na adrese `https://localhost:5001`.

Na ľavej strane obrazovky je možné umiestniť na plátno až 5 krúžkov. Ľavý klik na plátno pridáva krúžok, ľavý klik na krúžok umožňuje zmeniť jeho farbu, pravý klik na krúžok umožňuje krúžok odobrať.

Aplikácia vyhľadáva v obrázkoch v priečinku `wwwroot/imgs/dataset`. Podporovaný je formát `jpg`.



Obrázok 5.1: Ukážka webovej aplikácie - výsledky vyhľadávania v obrázkoch a úprava nákresu - dotazu pre vyhľadávanie v obrázkoch.

5.3 Vývojárska dokumentácia

5.3.1 Prehliadačová časť

Základom prehliadačovej časti aplikácie je weová hra, ktorá bola použitá pri zbere dát od užívateľov, ktorej autorom je Artur Finger. Bližšie je popísaná v kapitole 3.2. V aplikácií boli vykonané len minimálne zmeny - premiestnenie elementov a skrytie nepotrebných elementov, odoslanie vyhľadávacieho dotazu v potrebnej forme a zobrazenie výsledku.

5.3.2 Serverová časť

Serverová časť webovej aplikácie je postavený na platforme ASP.NET Core 3.1. Aplikácia využíva kód programu RankingModelEvaluator, ktorý je popísaný v kapitole 4 - ide o ďalsí projekt v tom istom Solution. Serverová časť obsahuje len približne 100 riadkov ručne písaného kódu.

Aplikácia prijíma POST dotazy na adrese https://localhost:5001/Api/Search s telom dotazu vo fromáte JSON:

```
{
  "Width": 320,
  "Height": 240,
  "Circles": [{
    "X": 10,
    "Y": 10,
    "Color": {"R": 10, "G": 10, "B": 10}
  },
  {
    "X": 90,
    "Y": 90,
    "Color": {"R": 90, "G": 50, "B": 10}
  }]
}
```

Položky „Width“ a „Height“ označujú šírku a výšku plátna na ktorom sú krúžky umiestnené. Položka „Circles“ obsahuje pole, kde každý objekt reprezentuje jeden krúžok.

Server vracia výsledky vyhľadávanie v nasledovnej podobe:

```
{
    "results": [
        "38439_90.jpg",
        "39678_12079.jpg",
        "35759_3041.jpg",
        "38039_6084.jpg",
        "... viac vysledkov ...",
    ]
}
```

Druhý typ dotazu, ktorý servrová aplikácia prijíma je GET request na adresu https://localhost:5001/Picture/Get/*obrazok.jpg*, kde *obrazok.jpg* je názov ktoréhokoľvek obrázka, ktorý sa nachádza v priečinku wwwroot/imgs/dataset. Tento request vráti daný obrázok.

Serverová časť je, samozrejme, schopná aj vybavovať požiadavky na statické súbory. V prípade GET dotazu na https://localhost:5001 vracia obsah súboru wwwroot/index.html.

Záver práce

V práci sme uviedli do problematiky vyhľadávania multimediálneho obsahu a pripomenuli základné princípy vnímania a zobrazovania farieb, ich reprezentáciu v počítači. Potom sme sa zamerali na problematiku hľadania známych obrázkov pomocou jednoduchého farebného nákresu. Pre tieto účely sme zozbierali množstvo dát od užívateľov, ktoré nám umožnili vyhodnotiť presnosť rôznych vyhľadávacích algoritmov.

Hlavným prínosom práce je, že vyhľadávacie modely sme porovnali na veľkom množstve dát od reálnych užívateľov. V kombinácií s ďalšími vyhľadávacími modelmi môže byť vyhľadávanie pomocou jednoduchých farebných skíc veľká pomoc, keďže dokáže odfiltrovať podstatnú časť databázy.

Seznam použité literatury

Trecvid data availability. URL https://trecvid.nist.gov/trecvid.data.html#tv17.

(1948). Minutes of the thirty-first meeting of the board of directors of the optical society of america, incorporated. *J. Opt. Soc. Am.*, **38**(7), 651–651. doi: 10.1364/JOSA.38.000651. URL http://www.osapublishing.org/abstract.cfm?URI=josa-38-7-651.

(1974). Information retrieval on-line. f. w. lancaster and e. g. fayen. los angeles: Wiley-becker & hayes. 417 pp. (1974). *Journal of the American Society for Information Science*, **25**(5), 336–337. doi: 10.1002/asi.4630250510. URL https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630250510.

(1999). Multimedia systems and equipment - colour measurement and management - part 2-1: Colour management - default rgb colour space - srgb. URL https://webstore.iec.ch/publication/6169.

(2005). URL https://www.adobe.com/digitalimag/pdfs/AdobeRGB1998.pdf.

Baeza-Yates, R. A. a Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., USA. ISBN 020139829X.

BenRG (2014). Comparison of some rgb and cmyk colour gamut on a cie 1931 xy chromaticity diagram. https://commons.wikimedia.org/wiki/File:CIE1931xy_gamut_comparison.svg.

Bowmaker, J. K. a Dartnall, H. J. (1980). Visual pigments of rods and cones in a human retina. *The Journal of Physiology*, **298**(1), 501–511. doi: 10.1113/jphysiol.1980.sp013097. URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1980.sp013097.

Deglr6328 (2006). Spectrum of an old style (halophosphate phosphor) fluorescent light. https://en.wikipedia.org/wiki/File:Spectrum_of_halophosphate_type_fluorescent_bulb_(f30t12_ww_rs).png.

der Wissenschaften in Wien., K. A. (1872). *Sitzungsberichte der Kaiserlichen Akademie der Wissenschaften. Mathematisch-Naturwissenschaftliche Classe: Physiologie, Anatomie und theoretische Medicin*. Number v. 66. K.-K. Hof- und Staatsdruckerei in Commission bei F. Tempsky. URL https://books.google.cz/books?id=u5MCAAAAYAAJ.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. a Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Goodfellow, I., Bengio, Y. a Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Googolplexbyte (2013). Diagram of the opponent process. https://en.wikipedia.org/wiki/File:Diagram_of_the_opponent_process.png.

Horvath, M. (2006). This image demonstrates the principle of additive color mixing. https://commons.wikimedia.org/wiki/File:AdditiveColor.svg.

J.K., B. a H.J.A., D. (1980). Visual pigments of rods and cones in a human retina. pages 501–511. Vectorized version of the GFDL image from 2007.

Kraaij, W. (2005). Variations on language modeling for information retrieval. *SIGIR Forum*, **39**(1), 61. ISSN 0163-5840. doi: 10.1145/1067268.1067291. URL https://doi.org/10.1145/1067268.1067291.

Krizhevsky, A., Sutskever, I. a Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Leibetseder, A., Kletz, S. a Schoeffmann, K. (2018). Sketch-based similarity search for collaborative feature maps. In Schoeffmann, K., Chalidabhongse, T. H., Ngo, C. W., Aramvith, S., O'Connor, N. E., Ho, Y.-S., Gabbouj, M. a Elgammal, A., editors, *MultiMedia Modeling*, pages 425–430, Cham, 2018. Springer International Publishing. ISBN 978-3-319-73600-6.

Lewis, L. (2019). This is what happens in an internet minute. https://www.allaccess.com/merge/archive/29580/2019-this-is-what-happens-in-an-internet-minute.

Lokoč, J., Phuong, A. N., Vomlelová, M. a Ngo, C.-W. (2017). Color-sketch simulator: A guide for color-based visual known-item search. In Cong, G., Peng, W.-C., Zhang, W. E., Li, C. a Sun, A., editors, *Advanced Data Mining and Applications*, pages 754–763, Cham, 2017. Springer International Publishing. ISBN 978-3-319-69179-4.

Lokoč, J., Kovalčík, G. a Souček, T. (2018). Revisiting siret video retrieval tool. In Schoeffmann, K., Chalidabhongse, T. H., Ngo, C. W., Aramvith, S., O'Connor, N. E., Ho, Y.-S., Gabbouj, M. a Elgammal, A., editors, *MultiMedia Modeling*, pages 419–424, Cham, 2018. Springer International Publishing. ISBN 978-3-319-73600-6.

Lokoč, J., Kovalčík, G., Münzer, B., Schöffmann, K., Bailer, W., Gasser, R., Vrochidis, S., Nguyen, P. A., Rujikietgumjorn, S. a Barthel, K. U. (2019). Interactive search or sequential browsing? a detailed analysis of the video browser showdown 2018. *ACM Trans. Multimedia Comput. Commun. Appl.*, **15**(1). ISSN 1551-6857. doi: 10.1145/3295663. URL https://doi.org/10.1145/3295663.

Long, B. a Chang, Y. (2014). Chapter 4 - visual search ranking. In Long, B. a Chang, Y., editors, *Relevance Ranking for Vertical Search Engines*, pages 59 – 80. Morgan Kaufmann, Boston. ISBN 978-0-12-407171-1. doi: https://doi.org/10.1016/B978-0-12-407171-1.00004-6. URL http://www.sciencedirect.com/science/article/pii/B9780124071711000046.

Lv, Y. a Zhai, C. (2009). Positional language models for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 299–306, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584836. doi: 10.1145/1571941.1571994. URL https://doi.org/10.1145/1571941.1571994.

Manak (2016). Kozolupy u křižovatky silnic. https://commons.wikimedia.org/wiki/File:123_Za_Kozolupami_u_k%C5%99i%C5%BEovatky_SSV.jpg.

Nassau, K. (1998). In Nassau, K., editor, *Color for Science, Art and Technology*, volume 1 of *AZimuth*. North-Holland. doi: https://doi.org/10.1016/S1387-6783(98)80004-4. URL http://www.sciencedirect.com/science/article/pii/S1387678398800044.

Roelleke, T. a Wang, J. (2007). Probabilistic logical modelling of the binary independence retrieval model. In *Proceedings of the 1st International Conference on Theory of Information Retrieval (ICTIR 07)-Studies in Theory of Information Retrieval*.

Roettgers, J. (2015). Special report: How we really use our camera phones. https://gigaom.com/2015/01/23/personal-photos-videos-user-generated-content-statistics/.

Salton, G., Wong, A. a Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, **18**(11), 613–620. ISSN 0001-0782. doi: 10.1145/361219.361220. URL https://doi.org/10.1145/361219.361220.

Salton, G., Fox, E. A. a Wu, H. (1983). Extended boolean information retrieval. *Commun. ACM*, **26**(11), 1022–1036. ISSN 0001-0782. doi: 10.1145/182.358466. URL https://doi.org/10.1145/182.358466.

Schoeffmann, K. (2019). Video browser showdown 2012-2019: A review. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–4. doi: 10.1109/CBMI.2019.8877397.

Sharma, M. a Patel, R. (2013). A survey on information retrieval models, techniques and applications.

Sun, X., Wang, C., Sud, A., Xu, C. a Zhang, L. (2013). Magicbrush: Image search by color sketch. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 475–476, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2404-5. doi: 10.1145/2502081.2502276. URL http://doi.acm.org/10.1145/2502081.2502276.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. a Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*. URL http://arxiv.org/abs/1409.4842.

Wagner, J., Baur, T., Zhang, Y., Valstar, M. F., Schuller, B. a André, E. (2018). Applying cooperative machine learning to speed up the annotation of social signals in large multi-modal corpora.

Zoznam obrázkov

A. Prílohy

A.1 Zoznam obrázkov použitých v experimente s krúžkami

Experiment s krúžkami, popísaný v kapitole 3, využíva nasledujúcich 100 obrázkov. Zoznam obrázkov má formát *Video ID-Frame ID*. Zdroj dát je popísaný v sekcii 3.1.

35350-89	35979-89	36584-3046	37835-75	38876-5076
35368-6091	36009-6084	36679-89	37940-6084	38888-10077
35378-3041	36042-12079	36692-1625	38020-89	38962-37
35402-6091	36090-6084	36729-89	38039-6084	38971-44
35405-6084	36122-6084	36847-3041	38045-89	39036-10077
35493-6091	36124-6091	36893-6084	38201-12079	39099-5076
35546-3039	36126-24157	36898-6084	38212-5076	39221-10077
35559-89	36132-6084	37006-44	38256-12079	39226-3041
35577-90	36141-6047	37068-72	38260-6084	39255-9674
35601-5076	36235-6084	37211-6091	38342-6084	39342-10077
35753-6091	36267-89	37214-10077	38439-90	39376-75
35759-3041	36334-5076	37301-12079	38517-6047	39382-5076
35797-6084	36358-20131	37370-9665	38542-6074	39419-6084
35808-6084	36375-89	37411-30	38554-10077	39476-10077
35815-89	36395-5076	37420-12079	38576-3041	39506-12079
35883-5076	36437-12124	37461-12079	38653-12079	39644-12079
35915-5076	36451-6084	37496-12079	38677-9119	39650-90
35917-6084	36471-90	37739-45	38701-6084	39678-12079
35932-6115	36510-3046	37763-75	38803-6084	39724-12079
35959-89	36549-3039	37829-6091	38871-87	39926-89