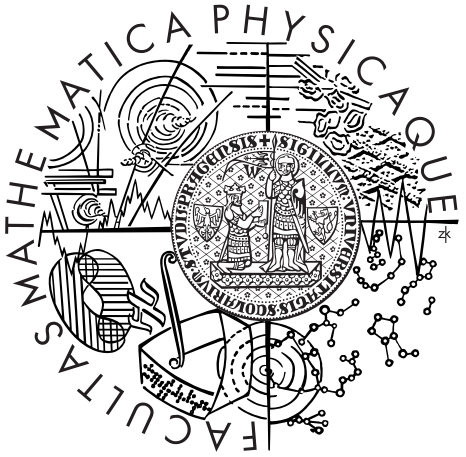
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Lukáš Navrátil

Podobnostní vyhledávání v kolekci obrázků

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Tomáš Bartoš

Studijní program: Informatika

Studijní obor: Programování

Praha 2012

Děkuji vedoucímu své práce RNDr. Tomáši Bartošovi za jeho připomínky a nápady při vedení této práce. Rovněž děkuji všem uživatelům, kteří se zapojili do uživatelského hodnocení podobnosti obrázků.

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 v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 24. května 2012 Lukáš Navrátil

Název práce: Podobnostní vyhledávání v kolekci obrázků

Autor: Lukáš Navrátil

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Tomáš Bartoš, Katedra softwarového inženýrství

Abstrakt: Detekce význačných bodů obrázku a jejich popis pomocí deskriptorů je často používaná technika v některých odvětvích počítačového vidění. Práce si klade za cíl prozkoumat a ověřit možnosti použití této techniky pro podobnostní vyhledávání v kolekci obrázků. Pro tyto potřeby bude pomocí vytvořené webové aplikace nasbíráno od uživatelů dostatečné množství hodnocení podobnosti, která budou poté porovnána s výsledky podobností spočítaných pomocí implementace algoritmu SURF, jednoho z algoritmů určeného pro detekci a popis význačných bodů. Bude diskutován vliv metrik a parametrů ovlivňujících výpočet výsledné podobnosti mezi obrázky a bude učiněna snaha o nalezení takových nastavení, při jejichž použití se výsledky budou co nejvíc blížit uživatelskému vnímání podobnosti.

Klíčová slova: SURF, podobnost obrázků, význačné body, deskriptory, podobnostní vyhledávání

Title: Similarity search in image collections

Author: Lukáš Navrátil

Department: Department of Software Engineering

Supervisor: RNDr. Tomáš Bartoš, Department of Software Engineering

Abstract: Detection of keypoints from image and their characterization by using descriptors is common technique in some branches of computer vision. The goal of this thesis is to explore and confirm usability of this technique for similarity retrieval in image collections. For this purpose it will be created a web application used for collecting ratings of similarity from users which will be subsequently compared with results computed by the implementation of SURF algorithm, one of algorithms used for detection and description of image keypoints. It will also be discussed the impact of metrics and parameters influencing results of computation of similarity between images and it will be made an effort to find settings for which computed results will be closest to user's similarity perception.

Keywords: SURF, image similarity, keypoints, descriptors, similarity search

Obsah

Úvod

Získávání obrázků podle obsahu (CBIR z anglického pojmu Content–based image retrieval) je technika vyhledávání obrázků založená na analýze samotného obsahu obrázků. Obsahem obrázku mohou být např. objekty vyskytující se na obrázku, jejich barva nebo textura. Naopak zde nehrají výraznou roli metadata přidružená k obrázku (která jsou většinou vytvořena lidmi), jako jsou např. klíčová slova nebo popis.

Jedním z odvětví CBIR je vyhledávání podobných obrázků v libovolné kolekci obrázků. Toto vyhledávání má několik aplikací např. query–by–example vyhledávání na webu (vyhledávání kde předlohou k hledání je již nějaký existující objekt – v tomto případě obrázek), segmentace v kolekcích obrázků a další.

Lidské oko je schopno rozpoznat téměř ihned zda jsou si obrázky podobné či nikoliv, ale pro počítač je to složitý problém. Pro tento účel je používáno několik různých metod, jejich vývoj ale stále probíhá a je zde prostor pro hledání nových i zlepšování existujících technik.

V počítačovém vidění se již několik let používají algoritmy SIFT a SURF, které úspěšně fungují v některých odvětvích jako jsou detekce pohybu, vyhledávání objektů na scéně, spojování obrázků a další. Efektivní a funkční použití této metody pro vyhledávání podobnosti v kolekci obrázků ale podle našich informací zatím nebylo nalezeno a ověřeno. Je dokonce i možné, že tyto metody nejsou pro získávání podobnosti vhodné a rozumně použitelné.

Tato bakalářská práce si klade za cíl zjistit použitelnost těchto algoritmů pro podobnostní vyhledávání v kolekcích obrázků a zkusit naleznout optimální metriky a parametry, pro které se budou výsledky co nejvíce blížit uživatelskému vnímání podobnosti. K tomuto účelu bylo prvním úkolem práce naprogramovat webovou aplikaci, jejímž prostřednictvím bude možné nasbírat dostatečné množství uživatelských hodnocení vůči kterým budou poté výsledky algoritmů porovnávány.

Průběh realizace této práce lze rozdělit do těchto kroků:

- Vytvoření aplikace, kde uživatelé hodnotí podobnost administrátorem vložených obrázků
- Nasbírání dostatečného množství uživatelských hodnocení
- Implementace algoritmů pro výpočet podobnosti obrázků
- Výběr optimálních metrik, nastavení parametrů algoritmu a jejich testování
- Analýza výsledků hodnocení uživatelů a algoritmu

V tomto pořadí je psán i text této práce, na začátku proto lze naleznout analýzu a informace k implementaci aplikace pro uživatelské hodnocení, dále následuje popis algoritmu pro hodnocení a práce končí vyhodnocením výsledků.

1. Analýza programu

Prvním úkolem práce bylo vytvořit webovou aplikaci, která by umožňovala vytvořit databázi obrázků. Administrátorovi by poskytovala pohodlné rozhraní pro její správu — přidávání, editaci a mazání obrázků, jejich přiřazování do skupin, aktivaci a deaktivaci — společně s dalšími nástroji pro celkovou správu aplikace jako je např. správa uživatelů a monitorování jejich činnosti.

Uživatelům má aplikace umožňovat hodnotit podobnost obrázků a to v těchto třech režimech:

- dvojice obrázků – podobnost ano/ne
- dvojice obrázků – podobnost na stupnici 0-100%
- N obrázků – k danému obrázku seřadit zbývajících N-1 obrázků od nejpodobnejšího k nejméně podobnému

Uživatelská hodnocení jsou v aplikaci ukládána tak, aby bylo možné porovnat nasbírané hodnoty uživatelského vnímání podobnosti s výsledky, které jsou spočítány počítačem pomocí vhodného algoritmu.

Další částí k analýze a naprogramování pak byla implementace vhodného algoritmu pro vyhodnocování podobnosti mezi obrázky.

V této kapitole bude uvedeno pouze stručné shrnutí klíčových částí a přehled architektury, detailní popisy jednotlivých případů užití a specifikace funkčnosti jednotlivých částí jsou popsány v dalších dokumentech analýzy na přiloženém disku.

1.1 Základní části architektury

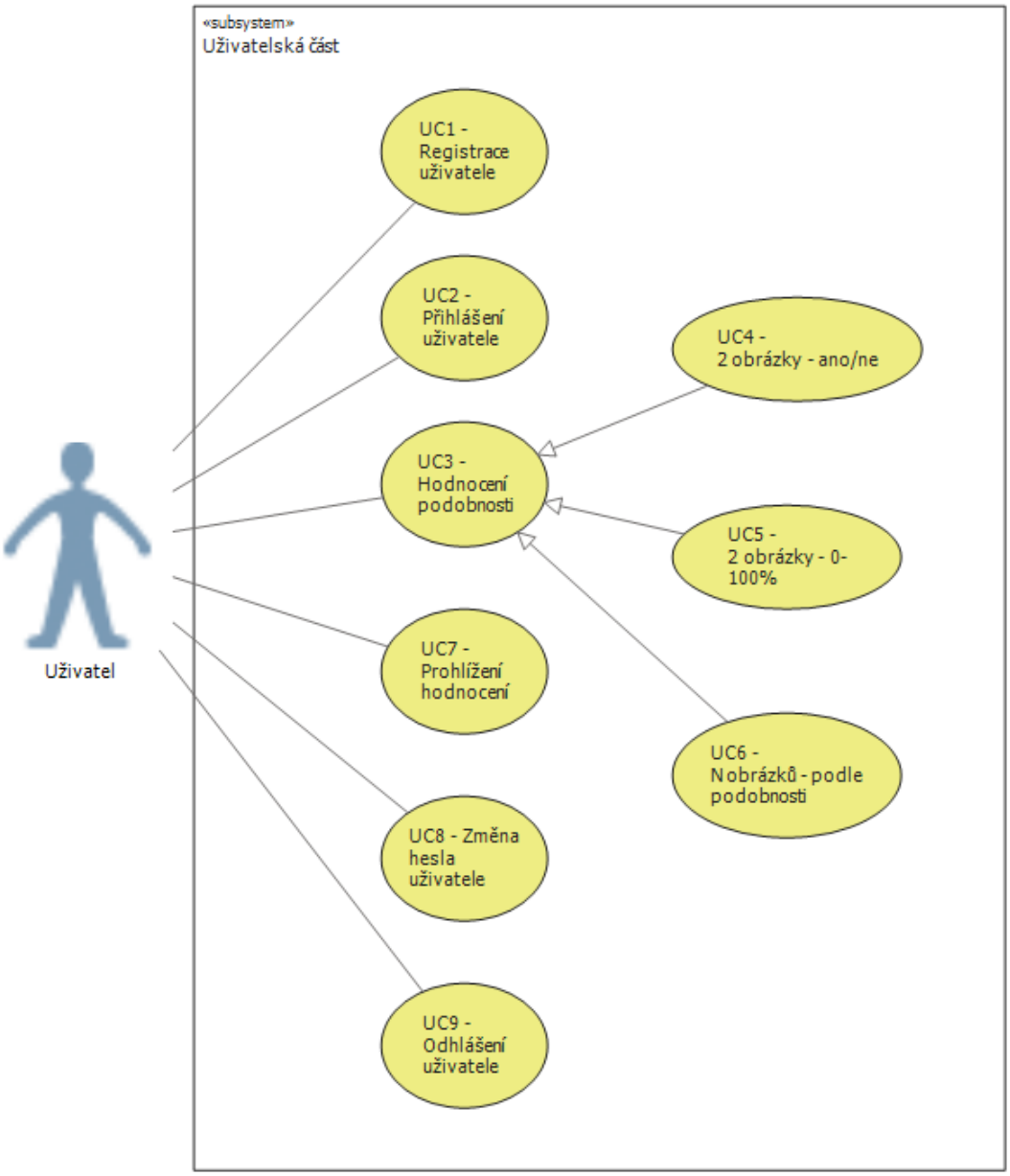
Celé programové řešení jsem se rozhodl rozdělit na 2 separátní části:

- SimBIm (zkratka pro Similarity Between Images) – webová aplikace, kde uživatelé hodnotí podobnost a administrátoři spravují data a výsledky
- SimCom (zkratka pro Similarity Computer) – desktopová aplikace pro výpočet podobností mezi obrázky

1.1.1 SimBIm

SimBIm lze rozdělit na 2 hlavní části — na část uživatelskou a administrátorskou.

V uživatelské sekci se může zaregistrovat libovolný uživatel, který pak hodnotí podobnost obrázků a prohlíží svoje hodnocení. Případy užití jsou zobrazeny na obrázku 1.1

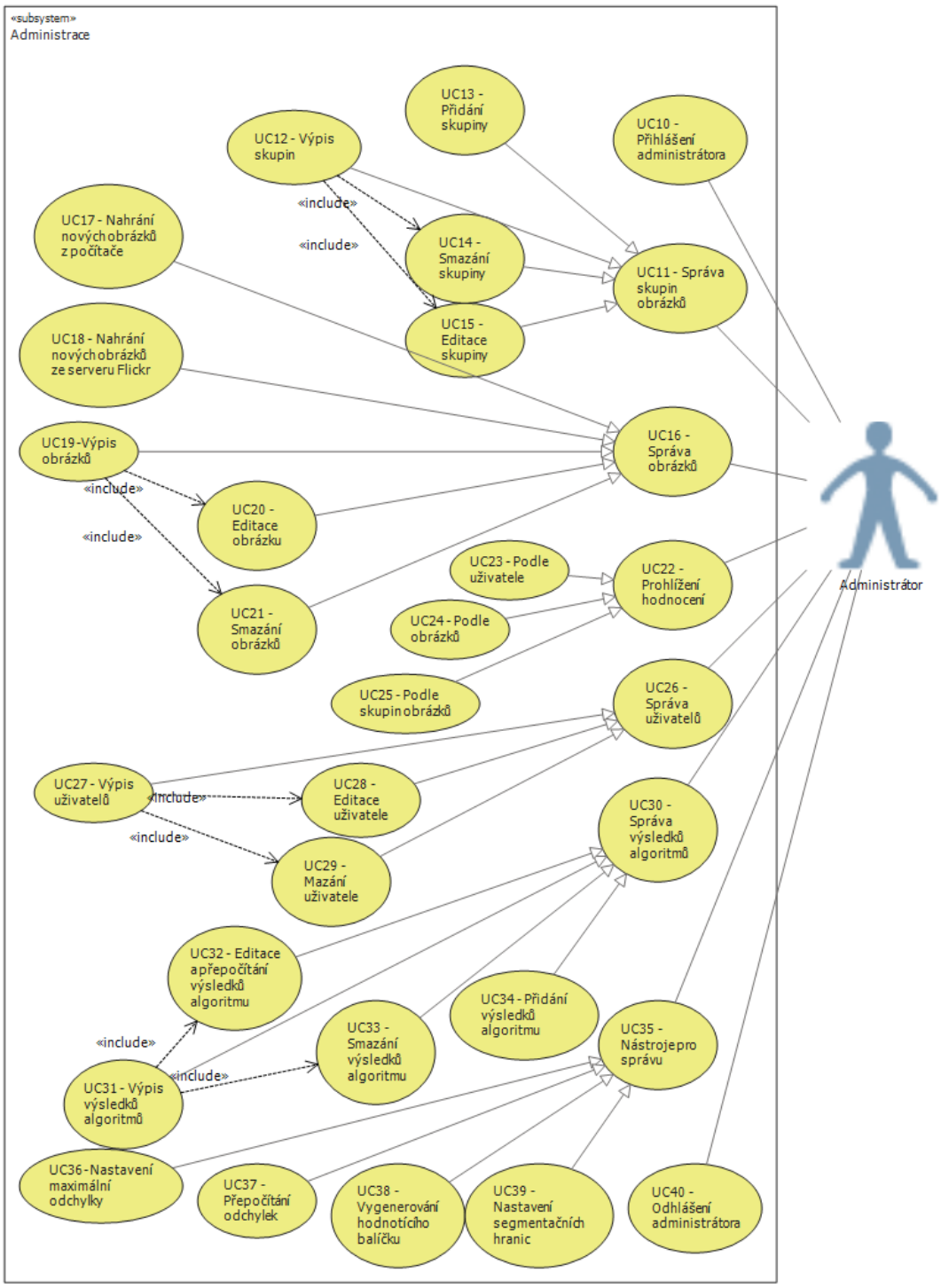


Obrázek 1.1: Diagram případů užití v uživatelské části aplikace SimBIm

Pomocí administrátorské sekce je pak prováděna kompletní správa aplikace, je možné zde provádět např. následující úkony:

- Správa obrázků k hodnocení – nahrávání obrázků, vytváření a přiřazování do skupin
- Správa uživatelů včetně prohlížení jejich hodnocení
- Nahrávání vypočtených hodnocení a prohlížení jejich výsledků

Diagram případů užití je zobrazen na obrázku 1.2, jejich kompletní popis je pak uveden v přiložené analýze.



Obrázek 1.2: Diagram případů užití v administrátorské části aplikace SimBIm

1.1.2 SimCom

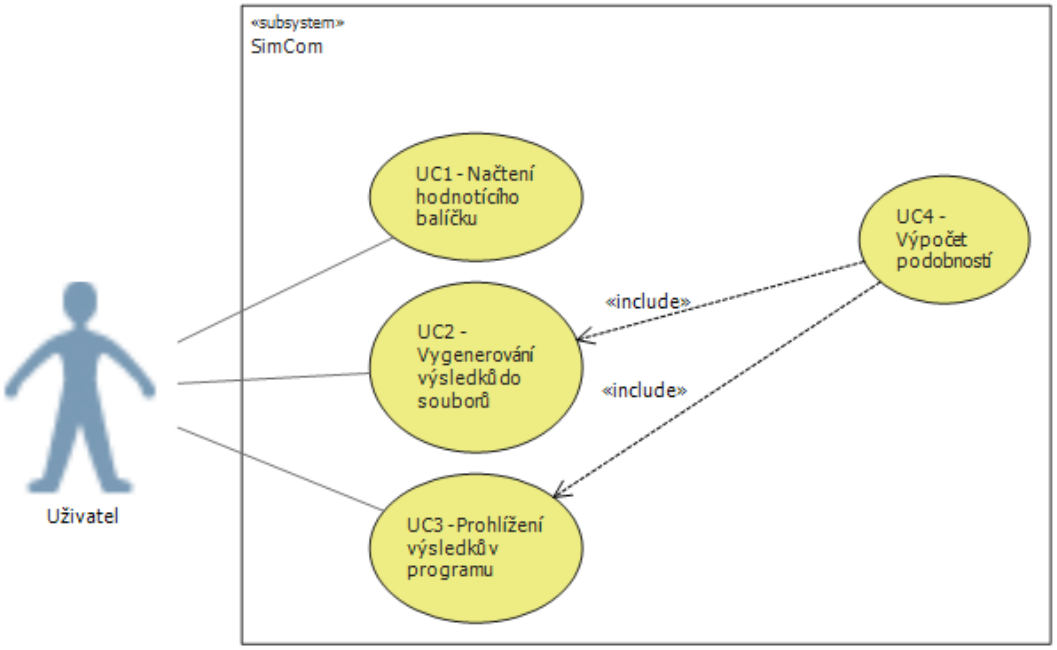
Není vždy vhodné spouštět dlouhotrvající výpočty podobností na vzdáleném webovém serveru, rozhodnul jsem se proto vytvořit separátní desktopovou aplikaci, která je určená na výpočet podobností mezi obrázky.

Toto rozdělení má výhodu především v oddělení výpočtů podobností od zbytku hodnotícího serveru. Výpočetní program může mít např. u sebe na počítači

několik různých uživatelů, kteří mohou vyvíjet hodnotící podobnostní metriky a poté mohou jednotlivé výsledky odeslat na server bez nutnosti do něj zasahovat (stačí jim jen účet mající oprávnění nahrát vypočtená data na server). Tyto výsledky jsou jim pak porovnány s výsledky uživatelů a oni tak dostávají zpětnou vazbu, jak je jejich metrika kvalitní. Tím se tak ze serveru stává zároveň platforma pro benchmarking různých podobnostních metrik.

Další výhodou je, že není nutné do webové aplikace nahrávat všechny použité grafické knihovny, které jsou jednak často veliké a zároveň na některých webserverech nemusí fungovat, jelikož často využívají v pozadí knihovny napsané v C++. Další výhodou pak může být i možnost běhu části výpočtu na grafické kartě, pokud to daná karta podporuje, která je většinou mnohem výkonnější na desktopovém počítači než na serveru.

SimCom má 3 hlavní úkoly – načtení hodnotícího balíčku, výpočet podobností a vygenerování vypočtených hodnocení. Tyto úkoly jsou zobrazeny v diagramu na obrázku 1.3.



Obrázek 1.3: Diagram případů užití v aplikaci SimCom

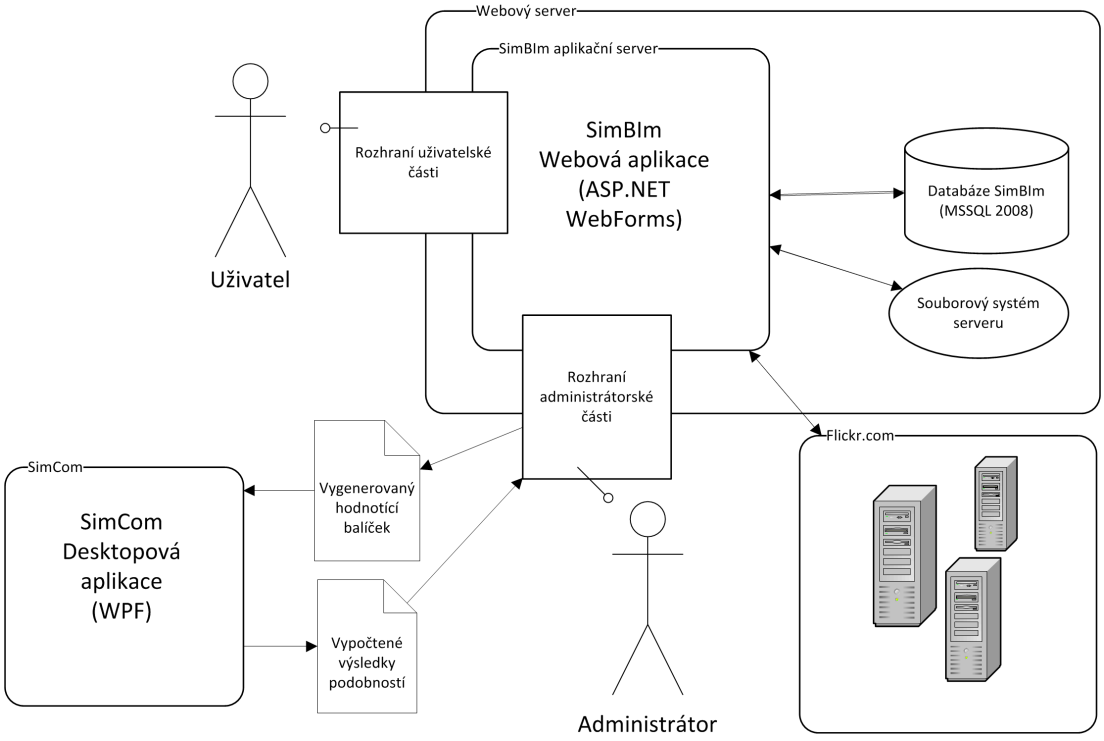
1.1.3 Interakce SimBIm a SimCom

Interakce mezi webovým serverem SimBIm a desktopovou aplikací SimCom neprobíhá přímo, zapojení uživatele je nutné. Celý proces funguje v následujících krocích:

- Ve webovém rozhraní SimBImu uživatel nechá vygenerovat hodnotící balíček
- V desktopové aplikaci SimCom uživatel načte stažený hodnotící balíček a spustí výpočet podobností
- Po dokončení výpočtu je uživateli vygenerován soubor obsahující vypočtená data

- Uživatel nahraje vygenerovaný soubor zpět na server SimBIm, který si načte vypočtená data

Celkový přehled architektury je znázorněn na obrázku 1.4. Jak je vidět, SimCom vůbec nekomunikuje přímo s databází, ani s jinými zdroji na serveru.

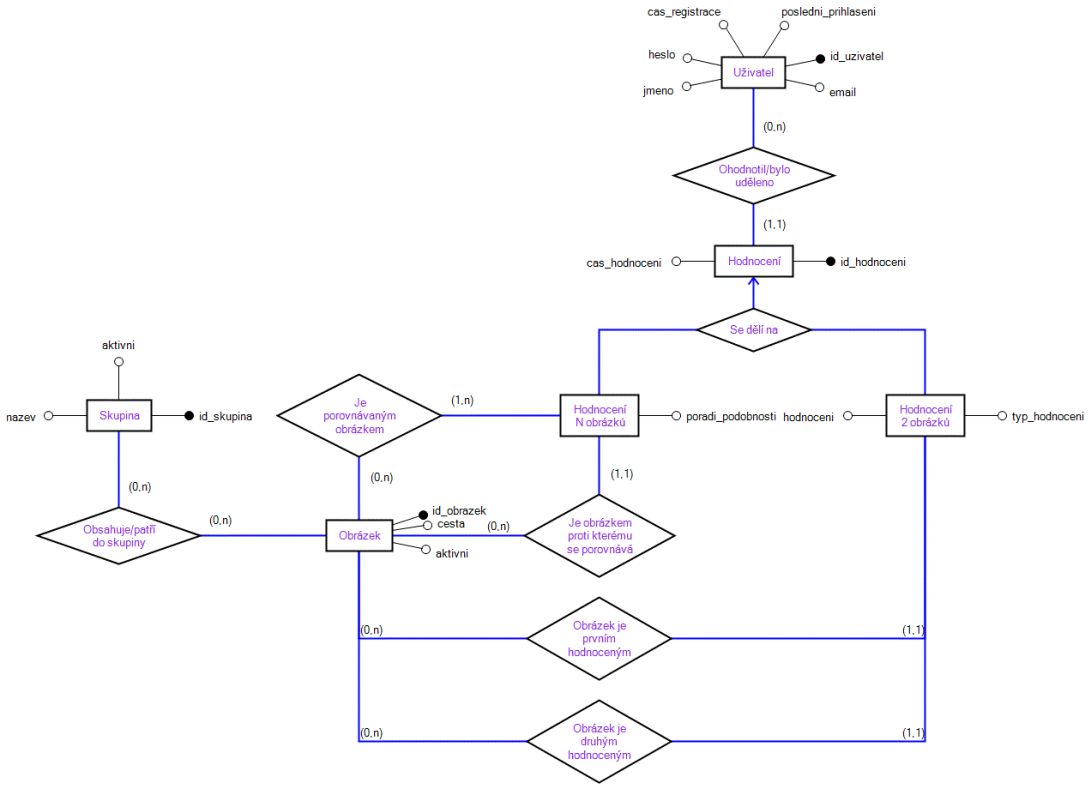


Obrázek 1.4: Celkový diagram popisující architekturu aplikací SimBIm, SimCom a jejich interakci

1.2 Databázový model pro SimBIm

Při návrhu databázového modelu jsem vycházel hlavně z faktu, že u uložených hodnocení budou používány operace vkládání a výběr dat, zatímco editace a mazání nebude vůbec potřeba (tyto operace nejsou z žádného případu užití k dispozici). Z tohoto důvodu jsem tabulky k ukládání hodnocení od uživatelů optimalizoval tak, aby bylo co nejjednodušší a nejrychlejší vkládání a dotazování nad daty. Datový model je možné normalizovat do 3NF, ale pro potřeby této aplikace postačuje i 2NF, kdy je možné ponechání částečné redundance dat, pokud zvolená struktura umožní snadnější a rychlejší dotazování. Není očekáváno tak veliké množství dat, aby místo zabrané částečnou redundancí dat mělo nezanedbatelný dopad.

Výsledný ER–model je zobrazen na obrázku 1.5. Pro zjednodušení nejsou v diagramu zobrazeny entity a vztahy sloužící k ukládání výsledků algoritmů, které jsou ale analogické entitám určeným k ukládání uživatelských hodnocení. Rovněž entity s nevýznamnou vazbou na zbytek (např. entita pro ukládání nastavení) entit nejsou zobrazeny.



Obrázek 1.5: ER-model

Některé vztahy z konceptuálního modelu nejsou příliš obvyklé databázové vztahy, bylo třeba zvážit možná řešení (kterých bylo většinou více), tak aby byl model dostatečně efektivní a snadno použitelný. Kompletní popis možností a odůvodnění použitých řešení je možné nalézt v přiložené analýze, výsledné databázové schéma je pak možné sestrojit pomocí SQL skriptu na přiloženém disku.

2. Implementace

V této kapitole je uveden základní přehled informací o implementaci aplikací SimBIm a SimCom, které byly analyzovány v předchozí části. Detailnější popis lze opět naleznout na přiloženém disku v dokumentacích k jednotlivým částem.

Obě aplikace jsou postavené na platformě Microsoft.NET, pro správnou funkčnost je požadován .NET Framework verze 4.0 a vyšší. Běh aplikací na jiných platformách než je Microsoft Windows není podporován.

2.1 SimBIm

SimBIm je ASP.NET WebForms aplikace a jako databázový server slouží Microsoft SQL Server (je podporován Microsoft SQL Server 2008 a novější).

2.1.1 Aplikační server

Aplikační server rozlišuje dva druhy uživatelů (běžné uživatele a administrátory), uživatelské účty jsou však propojené a rozlišování probíhá na základě přiřazování do rolí. To zajišťuje snadnou budoucí modifikovatelnost pro vytváření nových nebo modifikaci existujících rolí.

SimBIm komunikuje s okolním světem prostřednictvím dvou rozhraní. S uživateli a administrátory pomocí HTTP protokolu při použití uživatelského rozhraní založeném na XHTML, CSS a JavaScriptu. Pro komunikaci se serverem Flickr.com, který je používán jako možný zdroj pro načítání obrázků, pak je použita knihovna Flickr.Net [11], která zaobaluje volání funkcí Flickr API [12] do metod platformy .NET.

2.1.2 Databázová část

Pro přístup z prostředí .NET k datům SQL serveru je použita třída `SqlClient` z knihovny ADO.NET, dotazování probíhá pomocí jazyka SQL, žádné objektově–relační mapování není použité.

Co nejvíce databázové logiky je implementováno již přímo na SQL Serveru v jazyce T-SQL, jde hlavně o definici integritních omezení, uložené procedury a funkce pro složitější výpočty a triggery. Složitější SQL dotazy jsou v databázi definovány jako pohledy, tak aby z aplikace byla prováděna pouze volání uložených procedur a jednoduché SQL dotazy na databázové tabulky a pohledy. Podrobný popis všech databázových objektů je pak uveden v přiložené programátorské dokumentaci.

Některé hodnoty jsou náročné na výpočet, nejsou vypočítávány při každém uživatelském požadavku, ale vypočítané hodnoty jsou uložené v databázi a jejich aktualizace je provedena až po explicitním spuštění výpočtu. Jde ale ve všech případech o hodnoty, jejichž aktuálnost není vyžadována — např. odchylky hodnocení uživatelů nebo míra podobností hodnocení uživatelů a algoritmů.

2.1.3 Distribuce

Aplikace SimBIm je v rámci této práce distribuována ve dvou verzích.

- Čistá verze – neobsahuje žádná data, obrázky ani výsledky.
- Verze s nasbíranými daty – obsahuje nasbíraná a vytvořená data, která vznikla v průběhu této práce. Jde především o obrázky k porovnávání a jejich uživatelská hodnocení nebo vygenerované výsledky podobností. Uživatelské účty jsou v aplikaci ponechány, ale jsou anonymizovány (jsou odstraněna jména, e-mailové adresy a přihlašovací jména) a není je již možné používat. Jsou ponechány z důvodu zachování vazby mezi jednotlivými uživateli a udělenými hodnoceními.

2.2 SimCom

Grafické uživatelské rozhraní aplikace SimCom je naprogramované ve WPF, obsahuje jen 2 jednoduché obrazovky – jednu pro výpočet a druhou pro zobrazení dat. Výpočet je spouštěn ve vlákně na pozadí (pomocí třídy `BackgroundWorker`) a uživatel je informován o průběhu výpočtu ve stavovém řádku.

Detailní implementace výpočtu podobnosti obrázků bude postupně popsána v následujících kapitolách. Využívá se k tomu však knihovna EmguCV, která poskytuje implementaci všech důležitých dílčích částí výpočtu (hlavně detekci a mapování deskriptorů pomocí algoritmu SURF).

2.2.1 Hodnotící balíček a výsledky

Hodnotící balíček, který je vygenerován z aplikace SimBIm je ZIP archiv, který obsahuje tyto soubory:

- Soubor Info.txt obsahuje informace, které obrázky se mají spolu hodnotit (jde pouze o obrázky, které jsou ve stejných skupinách a které jsou aktivní pro hodnocení v době, kdy byl balíček v SimBIm vygenerován), jejich ID a jméno souboru.
- Obrázky ve střední velikosti (o šířce 425px), jejichž jména souborů jsou ve tvaru `middle_[jmeno_souboru_z_bodu1]`
- Obrázky v plné velikosti (šířka maximálně 1000px), jména souborů ve tvaru `big_[jmeno_souboru_z_bodu1]`

Důvodem, proč jsou v balíčku všechny obrázky ve dvou velikostech je možnost nastavení uživatelem, nad kterou velikostí obrázků bude výpočet probíhat. Pří použití obrázků ve střední velikosti je výpočet několikanásobně rychlejší, hlavně z důvodu menšího množství nalezených deskriptorů. Naopak při použití obrázků v plné velikosti jsou vypočtené výsledky přesnější, jak bude ukázáno v kapitole 6.2.

Výstupem hodnocení je několik (záleží na počtu nastavených výstupních metrik) souborů, jejichž obsah je vždy ve tvaru ID prvního obrázku;ID druhého obrázku;Výsledná hodnota sledované metriky. Tento výsledný soubor je pak možné nahrát zpět na server SimBIm, kde jsou výsledky načteny do databáze a porovnány s výsledky od uživatelů.

2.2.2 Modifikace algoritmu výpočtu

Je očekáváno, že uživatelé nebudou pouze počítat podobnosti pomocí metrik, které jsou implementovány v dodávané verzi k této práci, ale že budou chtít také modifikovat stávající algoritmy a metriky výpočtu nebo vyvíjet vlastní řešení výpočtu podobností. K tomu bylo přihlíženo při implementaci a proto je výpočet hodnocení oddělený v samostatné třídě. Tato třída musí implementovat rozhraní `IImageSimilarityProcessor`, které je velice jednoduché – obsahuje pouze 2 metody. První z nich vrací pole názvů použitých metrik a druhá pak pro předané obrázky výsledky těchto metrik spočítá.

Vzhledem k tomu, jak je aplikace i rozhraní triviální, neimplementoval jsem do aplikace technologii zásuvných modulů pro načítání externích knihoven na výpočet podobností. Proto, pokud chce uživatel upravit hodnotící algoritmy, musí upravit k této práci dodávaný projekt a zkompilovat si vlastní verzi aplikace. Vzhledem k jednoduchosti těchto úprav bude pro programátora modifikace dodávaného projektu pravděpodobně i rychlejší variantou než kdyby musel vytvářet vlastní knihovnu.

Ve verzi dodávané společně s touto prácí je výpočet implementován pomocí třídy `SurfSimilarityProcessor`. Výpočet je možné změnit modifikací této třídy, v případě, že bude vytvořená nová třída implementující potřebné rozhraní, je ještě třeba v souboru `MainWindow.xaml.cs` změnit hodnotu proměnné `imageSimilarityProcessor` na instanci nově vytvořené třídy.

2.3 Použité komponenty třetích stran

Při implementaci programu byly použity tyto knihovny a pluginy třetích stran:

- EmguCV [8], verze 2.3.0.1416 – wrapper pro .NET ke grafické knihovně Open CV psané v C++. Tato knihovna je použita pro extrakci a mapování SURF deskriptorů u výpočtu podobnosti obrázků v aplikaci SimCom.
- Dragsort [9], verze 0.4.3 – plugin do jQuery [10] přidávající podporu třídění objektů pomocí metody drag&drop. Tato funkcionalita je použita v aplikaci SimBIm u hodnocení v řežimu N obrázků pro řazení porovnávaných obrázků.
- Flickr.Net API Library [11], verze 3.2.4310 – knihovna pro .NET zapouzdřující komunikaci s Flickr API, která je použita v administraci SimBIm, kdy administrátor může vyhledat a nahrát do databáze obrázky ze serveru Flickr.com.
- Altairis Web Security [13], verze 2.3.1 – role a membership provider pro ASP.NET. Knihovna poskytuje alternativní sadu providerů pro autentizaci a správu rolí v ASP.NET. Je použita v aplikaci SimBIm, kde zajišťuje registraci, přihlašování, autorizaci uživatelů a jejich přiřazení do rolí.

3. Průběh experimentu

Pro potřeby této práce bylo potřeba od uživatelů nasbírat dostatečné množství hodnocení, vůči kterým pak budou porovnávány výsledky vypočtené pomocí algoritmů. V této kapitole budou popsány činnosti které předcházely před vypuštením aplikace mezi uživatele a také procedury, které jsou používané pro zpracování nasbíraných dat.

3.1 Výběr dat

Počátečním úkolem v experimentu byl výběr vhodných obrázků a jejich rozčlenění do skupin, jak je uživatelé budou hodnotit.

Rozhodl jsem se, že obrázky rozčlením do skupin a budou se hodnotit vzájemně jen v rámci skupiny a to hlavně ze dvou důvodů:

- Chtěl jsem výsledky otestovat na různých tématických skupinách obrázků a uživatelské hodnocení mezi skupinami by většinou nevedlo k žádným výsledným podobnostem (mezi obrázkem moře a sportovního auta bude málokdy něco podobného).
- Počet hodnocení potřebných k ohodnocení všech dvojic v jedné velké skupině je mnohem větší než v případě ohodnocení všech dvojic v rámci několika skupin — např. pokud bude jedna skupina s 90ti obrázky, je potřeba více něž 3krát více hodnocení než pokud budou tři skupiny po 30ti obrázcích. Obával jsem se, že se mi nepodaří nasbírat dostatečný počet kvalitních uživatelských hodnocení.

Vytvořil jsem tedy 3 následující skupiny obrázků:

- Příroda (25 obrázků)
- Praha (31 obrázků)
- Auta (22 obrázků)

Jako zdroj obrázků jsem použil server Flickr.com, vyhledání i nahrání do databáze bylo provedeno přímo z aplikace pomocí vestavěného rozhraní pro přístup k datům serveru Flickr. Vybírání vhodných obrázků nebylo úplně náhodné, snažil jsem se vybrat takovou kolekci, aby se daly jisté podobnosti najít, ale zase aby nebyla hodnocení úplně zřejmá — aby pokrývala optimálně celou škálu od 0 do 100% podobnosti.

U obrázků ve skupině Auta jsem předpokládal obtížnou hodnotitelnost (očekával jsem, že tam mohou hrát roli věci jako jsou typ auta a jeho výrobce, barva vozidla, pozadí fotografie a další, zatímco např. u fotek přírody není hlavních ukazatelů tolik), u té jsem předem očekával jak nepříliš konzistentní hodnocení od uživatelů, tak i horší výsledky u porovnání pomocí algoritmů, bral jsem ji tedy spíše jako doplňkovou k prvním dvěma skupinám. Proto jsem i některé statistiky vyhodnocoval dvakrát — s aktivní skupinou Auta i bez ní, abych se ujistil, že nemá na celkové hodnocení výrazný vliv.

3.2 Sběr a úprava dat

Několik měsíců byla aplikace volně přístupná na internetu na adrese `http://simbim.aspone.cz` , kde uživatelé mohli hodnotit podobnost obrázků. Většina uživatelů byli moji přátelé, spolužáci a známí, ale i několik lidí, které jsem neznal, se zapojilo do hodnocení. Uživatelé pak ke dni 22.5.2012 nashromáždili celkem:

- 2627 hodnocení dvojic obrázků v režimu ano/ne
- 1988 hodnocení dvojic obrázků v režimu 0–100%
- 748 hodnocení N–tic obrázků

Po získání tohoto množství hodnocení bylo ještě třeba provédst vhodné zpracování a vyhodnocení nasbíraných dat.

3.2.1 Čištění dat

Vzhledem k tomu, že aplikace byla volně přístupná na internetu a mohl se do ní zaregistrovat kdokoli, bylo před interpretováním získaných dat nutné ověřit, že někteří uživatelé nehodnotili záměrně špatně aby nedošlo k znehodnocení ostatních naměřených dat.

U některých uživatelů nebyl problém toto záškodnictví odhalit (např. již podle sprosté přezdívky uvedené při registraci nebo jen prostým pohledem na výsledky hodnocení), ale pro celkovou analýzu hodnocení bylo potřeba vytvořit nějaký měřitelný ukazatel. Vzhledem k tomu, že hodnocení jednotlivých uživatelů jsou subjektivní a většinou neexistuje žádná správná odpověď, není možné porovnávat hodnocení proti nějakému statickému souboru odpovědí.

Rozhodl jsem se proto pro implementaci odchylky hodnocení daného uživatele od průměrných hodnocení ostatních uživatelů. Odchylku jsem spočítal zvlášť jak pro hodnocení v režimu ano/ne tak v režimu 0-100%, u hodnocení N obrázků jsem odchylku nepočítal z důvodu obtížné volby výpočetní funkce i z důvodu menšího statistického souboru dat.

Funkce pro výpočet odchylky vypadá takto:

$$D(U) = \sqrt{\frac{\sum_{i=1}^{n} (O_i - U_i)^2}{n}}$$

Kde U_i je průměrné hodnocení i-té dvojice obrázků uživatele U, O_i je průměrné hodnocení i-té dvojice od ostatních důvěryhodných uživatelů a n je celkový počet dvojic, které mají alespoň jedno hodnocení od uživatele U a alespoň jedno hodnocení od jiného uživatele. Výpočet je implementován ve funkci SQL Serveru s názvem `dbo.ComputeUserRatingsDeviation`.

Odchylky jsou uložené v databázi jako vypočtené hodnoty, nejsou tedy vždy aktuální. Aktualizaci hodnot je možné provést v administraci v sekci Nástroje kliknutím na Přepočítat odchylky u hodnocení uživatelů, kde je rovněž možné nastavit maximální hodnotu odchylky pro kterou jsou data považována za relevantní. Vzhledem k tomu, jak je výpočet navržen nemusí být po prvním přepočítání odchylky přímo na výsledných hodnotách, vypočítané hodnoty mohou

k výsledné hodnotě pouze postupně konvergovat, proto je vhodné při velké změně zdrojových dat nebo při přenastavení maximální odchylky spustit přepočet 2-3krát po sobě.

Výsledky vypočítaných odchylek bych zhodnotil jako relevantní, uživatelům které znám a věřím, že hodnotili pocitivě vyšla výsledná odchylka nízká (zhruba 25 a méně), zatímco podezřelí uživatelé dosahovali odchylek až 80. Maximální odchylku, pro kterou je možné považovat hodnocení za relevantní bych volil v intervalu 40-50. Dále v této práci budu používat hodnotu 45. Zároveň je ale možné, že i uživatel, který prováděl hodnocení v dobrém úmyslu je vyhodnocen jako nedůvěryhodný, což může být způsobeno buď jistou odlišností ve stupnici jeho hodnocení nebo malým statistickým vzorkem uživatelových hodnocení. Ale vzhledem k tomu, že data kolem zvolené hranice nepředstavují významnou část nasbíraných dat, jejich nesprávným ignorováním nebo naopak zapojením by neměla být narušena ostatní nasbíraná hodnocení.

Od teď až do konce práce pokud bude zmíněna množina získaných hodnocení bude implicitně uvažována pouze množina důvěryhodných hodnocení, nedůvěryhodná budou z této množiny vynechána.

Po odstranění (fyzicky ale nedochází k žádnému mazání dat z databáze, pouze jsou dotazy kladeny nad pohledy, které obsahují pouze hodnocení od důvěryhodných uživatelů – `TrustedYesNoRatings` a `TrustedPercentageRatings`) nedůvěryhodných uživatelských hodnocení zůstalo v databázi celkem:

- 2046 hodnocení dvojic obrázků v režimu ano/ne
- 1675 hodnocení dvojic obrázků v režimu 0–100%
- 721 hodnocení N–tic obrázků

Tedy 82,8 procent ze všech nasbíraných dat bylo označeno jako důvěryhodná hodnocení.

4. Výpočet podobnosti obrázků

Jak již bylo naznačeno v úvodu, dalším krokem práce je implementace algoritmu pro výpočet podobnosti mezi obrázky. V této práci bylo úkolem zhodnotit výsledky vytvořené na základě algoritmů SIFT (Scale Invariant Feature Transform, [1]) nebo SURF (Speeded Up Robust Features, [2]), které jsou v současné době pravděpodobně nejpoužívanějšími algoritmy pro detekci význačných bodů a jejich popis pomocí deskriptorů. V poslední době převažuje spíše použití algoritmu SURF, který není zatížen patenty a je výrazně rychlejší než SIFT, což je důležité například při zpracování obrazu v reálném čase.

Existují i další algoritmy pro detekci a popis deskriptorů (např. GLOH [5]), které dosahují lepších výsledků pro některé specifické typy obrazových transformací, v celkovém měřítku však pravděpodobně nejsou výrazně lepší než zmiňovaný SIFT a SURF, proto jsem jejich užití nyní nezvažoval, do budoucna by ale bylo jistě zajímavé porovnat vliv algoritmu pro detekci deskriptorů na výsledky spočítaných výsledných podobností.

Kvalita výsledků algoritmů SIFT a SURF je srovnatelná [4], proto jsem se v této práci na základě citovaných porovnání rozhodl pro použití algoritmu SURF.

V této kapitole bude uveden stručný teoretický základ fungování algoritmu SURF postačující pro potřeby této práce, podrobný popis fungování popisují autoři v článku, kde byl algoritmus představen [2] a širší základy problematiky význačných bodů a deskriptorů jsou dobře popsány např. v bakalářské práci P. Bílka [6].

4.1 Úvod do problematiky

Oba zmíněné algoritmy (SIFT a SURF) fungují v těchto krocích:

- Detekce význačných bodů v daném obrázku
- Popis nalezených význačných bodů – vytvoření deskriptorů
- Mapování deskriptorů mezi jednotlivými obrázky

Význačné body v obrázku jsou důležité pro počítačové zpracování obrazu. Měly by být snadno detekovatelné, mělo by je být možné naleznout i po působení změn v obrázku (rotace, změna měřítka, osvětlení atd.). Jejich opakovatelná detekce je ale samozřejmě závislá i na kvalitě detektoru. A význačnost bodu se projevuje bohatostí okolí bodu na informace, které jsou použitelné pro pozdější zpracování obrazu.

Pro každý nalezený význačný bod je vytvořen jeden deskriptor. Tyto deskriptory popisují okolí význačných bodů a mají co nejjednoznačněji identifikovat popisující nalezený bod a jeho okolí, mají být co nejvíce nezávislé na posunutí, změně velikosti, osvětlení nebo úhlu pohledu. Deskriptory mají rovněž být charakteristické, což znamená, že stejné nebo velice podobné body mají přiřazeny podobné deskriptory a naopak odlišné body mají lišící se deskriptory, tak aby byla možná detekce odpovídajících si objektů s vysokou pravděpodobností úspěchu

i ve velké kolekci obrázků a naopak aby bylo zabráněno nesprávné identifikaci neodpovídajících si objektů.

Porovnáváním jednotlivých deskriptorů je pak možné najít stejné nebo podobné body v jiném obrázku, což je stěžejní část algoritmu pro potřeby této práce.

4.2 Scale–space

Scale–space je reprezentace obrázku nezávislá na měřítku, která se používá v algoritmu SURF pro zajištění nezávislosti na změně velikosti obrázku. Jde o 3D reprezentaci původního obrázku, kde informace v třetí dimenzi je určena měřítkem. Měřítko pak určuje jaká úroveň detailů je zachována. Vrstvy s malým měřítkem obsahují vysokou úroveň detailů, vrstvy s větším měřítkem pak nižší.

Bylo ukázáno, že pro převod do scale–space je vhodná Gaussova funkce, která pro dvě proměnné x a y a měřítko σ vypadá následovně:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

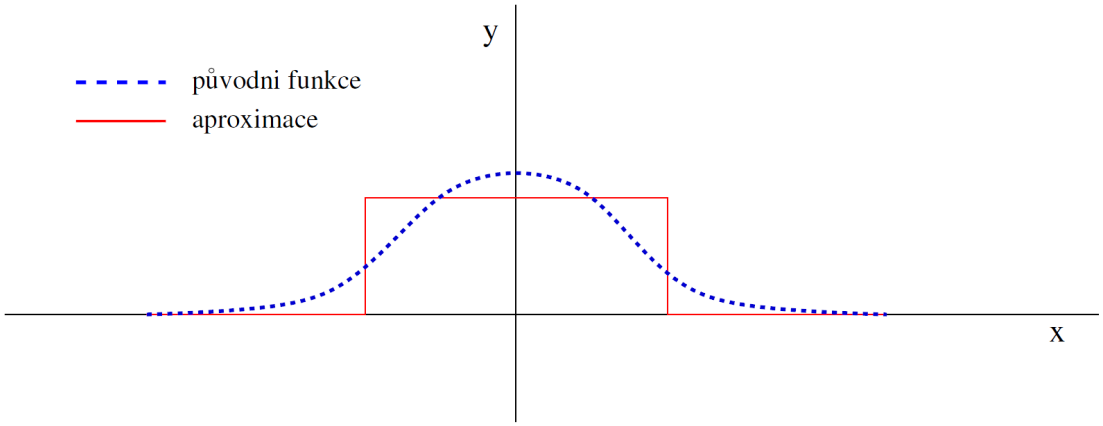
Konvolucí této funkce s obrázkem I pak může vzniknout vrstva scale–space L s měřítkem σ.

Vyhledávání význačných bodů se ale v praxi provádí pomocí konvoluce druhé derivace Gaussovy funkce, která také generuje scale–space, ale výsledkem jsou změny intenzit v daném obrázku. Vzhledem k použití druhé derivace nejsou ale extrémy detekovány v místech největší změny, ale v jejich okolí, a to tak, že se vzrůstajícím měřítkem se posouvají dále od počátku souřadnicového systému. Tento výpočet je pak v praxi realizován pomocí Laplaciánu Gaussovy funkce, který je definován jako:

$$LoG = \frac{\partial^2}{\partial x^2} G(x, y, \sigma) + \frac{\partial^2}{\partial y^2} G(x, y, \sigma)$$

4.2.1 Aproximace LoG

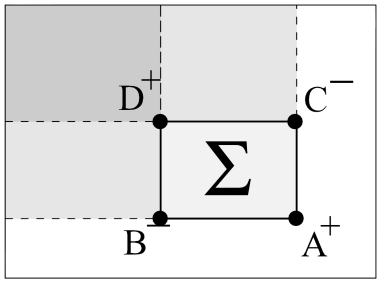
Protože by výpočet přesné hodnoty Laplaciánu byl příliš pomalý, používá se pouze jeho aproximace. Jednou možností aproximace je rozdíl Gaussových funkcí, jak ji uvedl Lowe [1], SURF ale používá poměrně hrubou aproximaci pomocí obdélníkových funkcí, ale bylo experimentálně ukazáno, že její výsledky naopak předčí i většinu ostatních aproximací.



Obrázek 4.1: Příklad aproximace funkce obdélníkovou funkcí. Převzato z [6]

4.2.2 Integrální obrázek

Společně s uvedenou aproximací se často používá struktura integrálního obrázku. Ten má stejné rozměry jako původní obrázek, ale hodnota v libovolném bodě obrázku x je určena jako suma bodů nacházející se v obdélníku, jehož levým horním rohem je levý horní roh původního obrázku a pravým dolním rohem je bod x.



Obrázek 4.2: Integrální obrázek, součet bodů v oblasti Σ je pak snadno vypočten jako $A + D - B - C$. Převzato z [6]

Integrální obrázek je používán pro zjištění součtu hodnot všech bodů uvnitř libovolného obdélníku v obrázku, kdy jsou pro tento výpočet potřeba pouze 3 celočíselné operace. Konstrukce integrálního obrázku je možná v lineárním čase díky používání již dříve spočítaných hodnot.

4.3 Detekce význačných bodů

Scale–space je u algoritmu SURF generováno pomocí determinantu Hessovy matice

$$L(x, y, \sigma) = \sigma^4 Det \begin{pmatrix} I_{xx}(x, y, \sigma) & I_{xy}(x, y, \sigma) \\ I_{xy}(x, y, \sigma) & I_{yy}(x, y, \sigma) \end{pmatrix}$$

kde $L(x, y, \sigma)$ je bod scale–space se souřadnicemi x, y a měřítkem σ. $I_{xx}(x, y, \sigma)$ je pak konvoluce $\frac{\partial^2}{\partial x^2} G(\sigma)$ s obrázkem I v bodu (x, y), analogicky pak i pro I_{xy} a I_{yy}.

Díky integrálním obrázkům a použití popsané aproximace pomocí obdélníkových funkcí je pak výpočet velice rychlý.

Samotné význačné body jsou pak detekovány porovnáváním bodu se všemi sousedy ve scale–space (zkoumaný bod je středem krychle $3 \times 3 \times 3$) a pokud nabývá maxima, je detekován jako význačný.

4.3.1 Určení orientace

Další fází je přiřazení orientace nalezenému význačnému bodu. To je prováděno k zajištění nazávislosti na rotaci obrázku.

K určení orientace se používají informace z kruhu o poloměru 6σ okolo nalezeného klíčového bodu. Proměnná σ zde představuje hodnotu měřítka scale–space, ve které byl bod detekován. Jak ve směru osy x, tak ve směru osy y jsou spočítány pro všechny body obrázku gradienty (dx, dy). Dále se vytvoří 6 okének každé o úhlové šířce 60 stupňů a v každém okénku se vytvoří suma ze všech gradientů $(\sum dx, \sum dy)$ bodů v okénku obsažených. Největší výsledný vektor ze všech okének je pak vybrán jako orientace významného bodu.

4.4 Vytvoření deskriptorů

U popisovaného algoritmu SURF jsou deskriptory n–dimenzionální vektory, typicky se používají 64 dimenzionální vektory s jednotkovou délkou. Je možná i varianta se 128 složkami, ale tato varianta není příliš používána, jelikož je mapování těchto deksriptorů pomalejší a nepřináší výrazné zlepšení přesnosti výsledků [2]. Dále tedy v této práci bude používán 64–dimenzionální deskriptor.

Pro výpočet deskriptoru se používá čtvercové okolí se stranou délky 20σ se středem v detekovaném význačném bodě. Tento region je natočen podle vypočtené orientace klíčového bodu. Okolí je dále rozděleno na 16 oblastí, kde každá má velikost $5\sigma \times 5\sigma$, v každé oblasti se vybere 5 pravidelně rozmístěných bodů a pro ně jsou spočteny hodnoty (dx, dy), které jsou ještě váženy Gaussovou funkcí. Pro každou z oblastí je vypočten subdeskriptor, který je tvořen vektorem $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Posledním krokem je pak již jen spojení těchto subdeskriptorů ze všech oblastí do jednoho 64-dimenzionálního vektoru, který tvoří výsledný deskriptor.

4.5 Mapování deskriptorů

Jak získat deskriptory z obrázku již bylo ukázáno, pro porovnání podobností dvou obrázků ale je potřeba vyhodnotit získané deskriptory z obou obrázků. Podobnost mezi deskriptory pak je určena vzdáleností mezi vektory, které tvoří deskriptory. K mapování prvně potřebujeme pro libovolnou dvojici deskriptorů (každý z jiného obrázku) zjistit vzájemnou vzdálenost a určit podmínky, kdy je tato dvojice označena jako namapovaná.

4.5.1 Metrika a výpočet vzdálenosti

Jelikož je mapování deskriptorů založené na hledání nejbližších deskriptorů, je nejprve nutné zvolit vhodnou metriku. Většinou se používá klasická Euklidovská vzdálenost, tedy pokud máme dva n–dimenzionální vektory p a q, kde p_i (resp. q_i) představuje i–tou složku vektoru p (resp. q), pak je $d(p, q)$ vzdálenost mezi těmito vektory definována následovně:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Jinou vhodnou metrikou může být např. Manhattanská vzdálenost, která je definována takto:

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

Dalším krokem je pomocí zvolené metriky nalézt pro všechny deskriptory z prvního obrázku nejbližší dva deskriptory z obrázku druhého. Pokud máme v prvním obrázku m a v druhém n deskriptorů, lze snadno nahlédnout, že nalezení dvou nejbližších deskriptorů lze provést v čase $O(m \cdot n)$ počítáním vzdáleností mezi všemi dvojicemi vektorů. Na to nám stačí $O(max\{m, n\})$ paměti. Při použití chytřejších algoritmů a vhodného indexování [3] lze výpočet provést v lepším čase, tím jsem se však v této práci nezabýval, protože zde je důležitá především přesnost výpočtu, jeho zrychlení může být v budoucnu implementováno jako zlepšení práce.

Nyní pokud máme spočítané vzdálenosti, můžeme přistoupit k výběru dvojic deskriptorů, které mohou být namapované — což znamená, že pravděpodobně v obou obrázcích představují stejný nebo velice podobný význačný bod.

Zde se nejčastěji používá algoritmus, který vybere ty deskriptory z prvního obrázku, jejichž poměr vzdáleností nejbližšího a druhého nejbližšího deskriptoru z druhého obrázku je menší než zvolená konstanta. Tedy nechť je x libovolný deskriptor z prvního obrázku a nechť y_1 a y_2 jsou k x dva nejbližší deskriptory z druhého obrázku a platí $d(x, y_1) < d(x, y_2)$. Pak dvojice deskriptorů (x, y_1) může být namapována, pokud platí:

$$\frac{d(x,y_1)}{d(x,y_2)} \leq c$$

Kde c je parametr z intervalu $(0, 1)$. Doporučená hodnota pro c je $0,8$ jak prezentuje Lowe [1] — při této hodnotě ukazuje, že je odstraněno 90% nesprávně namapovaných párů, zatímco pouze 5% správných dvojic, které by měly být namapované, je vyřazeno. Tato hodnota byla ale prezentována v souvislosti s algoritmem SIFT, většina implementací SURF ji však také doporučuje vzhledem ke stejné implementaci mapování deskriptorů. Zároveň ale i některá literatura [7] používá a doporučuje jiné hodnoty, proto jsem se rozhodl v sekci 6.1 vyzkoušet, jaká hodnota bude optimální pro použití při výpočtu podobností. Navíc je i možné, že optimální hodnota pro hledání podobnosti a při použití v jiných aplikacích se může lišit.

4.5.2 Orientace a velikost deskriptorů

Další informace, které byly deskriptoru (v tomto případě přesněji význačnému bodu, ale ten je možné také pokládat za jednu ze složek deskriptoru) přiřazeny je jeho orientace a velikost. Tyto hodnoty je možné použít k dalšímu odfiltrování dvojic, které nemají být namapovány. Tento test na orientaci a velikost snižuje celkový počet namapovaných deskriptorů. Jestli je pro potřeby vyhledávání podobných obrázků žádoucí odfiltrovat deskriptory s rozdílnou orientací nebo to naopak v těchto případech škodí bude rozebráno při porovnávání výsledků.

4.5.3 Symetrické a nesymetrické mapování

Zatím v této sekci u mapování deskriptorů byly uvažovány pouze případy, že máme jeden daný obrázek, z něho vybereme nějaký deskriptor a v druhém obrázku hledáme nejbližší deskriptory a jejich vzdálenosti. Tento přístup můžeme nazvat nesymetrický a je většinou používán, pokud hledáme výskyt daného objektu (v tomto případě první obrázek) v nějakém jiném obrázku — deskriptory z prvního obrázku namapujeme na deskriptory druhého obrázku a oblast kde je jejich dostatečný výskyt pak pravděpodobně obsahuje hledaný objekt.

Pro použití vyhledávání podobnosti mezi libovolnými obrázky se ale spíše hodí symetrické porovnávání. To spočívá jak v mapování deskriptorů z prvního obrázku na deskriptory obrázku druhého, tak i obráceně. Vzdálenosti mezi jednotlivými dvojicemi deskriptorů sice zůstávají stále stejné, ale proces mapování v obou případech nalezne většinou jiné dvojice deskriptorů.

Máme–li nějaké deskriptory a_1, a_2 z prvního obrázku a deskriptory b_1, b_2 z obrázku druhého, pak je a_1 namapován na b_1 pokud:

$$\frac{d(a_1, b_1)}{d(a_1, b_2)} \leq c$$

Ovšem pokud budeme provádět mapování obráceně, tak např. pro $d(b_1, a_1) = d(b_1, a_2)$ nebude pro $c < 1$ dvojice (a_1, b_1) vůbec namapována.

Proto budou při porovnávání výsledků uvažovány možnosti, kdy bude mapováni provedeno oběma směry. V tom případě je ale možné uvažovat dvě možné interpretace.

První možností je, že dílčí výsledky obou mapování budou oddělené, pouze výsledný ukazatel bude sestaven z obou hodnot. Např. pokud budeme uvažovat jako metriku počet namapovaných deskriptorů a m_1 bude výsledek pro mapování jedním směrem a m_2 směrem druhým, pak výsledná hodnota metriky bude $m_1 + m_2$. Tato varianta bude dále pro potřeby této práce nazývána jako mapování v režimu OR.

Druhou možností je uvažovat výsledky metriky pouze na dvojicích deskriptorů, které byly namapovány zároveň v obou směrech, tedy pokud při prvním mapování je výsledkem množina dvojic deskriptorů $\{(a_1, b_1), (a_1, b_2), (a_1, b_5), (a_2, b_1)\}$ a výsledná množina druhého mapování je $\{(a_1, b_1), (a_2, b_1), (a_1, b_3)\}$, pak namapované deskriptory jsou jen ty z průniku obou množin, tedy pouze $\{(a_1, b_1), (a_2, b_1)\}$. Tímto krokem ale klesne množství namapovaných deskriptorů a to poměrně výrazně, jak bude ukázáno v sekci 6.1. Tento způsob je pak dále v této práci nazýván jako mapování v režimu AND. Pokud zvolená metrika je založená na poměrech vzdáleností mezi nejbližším a druhým nejbližším deskriptorem, je pak vhodné

ještě zkombinovat hodnoty z obou mapování, jelikož v každém směru jsou výsledky jiné. U vzdáleností není třeba nic upravovat, ty se díky axiomu symetrie metrického prostoru nemění.

4.6 Získání výsledné podobnosti

Během procesu detekce a mapování deskriptorů jsme získali mnoho hodnot, které mohou sloužit k výpočtu míry podobnosti mezi danými obrázky. Konkrétně lze použít např. tyto hodnoty:

- Počet namapovaných deskriptorů
- Poměr počtu namapovaných deskriptorů ku počtu všech nalezených deskriptorů
- Minimální/průměrná/maximální vzdálenost mezi namapovanými deskriptory
- Poměr mezi nejbližším a druhým nejbližším deskriptorem

Výsledky kterých z těchto hodnot (a jejich kombinací) jsou nejbližší uživatelskému vnímání podobnosti obrázků bude ukázáno dále v této práci v kapitole 6.

5. Metodika vyhodnocení a sledované ukazatele

Pro porovnávání výsledků uživatelského hodnocení s výsledky spočítanými algoritmy je potřeba stanovit ukazatele, které vypočítají korespondenci mezi těmito hodnoceními a stanoví správnost výsledků jednotlivých metrik.

5.1 Příprava dat

Jelikož se uživatelská hodnocení sbírají ve třech různých režimech, je nejprve třeba stanovit způsob jaká uživatelská hodnocení se budou používat. Rozhodl jsem se pro spojení jednotlivých hodnocení od uživatelů a vytvoření souhrnného ukazatele, který pak bude používán při porovnávání výsledků.

5.1.1 Vytvoření souhrnného ukazatele

Souhrnný ukazatel spojuje data z obou druhů uživatelského hodnocení dvojic obrázků (režim ano/ne a procentuální hodnocení). Výsledky z hodnocení v režimu ano/ne mají malou granularitu (např. rozsah pro hodnocení "ano" může být pro nějaký obrázek hodnota 50-100% pokud by uživatel hodnotil v režimu procentuálního hodnocení), proto jsem se rozhodl souhrnný ukazatel založit hlavně na výsledcích z procentuálního hodnocení.

Vzorec pro výpočet tohoto kumulovaného hodnocení pro dvojici obrázků I je následující

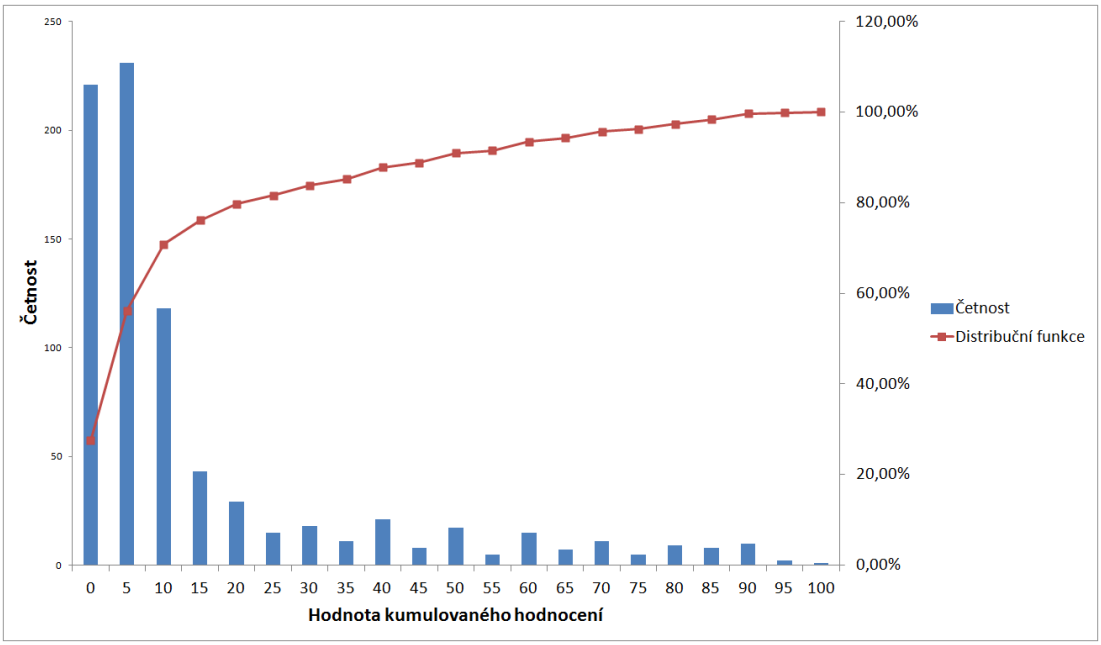
$$CR(I) = \frac{Avg_P(I) * (1 + \frac{Avg_{YN}(I)-50}{100})}{1,5}$$

pokud existuje pro danou dvojici obrázků I alespoň jedno hodnocení v režimu ano/ne (což v nasbírané množině dat existuje téměř pro všechny dvojice), jinak je následující

$$CR(I) = \frac{Avg_P(I) * (1 + \frac{Avg_P(I)-50}{100})}{1,5}$$

Kde $Avg_P(I)$ znamená průměrné procentuální hodnocení pro dvojici obrázků I v procentuálním režimu hodnocení a $Avg_{YN}(I)$ je procentuální poměr hodnocení "ano" ze všech hodnocení v režimu ano/ne pro danou dvojici obrázků I.

Základem je tedy průměrné procentuální hodnocení, výsledky z hodnocení v režimu ano/ne můžou tuto hodnotu snížit nebo zvýšit maximálně o 50% (snížit pokud budou všechna hodnocení "ne", zvýšit v případě varianty pouze samých kladných hodnocení). Tím se hodnoty dostanou do intervalu $[0, 150]$, aby výsledná hodnota byla z intervalu $[0, 100]$ byl ještě celý výraz vydělen číslem $1,5$. Histogram a distribuční funkce této veličiny jsou zobrazeny na obrázku 5.1. Hodnoty tohoto souhrnného hodnocení je možné z databáze získat ve sloupci `CumulativeRating` v pohledu `dbo.TwoImagesRatingsResults`.



Obrázek 5.1: Histogram a distribuční funkce veličiny kumulované hodnocení. Jak je vidět, velká část dvojic obrázků si není vůbec podobná

5.1.2 Rozdělení na skupiny

Pro další využití tohoto kumulovaného hodnocení jsem se rozhodl vytvořit 3 segmentační skupiny, do kterých bych všechny dvojice rozřadil. Zde jsem se rozhodl určit hranice konkrétními hodnotami kumulovaného hodnocení a to následovně:

- Naprosto odlišné obrázky (interval hodnot 0 až 10)
- Mírně podobné obrázky (interval hodnot 10 až 50)
- Podobné obrázky (interval hodnot 50 až 100)

Kvantily pro uvedené hranice nabývají hodnot $0,708$ a $0,909$. Podobných dvojic je tedy pouze 9,1% ze všech dvojic, naopak téměř 71% dvojic je velice odlišných. Nastavení těchto hranic je možné změnit v administraci aplikace SimBIm, po změně je ale třeba nechat explicitně přepočítat data, která tato hodnota ovlivňuje.

5.2 Porovnávání výsledků a sledované ukazatele

Hlavním úkolem celé bakalářské práce bylo porovnání uživatelského vnímání podobnosti obrázků s výsledky, které jsou spočítané algoritmicky pomocí různých technik a určení míry shodnosti mezi těmito hodnoceními.

K tomuto účelu jsem vytvořil několik jednoduchých metrik, které určují míru shodnosti hodnocení uživatelů a výsledků algoritmicky spočítaných a umožní porovnat různé metriky.

5.2.1 Skóre shodnosti v 5ti nejpodobnějších obrázcích

Tento ukazatel je založen na porovnání 5ti nejpodobnějších obrázků podle uživatelských hodnocení a podle algoritmu.

Z databáze je získáno pro daný obrázek 5 nejpodobnějších obrázků (včetně koeficientu podobnosti, kterým je zde kumulované hodnocení popsané v sekci 5.1.1) podle uživatelů a 5 nejpodobnějších obrázků podle algoritmu. Maximálním hodnocením, které je možné dosáhnout je součet koeficientů všech 5ti nejpodobnějších obrázků podle uživatelů. Získaným hodnocením pak je pouze suma koeficientů obrázků, které se vyskytují společně v obou pěticích (na pořadí výskytů nezáleží).

Pokud by pro daný obrázek neexistovalo 5 obrázků s nenulovým výsledkem od algoritmu, pak bude snížen počet vybíraných obrázků (tedy nebude se vybírat 5 ale např. pouze 3 obrázky podle obou hodnocení, pokud by neexistoval žádný, pak se hodnocení u tohoto obrázku přeskočí). Toto opatření bylo učiněno aby metoda fungovala i při ohodnocování výsledků algoritmů, kde je detekováno pouze velice málo podobností.

Takto je výpočet proveden pro všechny obrázky a celkovým hodnocením pak je

$$Top5HitsPercents = \frac{\sum_{i=1}^{n} R_{O_i}}{\sum_{i=1}^{n} M_{O_i}} * 100$$

kde O_i je i–tý obrázek v databázi, a hodnoty R_{O_i} a M_{O_i}, které představují získané a maximální skóre pro obrázek O_i, jsou vypočteny pomocí následujícího algoritmu:

Algoritmus 1 Výpočet hodnot R_{O_i} a M_{O_i}. Funkce GetMSIAccordingToAlgorithm vrací pro obrázek O_i požadovaný počet nejpododobnejších obrázků podle hodnocení algoritmu, GetMSIAccordingToUsers analogicky, ale řadí podle uživatelských hodnocení

```
AlgorithmMSI ← GetMSIAccordingToAlgorithm(O_i, 5)
ImagesCount ← 0
for all Image ∈ AlgorithmMostSimilarImages do
  if Image.Rating ≠ 0 then
    ImagesCount ← ImagesCount + 1
  end if
end for
AlgorithmMSI ← GetMSIAccordingToAlgorithm(O_i, ImagesCount)
UsersMSI ← GetMSIAccordingToUsers(O_i, ImagesCount)
MaxRating ← 0
ReachedRating ← 0
for all Image ∈ UsersMostSimilarImages do
  MaxRating = MaxRating + Image.CumulativeRating
  if Image ∈ AlgorithmMostSimilarImages then
    ReachedRating = ReachedRating + Image.CumulativeRating
  end if
end for
return ReachedRating, MaxRating
```

Příklad výpočtu skóre: Máme–li pro nějaký obrázek I tyto výsledky:

ID Obrázku	Hodnocení
10	95
12	87
2	79
29	77
21	62

Tabulka 5.1: 5 nejpodobnějších obrázků podle uživatelů

ID Obrázku
2
21
7
34
10

Tabulka 5.2: 5 nejpodobnějších obrázků podle algoritmu

Pak maximální hodnocení je zde suma hodnocení pěti nejpodobnějších dvojic, v tomto případě tedy $95 + 87 + 79 + 77 + 62 = 400$.

Získané hodnocení je ale jen součet hodnocení obrázků, které se vyskytují v obou množinách, tedy $79 + 62 + 95 = 236$.

Pokud by stejné hodnoty vyšly pro všechny obrázky v databázi, pak by výsledné skóre pro dané algoritmické hodnocení bylo 59%.

Tento ukazatel tedy hodnotí, jak se algoritmu podařilo najít nejpodobnější obrázky ke všem obrázkům v databázi. Vzhledem k tomu, že jsou výsledky váženy hodnotou kumulovaného hodnocení od uživatelů, je nalezení obrázku s vysokým uživatelským hodnocením ohodnoceno nejvíce body, naopak pokud není nalezen obrázek, který má malé hodnocení, je penalizace nízká.

Výpočet této hodnoty je implementován uvedeným algoritmem v databázové proceduře `ComputeRatingsResultsScore`. Spočítaná data jsou uložená v databázi a nejsou aktualizována dokud není z administrace učiněn explicitní požadavek k přepočítání.

I pokud jsou do systému nahrána hodnocení přiřazující podobnost naprosto náhodně, v tomto ukazateli získají většinou tato hodnocení nenulové skóre. Vzhledem k tomu, že hodnota skóre závisí na aktuálních výsledcích od uživatelů, aktuální velikosti a nastavení kolekce obrázků, není možné skóre pro náhodné ohodnocení spolehlivě matematicky odhadnout, proto jsem provedl, nad v době psaní této práce aktuálními daty, několik měření a průměrná hodnota skóre u těchto měření byla 20%.

Proto abychom mohli nějaké vypočítané hodnocení považovat za relevantní, mělo by v této metrice dosahovat výrazně vyšších hodnot než náhodné ohodnocení.

5.2.2 Skóre založené na správnosti segmentace

V tomto vyhodnocení správnosti je využito rozčlenění do skupin, jak bylo popsáno v sekci 5.1.1. Mějme tedy 3 skupiny dvojic obrázků podle uživatelských hodnocení

- Podobné obrázky
- Částečně podobné obrázky
- Odlišné obrázky

Výsledné skóre se pak počítá v těchto krocích. Zjistíme aktuální počet dvojic obrázků ve skupině podobných obrázků podle uživatelů. Vezměme pak stejný počet dvojic s nejlepším hodnocením podle algoritmu. Sledujeme pak 2 hodnoty:

- Kolik procent dvojic je obsaženo zároveň v obou množinách nejpodobnějších obrázků (první podle uživatelů, druhá podle algoritmu).
- Kolik procent dvojic obrázků z nejpodobnějších dvojic podle algoritmu se vyskytuje naopak v množině odlišných dvojic obrázků podle hodnocení uživatelů.

První číslo by mělo být co nejvyšší, protože představuje soulad mezi uživatelským a algoritmickým hodnocením. Naopak druhé číslo by mělo být co nejmenší, protože určuje míru chybovosti – uživatelé označili obrázky za velice odlišné, algoritmus je ale naopak určil jako podobné.

Zbytek do 100% pak představuje procento mírných chyb – tedy, že algoritmus detekoval dvojici jako podobnou, ale uživatelé pouze jako částečně podobnou.

Příklad výpočtu definovaných ukazatelů:

ID 1.obr	ID 2.obr	Hodnocení
1	2	75
3	4	66
1	3	30
1	4	25
2	4	8
2	3	5

Tabulka 5.3: Nejpodobnější dvojice obrázků podle uživatelů

ID 1.obr	ID 2.obr
3	4
2	4
1	3
1	2
2	3
1	4

Tabulka 5.4: Nejpodobnější dvojice obrázků podle algoritmu

Pak jsou skupiny podle uživatelského hodnocení vytvořeny takto:

- Podobné obrázky – $\{(1,2),(3,4)\}$
- Částečně podobné obrázky – $\{(1,3),(1,4)\}$
- Odlišné obrázky – $\{(2,4),(2,3)\}$

Podobné dvojice podle uživatelů jsou 2, proto vezmeme rovněž 2 nejpodobnější dvojice podle algoritmu, to jsou $\{(3,4),(2,4)\}$. Jelikož pouze první dvojice je podobná i podle uživatelů, tak první hodnota ukazatele je 50%. A protože druhá dvojice která byla algoritmem určená jako jedna z nejpodobnějších je ale podle uživatelů odlišná, tak hodnota druhého ukazatele je rovněž rovna 50%.

Tento příklad byl uveden pouze na velice malé množině, v kolekci obrázků zvolené pro účely této práce je podobných dvojic zhruba 100.

Stejně jako u předchozího ukazatele i náhodné přiřazení podobnosti získá nenulové hodnocení. Opět na základě několika náhodných pokusů bylo zjištěno, že pro data (stav uživatelských hodnocení a nastavení hranic jednotlivých skupin) aktuální při psaní práce, skóre ukazatele vycházelo průměrně 7% pro první a 63% pro druhou hodnotu.

6. Vyhodnocení experimentu

V této poslední kapitole bude diskutován vliv nastavení jednotlivých parametrů na proces mapování deskriptorů, budou rozebrány možné metriky počítání výsledné podobnosti a porovnány jejich výsledky s hodnoceními získanými od uživatelů.

6.1 Parametry ovlivňující mapování deskriptorů

Jak bylo již popsáno, proces mapování deskriptorů je závislý na nastavení některých parametrů a na konkrétní implementaci algoritmu mapování. V této sekci bych rád diskutoval vliv těchto faktorů:

- Nastavení maximálního poměru mezi nejbližším a druhým nejbližším deskriptorem (popsáno jako parametr c v sekci 4.5.1)
- Mapování při použití sjednocení deskriptorů z obou směrů mapování (režim OR) nebo použití pouze deskriptorů namapovaných oběma směry (režim AND)
- Filtrování nebo ponechání deskriptorů s odlišnou orientací a velikostí

Sledovaným ukazatelem zde bude počet namapovaných deskriptorů a abychom mohli naleznout nejlepší parametry, tak bude provedeno porovnání tohoto ukazatele s uživatelskými hodnoceními pomocí technik popsaných v kapitole 5.2.

Při výpočtu byly detekovány počty namapovaných deskriptorů zobrazených v tabulce 6.1.

Režim	Hodnota parametru c				
	0,5	0,6	0,7	0,8	0,9
OR+filtr	0/1/79	0/2/159	2/13/262	14/60/467	76/385/1118
OR	0/1/79	0/5/166	7/33/322	57/228/643	320/1326/2989
AND+filtr	0/0/24	0/1/48	0/1/69	0/3/147	0/33/243
AND	0/0/24	0/1/50	0/3/94	0/22/185	32/211/536

Tabulka 6.1: Počty namapovaných deskriptorů (zaokrouhleno na jednotky, hodnoty popořadě představují minimální, průměrný a maximální počet namapovaných deskriptorů) v závislosti na hodnotě parametru c a zvoleném režimu mapování.

Jak se dalo očekávat, počty namapovaných deskriptorů v režimu OR mnohonásobně převažují počty deskriptorů namapovaných v režimu AND.

Pro tyto výsledky pak byly spočítány hodnoty ukazatelů, ty jsou zobrazeny v tabulce 6.2

6.1.1 Poměr mezi vzdálenostmi deskriptorů

Jak je vidět z obou tabulek, hodnota $0,5$ je pro potřeby vyhledávání podobností moc malá, ve všech režimech mapování mimo režimu OR bez filtrování jsou výsledky velice slabé, což je dáno tím, že pro většinu dvojic není namapován žádný

Režim	Hodnota parametru c				
	0,5	0,6	0,7	0,8	0,9
OR+filtr	12/17/61	26/21/41	29/19/51	20/10/68	18/7/76
OR	30/20/45	31/29/29	26/18/46	26/12/67	21/11/70
AND+filtr	11/14/56	27/21/50	28/25/46	36/26/37	31/17/60
AND	13/17/52	31/29/45	40/27/32	42/29/33	33/17/67

Tabulka 6.2: Skóre hodnocení (první hodnota je skóre založené na porovnávání pěti nejpodobnějších obrázků definované v sekci 5.2.1 a další dvě hodnoty jsou pak obě skóre metriky založené na správnosti segmentace popsané v sekci 5.2.2) shodnosti výsledků výpočtu s uživatelskými hodnoceními v závislosti na hodnotě parametru c a zvoleném režimu mapování.

deskriptor. Ale výsledky z mapování OR bez filtrování jsou v prvním ukazateli poměrně vysoké. Když se podíváme na vypočtené hodnoty, je vidět, že je zde několik dvojic, které mají několikanásobně vyšší počet namapovaných deskriptorů než je průměrná hodnota a opravdu jde o ty nejpodobnější dvojice, které se v databázi nacházejí. Mezi nalezené nejpodobnější dvojice (počet deskriptorů 7 a více) není tedy přimícháno velké množství nepodobných dvojic. Pokud je tedy potřeba vyhledávání pouze několika těch nejpodobnějších dvojic, může to být správný ukazatel.

Pro hodnotu $0,9$ je naopak množství nalezených deskriptorů již moc veliké, většina namapovaných deskriptorů je nesprávných a správné dvojice, které při menších hodnotách výrazně převyšovaly ostatní nepodobné dvojice jsou nyní ztraceny mezi nepodobnými dvojicemi, které mají stejně nebo i více deskriptorů – avšak nesprávně namapovaných. Rovněž distribuční funkce výsledné veličiny je téměř lineární na celém svém definičním oboru.

Výsledky pro hodnoty $0,6 - 0,8$ pak záleží na režimu mapování. V režimu OR pro hodnoty $0,7$ a $0,8$ je množství deskriptorů již poměrně veliké a to se projevuje tím, že se sice nejpodobnější dvojice, které mají dobrou detekci i při nižších hodnotách drží stále v popředí hodnocení, ale těsně za nimi se začínají objevovat odlišné dvojice, což je i dobře vidět v tabulce výsledků na třetím uvedeném ukazateli, který pro hodnoty vyšší než $0,6$ začíná v režimu OR dramaticky narůstat.

Naopak v režimu AND jsou nejlepší výsledky dosaženy pro hodnotu $0,8$. Rovněž je možné vypozorovat, že výsledky u nichž byl u nějaké dvojice detekován počet deskriptorů více než 200 začínají ztrácet na kvalitě, stejně tak pokud je minimální počet namapovaných deskriptorů větší než 0.

6.1.2 Filtrování podle orientace a velikosti

V tomto případě naměřené výsledky hovoří velice jasně, pokud porovnáme odpovídající režimy mapování s filtrovanými a nefiltrovanými deskriptory podle orientace a velikosti, tak pro odpovídající hodnoty c jsou výsledky získané mapováním bez filtrování až na jeden případ (režim OR, $c = 0,7$) výrazně lepší. Pro měření podobnosti tedy není vhodné při mapování filtrovat deskriptory podle velikosti a orientace.

6.1.3 Režim mapování

Své nejlepší výsledky dosahuje každý režim při jiné hodnotě parametru c. Zatímco pro režim OR je pro c optimální hodnota přibližně $0,6$, tak mapování v režimu AND dosahuje nejlepších výsledků pro hodnoty $0,7 - 0,8$. Výsledky všech třech (AND s $c = 0,7$ a $c = 0,8$, OR s $c = 0,6$) metrik se vyznačují tím, že je zde několik málo dvojic, které mají výrazně vyšší počet namapovaných deskriptorů než zbytek. A ve většině případů jde opravdu o velice dobře nalezené podobnosti, kde nejsou přimíchávány odlišné dvojice. Podle ukazatelů se ale jeví jako nejlepší režim AND s hodnotou $c = 0,8$ nebo $c = 0,7$, který nejlépe odhalí výrazné podobnosti, přičemž procento nesprávně odhalených podobností zůstává akceptovatelné. Kvalita této metriky je vidět i bližším pohledem na výsledky (např. v aplikaci SimBIm ve výsledcích algoritmů), které se opravdu v porovnání s ostatními zdají jako nejlepší.

6.2 Vliv rozměrů obrázku na výsledky

Hodnotící aplikace SimCom umožňuje spustit výpočet nad obrázky ve větší velikosti (omezeno šířkou na 1000px, průměrně zhruba 0,6 megapixelů) a nebo nad obrázky zmenšenými (omezeno šířkou 425px, zhruba 0,12 megapixelů). V následujcích odstavcích bude ukázáno, jaký je vliv velikosti obrázku na počtu nalezených a namapovaných deskriptorů a jak si vedou tyto výsledky v hodnocení.

6.2.1 Počty nalezených a namapovaných deskriptorů

V případě spuštění výpočtu nad velkými obrázky je počet nalezených deskriptorů průměrně kolem 3000, ale tyto hodnoty mají veliký rozptyl (hodnoty od cca 350 až např. do 7500), kdy záleží na obsahu obrázku, kolik deskriptorů je nalezeno. Nejméně deskriptorů je, jak lze očekávat, nalezeno v obrázcích, kde převažují jednobarevné velké plochy nebo nečlenité objekty (např. nebe, pole, vodní hladina), nejvíce jich je nalezeno naopak ve členitých obrázcích (např. domy, stromy s listy).

Při použití menších obrázků je pak nalezených deskriptorů průměrně kolem 600, přičemž hodnoty jsou z intervalu od 110 do 1800. Odpovídá tedy, že počet nalezených deskriptorů je přímo úměrný velikosti obrázku, kdy počty nalezených deskriptorů jsou u větších obrázků zhruba 5 krát větší než u menších obrázků, což i odpovídá poměru velikosti těchto obrázků.

Podíváme–li se na počty namapovaných deskriptorů pro jednotlivé velikosti obrázků, pak lze vidět, že průměrné množství namapovaných deskriptorů pro menší obrázky není již přibližně 5 krát menší, jak tomu bylo u počtu nalezených deskriptorů, ale pouze 3–4 krát menší. Přesné hodnoty počtu namapovaných deskriptorů a výsledné hodnocení je pak vidět v tabulce 6.3.

Jak je jasně vidět, všechny metriky výrazně ztratily na kvalitě, množství namapovaných deskriptorů je příliš malé a jejich porovnávání nevede k dobrým výsledkům. Ani při zvětšení hodnoty c tak, aby byl počet namapovaných deskriptorů u malých obrázků podobný jako u velkých, nevede k dobrým výsledkům, protože toto umělé zvýšení přináší velké množství nesprávně namapovaných deskriptorů.

Proto dále v této práci budou uvažovány pouze výsledky spočítané nad obrázky větší velikosti.

Režim	Hodnota c	Obrázky	Počet deskriptorů	Skóre metriky
OR	0,6	velké	0/4,66/166	31/29/29
		malé	0/1,68/58	27/15/49
AND	0,7	velké	0/2,61/94	40/27/32
		malé	0/0,75/28	21/12/47
	0,8	velké	0/21,87/185	42/29/33
		malé	0/6,42/58	26/18/47

Tabulka 6.3: Porovnání počtu namapovaných deskriptorů (minimálního, průměrného a maximálního) a shodnost hodnocení s uživateli při detekci z velkých a zmenšených obrázků

6.3 Kombinace hodnot

Zatím byla jako jediná metrika uvažován počet namapovaných deskriptorů. K dispozici ale máme další údaje z procesu detekování a mapování deskriptorů, které by bylo možné k výpočtu podobnosti využít. Jsou to např. tyto hodnoty:

- Vzdálenosti mezi jednotlivými deskriptory – průměrná, minimální, maximální...
- Poměry vzdáleností mezi nejbližším a druhým nejbližším deskriptorem
- Počty detekovaných a namapovaných deskriptorů

Prvním vylepšením by mohlo být uvažování počtu nalezených deskriptorů. Jak bylo ukázáno, počty nalezených deskriptorů se pro jednotlivé obrázky velice liší, proto pravděpodobně dvojice obrázků, kde mají oba obrázky velké množství deskriptorů jsou, pokud uvažujeme jako metriku pouze počet namapovaných deskriptorů, zvýhodněny nad dvojicemi, které mají menší počet deskriptorů. Proto by mohlo být vhodné výsledný počet namapovaných deskriptorů nějak vážit počtem nalezených deskriptorů.

Avšak po vypočítání hodnot je vidět, že výsledky rozhodně nejsou dobré, podobné dvojice, které byly podle počtu deskriptorů jasně detekovány jsou nyní mnohem hlouběji ve výsledcích a do popředí se dostalo několik obrázků, které nejsou podobné, ale těží hlavně z toho, že nemají mnoho deskriptorů. Metrika procento namapovaných deskriptorů dosahuje výsledků 31/19/43, což je mnohem horší skóre než u metriky počet namapovaných deskriptorů (42/29/33).

Další nabízející se možností bylo využití informací o vzdálenostech mezi namapovanými deskriptory a poměrech mezi nejbližším a druhým nejbližším deskriptorem.

Jako první jsem se pokusil vyzkoušet pouze tyto údaje, aniž bych použil počet namapovaných deskriptorů, když jsem se snažil použít podobnou metriku jako je použita v [7]. Použití této metriky, ani různých kombinací hodnot nevedlo k výsledkům, které by se alespoň trochu přiblížily kvalitě výsledků, kdy je použit počet namapovaných deskriptorů jako jediná metrika.

Proto jsem se snažil ještě zkombinovat počet namapovaných deskriptorů s těmito hodnotami. Provedl jsem velké množství testování různých kombinací využití parametrů, ve většině případů se výsledné skóre metriky pohybovalo kolem skóre,

kdy byl uvažován pouze počet deskriptorů. Nejlepší výsledky jsem ale dosáhl pro metriky, jejichž výsledné hodnoty byly počítány následovně:

$$s = MD(I_1, I_2) * (1 - \sqrt{D_{avg}}) * (1 - \sqrt{R_{avg}}) * (1 - \sqrt{R_{min}})$$

a

$$s = MD(I_1, I_2) * (1 - D_{avg}) * (1 - {R_{avg}}^2) * (1 - R_{min})$$

kde $MD(I_1, I_2)$ je počet namapovaných deskriptorů z mapování v režimu AND a $c = 0,8$, D_{avg} je průměrná vzdálenost mezi namapovanými dvojicemi vektorů, R_{avg} (respektive R_{min}) je průměrný (respektive minimální) poměr vzdáleností mezi nejbližším a druhým nejbližším deskriptorem a s pak je výsledná hodnota podobnosti mezi obrázky I_1 a I_2. Vyšší hodnota s pak znamená vyšší podobnost.

První uvedená metrika dosáhla skóre 44/30/29 (dokonce 49/30/29 pokud nejsou uvažována hodnocení obrázků ze skupiny Auta) a druhá pak 43/30/27 (respektive 48/30/27), což znamená mírné vylepšení hodnot než když byl uvažován pouze počet namapovaných deskriptorů. Největší zlepšení je v posledním ukazateli, který představuje procento špatně označených dvojic.

6.4 Zhodnocení výsledků

Nejlepší nalezené metriky a jejich výsledky byly popsané v předchozích sekcích, nelze však vyloučit, že existují a v budoucnu se naleznou buď takové kombinace parametrů nebo úplně nové techniky výpočtů, jejichž výsledky budou více odpovídat uživatelskému vnímání podobnosti. Tento výzkum může být předmětem další práce.

6.4.1 Relativnost výsledků

Prezentované metriky výpočtu podobnosti neposkytují univerzální výsledek, který by hodnotil číselně absolutní podobnost dvou obrázků, ale spíše jsou určeny pro relativní srovnávání v rámci kolekce. Jelikož jsou hodnoty výsledné metriky závislé na parametrech kolekce (jako je charakter obrázků, jejich rozlišení a další parametry) ani není možné poskytnout nějakou univerzální hodnotu, která bude určovat hranici od které je možné obrázky považovat za podobné. Tuto hodnotu je potřeba zvolit na základě výsledků v konkrétní kolekci, buď ručně a nebo metodou, která bude zohledňovat požadované účely vyhledávání a charakter kolekce.

6.4.2 Korektnost výsledků

Na začátku realizace práce byly očekávány lepší výsledky pro hodnocení podobnosti touto metodou, jak se ale ukázalo během práce, výpočet podobností metodou SURF má své limity. Metoda funguje především při odhalování podobnosti obrázků, které obsahují stejný nebo podobný objekt. Např. většina obrázků z testovací kolekce, na kterých byl zobrazen Pražský hrad měla velice správně velmi vysoké skóre, naopak odhalování podobnosti mezi obecnými obrázky obsahujícími objekty, které nepředstavují stejné ale pouze podobné objekty není vždy odhalena.

Závěr

V této práci byla provedena analýza a implementace aplikace pro sběr uživatelských hodnocení podobnosti obrázků a pomocí této aplikace byla nasbírána data, která byla později použita pro porovnávání výsledků uživatelského vnímání podobnosti s výsledky spočítanými v práci prezentovanými metrikami založenými na použití algoritmu SURF. Využití nasbíraných dat však není vázáno pouze pro potřeby této práce, je možné prostřednictvím aplikace SimBIm provádět ověřování kvality výpočtů podobností mezi obrázky založených na libovolných algoritmech. Aplikace SimBIm tedy může do budoucna sloužit jako benchmark pro všechny vyvíjené algoritmy.

Dále bylo diskutováno použití algoritmu SURF pro účely podobnostního vyhledávání v kolekci obrázků a byly učiněny pokusy o nalezení takových způsobů hodnocení a nastavení parametrů, jejichž výsledky budou co nejvíce odpovídat uživatelskému vnímání podobnosti, tak jak byla zachycena při sběru dat na začátku realizace této práce.

Na základě prvních výsledků se jevilo, že použití algoritmu SURF není pro vyhledávání podobností vůbec vhodné, ale v průběhu práce se podařilo naleznout takové techniky a nastavení parametrů, jejichž výsledky jsou již použitelnější, avšak stále ne ideální, např. z důvodu, že detekované podobnosti obsahují kolem 30% falešných výskytů (mezi podobné obrázky se dostanou i dvojice, které vůbec podobné nejsou). Použití na některých specifických kolekcích může ale fungovat velice dobře.

Možná vylepšení

Do budoucna je možná celá řada vylepšení a to téměř ve všech dílčích částech této práce.

V aplikaci SimBIm je potenciální velké množství dat, které je možné vydolovat z databáze a část z nich je pro uživatele zatím nedostupná. Bylo by zajímavé vidět podrobnější vyhodnocení algoritmů, např. podle skupin obrázků nebo možnost zjišťovat, v čem jsou některé zvolené metriky lepší než ostatní. Tyto údaje by mohly védst k snažším odhalování slabých míst algoritmů.

Aplikace SimCom může být vylepšena hned v několika bodech, jedním je např. implementace výpočtu, kde je možné využít běhu výpočtu na GPU (použitá knihovna EmguCV to podporuje, ale zatím to není v aplikaci implementováno). Využití paralelizace výpočtu aby mohl běžet na více jádrech najednou by přineslo na stroji s více jádry znatelné zrychlení výpočtu. Rovněž algoritmus pro mapování dskriptorů může být vylepšen o vhodné indexování které přinese výrazné zrychlení. Případně by bylo vhodné i upravit použití aplikace tak, aby nebylo nutné ruční stahování hodnotícího balíčku a zpětné nahrávání výsledků na server, ale aby mezi aplikacemi SimBIm a SimCom probíhala komunikace automaticky na principu klient/server.

Velký prostor je jistě i ve zlepšování detekování podobnosti, kde se nabízí např. kombinace prezentovaného výpočtu s algoritmy fungujícími na jiném principu — např. algoritmy založené na barevné analýze obrazu. Zlepšení přesnosti by pak mohla přinést hlubší analýza namapovaných deskriptorů, založená hlav-

ně na jejich vzájemných pozicích. Nyní je např. na obou obrázcích nalezen jeden podobný objekt, který je ale naprosto nevýznamný pro celkovou podobnost, ale velké množství deskriptorů (které jsou ale soustředěny jen v malé oblasti) indikuje poměrně vysokou podobnost. Např. dvojice s rovnoměrně namapovanými deskriptory na celé ploše obrázků by mohly být upřednostňovány. Tato vylepšení by pak mohla védst k přesnějším a kvalitnějším výsledkům.

Seznam použité literatury

[1] D.G Lowe. Distinctive image features from scale-invariant key points. *International Journal of Computer Vision*, 60(2):91–110, 2004

[2] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool Speeded-Up Robust Features (SURF) *ECCV 2006 In Computer Vision*, 3951:404–417, 2006

[3] C. Silpa-Anan, R. Hartley Optimised KD-trees for fast image descriptor matching *IEEE Conference on Computer Vision and Pattern Recognition*, 2008

[4] L. Juan, O. Gwon A Comparison of SIFT, PCA-SIFT and SURF *International Journal of Image Processing*, 4:143–152, 2009

[5] K. Mikolajczyk, C. Schmid A performance evaluation of local descriptors *Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005

[6] P. Bílek Významné body v obraze: detekce, lokalizace a korespondence ve 3D *České vysoké učení technické, Fakulta elektrotechnická*. Bakalářská práce, Praha 2007. Dostupné z: `https://dip.felk.cvut.cz/browse/pdfcache/bilekp3_2007bach.pdf`

[7] G. Du, F. Su, A. Cai Face recognition using SURF features *MIPPR 2009: Pattern Recognition and Computer Vision*, 2009. Dostupné z: `http://robotics.csie.ncku.edu.tw/HCI_Project_2009/Face_recognition_using_SURF_features.pdf`

[8] EmguCV – grafická knihovna, domovská stránka: `http://www.emgu.com`

[9] Dragsort – jQuery plugin, domovská stránka: `http://dragsort.codeplex.com`

[10] jQuery – JavaScriptová knihovna, domovská stránka: `http://jquery.com`

[11] Flickr.Net API Library – knihovna pro komunikaci se serverem `http://flickr.com`, domovská stránka: `http://flickrnet.codeplex.com`

[12] Flickr API – rozhraní pro komunikaci se serverem Flickr.com, domovská stránka: `http://www.flickr.com/services/api/`

[13] Altairis Web Security – role a membership provider, domovská stránka: `http://altairiswebsecurity.codeplex.com`

[14] C. Nagel a kolektiv C# 2008: Programujeme profesionálně *Brno: Computer Press, 2009*. ISBN 978-80-251-2401-7

[15] Microsoft Developer Network. Dostupné z `http://msdn.microsoft.com/`

[16] Books Online for SQL Server 2012. Dostupné z `http://msdn.microsoft.com/en-us/library/ms130214.aspx`

Seznam použitých zkratek

CBIR Content Based Image Retrieval

SIFT Scale Invariant Feature Transform

SURF Speeded Up Robust Features

SimBIm Similarity Between Images

SimCom Similarity Computer

2NF Druhá normální forma

3NF Třetí normální forma

WPF Windows Presentation Foundation

GLOH Gradient Location and Orientation Histogram

LoG Laplacian of Gaussian

Příloha A - Obsah přiloženého disku

Zdrojové kódy

Zdrojové kódy aplikace SimBIm i SimCom jsou přiložené jako Visual C# projekty v těchto adresářích:

- SimBIm v adresáři `/SimBIm/Zdrojovy_projekt/`
- SimCom v adresáři `/SimCom/Zdrojovy_projekt/`

Instalační soubory

Při instalaci aplikace SimBIm je doporučeno postupovat podle návodu, který je v souboru `/SimBIm/Dokumentace/SimBIm_Navod_k_instalaci.pdf`, data potřebná k instalaci aplikace jsou uložená v adresáři `/SimBIm/Instalace/`.

Aplikaci SimCom lze nainstalovat pomocí přiloženého ClickOnce instalátoru, instalace proběhne spuštěním souboru `/SimCom/Instalator/setup.exe`

Analýza

Analýza k aplikaci SimBIm je uložena v adresáři `/SimBIm/Analyza/`,

- soubor `Analyza.pdf` obsahuje textovou část analýzy
- soubor `Pripady_uziti.pdf` obsahuje popisy jednotlivých případů užití
- v adresáři `Diagramy` lze pak naleznout vyexportované diagramy (případy užití, ER model a další)

Dokumentace

K aplikaci SimBIm jsou v adresáři `/SimBIm/Dokumentace/` přiložené následující dokumentace:

- `SimBIm_Administratorska_dokumentace.pdf` – uživatelská dokumentace administrátorské části aplikace
- `SimBIm_Navod_k_instalaci.pdf` – návod k instalaci aplikace
- `SimBIm_Programatorska_dokumentace.pdf` – textová část programátorské dokumentace
- `SimBIm_Programatorska_dokumentace_vygenerovana.zip` – archiv obsahující vygenerovanou dokumentaci ze zdrojových kódů, která rozšiřuje textovou část programátorské dokumentace

- `SimBIm_Uzivatelska_dokumentace.pdf` – uživatelská dokumentace uživatelské (část pro hodnocení) části aplikace

K aplikaci SimCom jsou v adresáři `/SimCom/Dokumentace/` přiložené následující dokumentace:

- `SimCom_Programatorska_dokumentace.pdf` – programátorská dokumentace
- `SimCom_Uzivatelska_dokumentace.pdf` – uživatelská dokumentace

Výsledky experimentů

Výsledky experimentů je možné prohlížet pohodlně na webu z aplikace SimBIm, pokud je nainstalována ve verzi s daty a jsou naimportována data s výsledky (popsáno v návodu k instalaci).