VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ FOTOGRAFIÍ PODLE OBSAHU

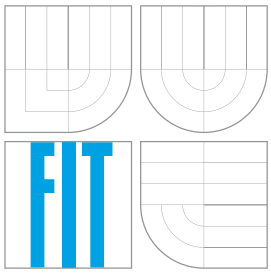
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL DVOŘÁK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ FOTOGRAFIÍ PODLE OBSAHU

CONTENT BASED PHOTO SEARCH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL DVOŘÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2014

Abstrakt

Tato závěrečná práce se zabývá návrhem aplikace pro rychlé vyhledávání podobných fotografií ve velké databázi desítek až stovek tisíc fotografií. Součástí je návrh způsobu extrakce příznaků a vytvoření slovníku vizuálních slov. S tím je dále spojena problematika reprezentace těchto informací v databázi a způsob jejich rychlého vyhledávání. Na závěr jsou ve zprávě popsány experimenty s implementovanou aplikací, testy rychlosti vyhledávání a rozbor škálovatelnosti řešení.

Abstract

This thesis covers design and practical realization of a tool for quick search in large image databases, containing from tens to hundreds of thousands photos, based on image similarity. The proposed technique uses various methods of descriptor extraction, creation of Bag of Words dictionaries and methods of storing image data in PostgreSQL database. Further, experiments with the implemented software were carried out to evaluate the search time effectivity and scaling possibilities of the design solution.

Klíčová slova

Vyhledávání fotografií, keypoint, deskriptor, Bag of Words, databáze, PostgreSQL, invertovaný index

Keywords

Photo search, keypoint, descriptor, Bag of Words, database, PostgreSQL, inverted index

Citace

Pavel Dvořák: Vyhledávání fotografií podle obsahu, diplomová práce, Brno, FIT VUT v Brně, 2014

Vyhledávání fotografií podle obsahu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Michala Španěla

.....
Pavel Dvořák
27. května 2014

Poděkování

Děkuji panu doktoru Španělovi za odbornou pomoc s touto prací a všem blízkým, kteří mě během psaní podporovali.

Obsah

Kapitola 1

Úvod

Cílem této práce je návrh aplikace schopné rychle (v řádech vteřin) vyhledávat množinu fotografií, které se dostatečně podobají uživatelem specifikované fotografii.

Práce je rozdělena do sedmi hlavních kapitol. V prvních kapitolách rozeberu teoretické základy k vyhledávání fotografií, budu se tedy věnovat různým způsobů vyhledávání v obrazových datech a několika způsobům vyhledávání těchto dat v databázi fotografií, spolu s jejich výhodami a nedostatky.

Dále se věnuji různým algoritmům pro detekci a popis význačných bodů ve fotografii, protože se práce věnuje primárně této metodě vyhledávání. V kapitole je popsáno množství nejpoužívanějších algoritmů spolu s ilustrací jejich funkčnosti a jejich vzájemnému porovnání. Dále je v kapitole popsána praktická aplikace těchto algoritmů na získání Bag of Words slovníku a deskriptoru, který nám bude charakterizovat jednotlivé fotografie.

V následující kapitole se přesuneme k databázím a metodám ukládání a efektivního indexování našich fotografií. Je zde stručně popsán zvolený databázový systém a podporované metody indexování.

Následují dvě vzájemně provázané kapitoly, které se věnují návrhu samotné aplikace spolu se zvolenými algoritmy a důvodům pro jejich volbu, abychom následně přešli k samotné implementaci a řešeným problémům.

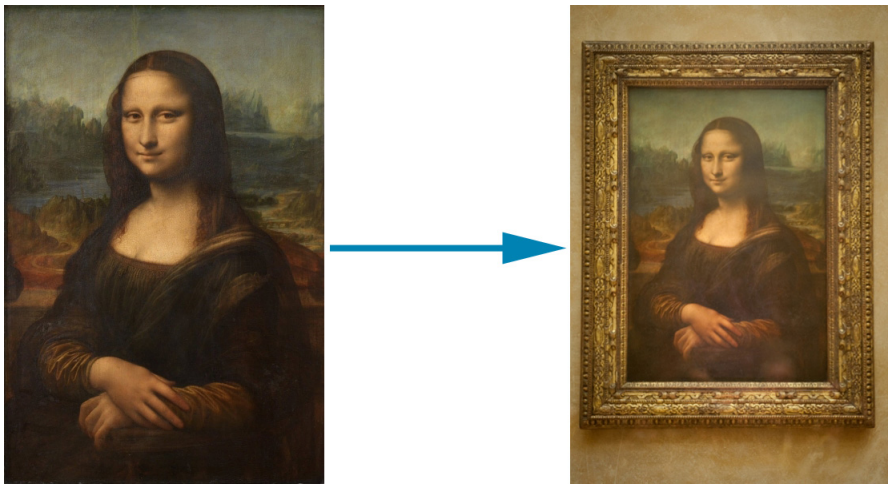
Za závěr je v práci uvedeno několik experimentů s vytvořenou aplikací, které se věnují složitosti algoritmů a přesnosti vyhledávání.

Kapitola 2

Vyhledávání v obrazových datech

Tato kapitola pojednává o některých možných formách vstupních dat, která uživatel chce vyhledávat. Při implementaci aplikace na vyhledávání fotografií podle obsahu je dobré si nejprve ujasnit, jaká data bude vlastně uživatel do procesu vkládat (a jakým způsobem to bude provádět, případně jaký výstup očekává). Od toho se odvíjí zvolení jedné nebo více forem vstupních dat, které jsou popsány dále.

2.1 Celková fotografie



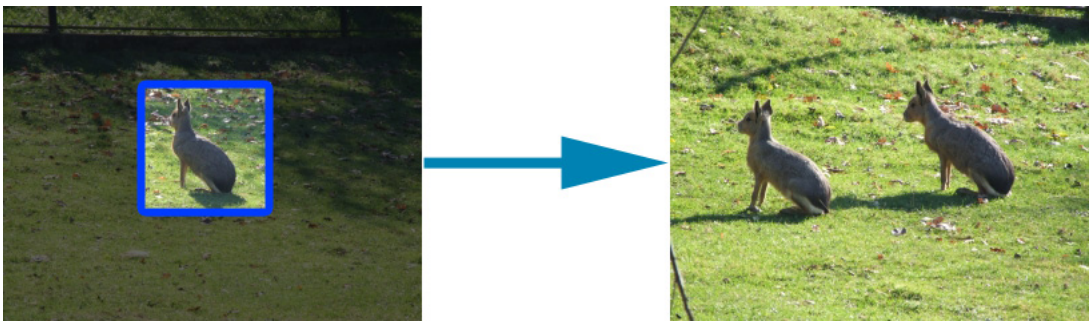
Obrázek 2.1: Vyhledávání celého obrázku.

První možností je vyhledávání celkově podobných fotografií. Může jít zároveň o nejjednodušší a zároveň o nejsložitější variantu implementace v závislosti na tom, co od výsledku očekáváme. Nijak neupravované fotografie mohou mít velké množství nadbytečných informací (například nás nezajímá pozadí fotografie, ale jen nějaký význačný objekt), které zkreslují výsledné vyhledávání. Na druhou stranu podporují například velice jednoduchá porovnání na základě barevných histogramů. Možné varianty tohoto způsobu:

- Vyhledávání na základě vlastností fotografie (rozměry, název) – Např. TinEye.
- Vyhledávání na základě podobnosti barev (histogramy, barevné prostory, segmentace).

- Vyhledávání na základě význačných bodů, strojové učení (nutné zakomponování informace o prostorové závislosti bodů).

2.2 Výřez



Obrázek 2.2: Vyhledávání obrázků podle výřezu.

Vyhledávání výřezu umožňuje specifikování hledaného objektu a minimalizuje zkreslující informace o pozadí (ilustrováno na obrázku 2.2). Obvykle se používá tam, kde máme databázi segmentovaných fotografií, u kterých již známe polohu a typ objektů. Jedním z použití je například detekce a vyhledávání dopravních značek, kde nás zajímá jen specifická oblast (značka) a pozadí je zcela irelevantní.

Co se týká způsobu vyhledávání, tak je možné použít v podstatě libovolný přístup v závislosti na typu objektu, který se bude nejčastěji ve výřezu vyskytovat. Možné typy hledaných objektů zahrnují:

- Dobře rozlišitelné geometrické tvary – Využití segmentace, strojové učení.
- Běžné objekty, postavy – Detekce význačných bodů.
- Textury, barvy – Segmentace, barevné histogramy.

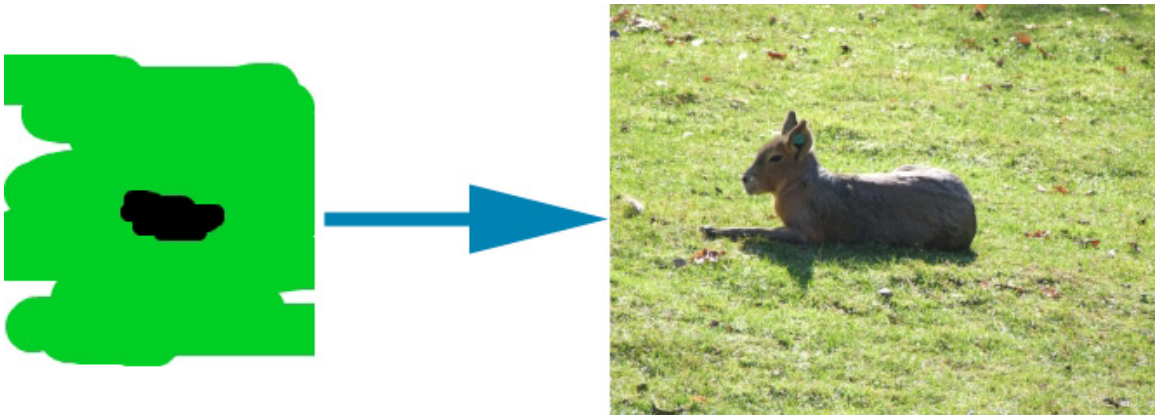
2.3 Nákres

Zajímavým a netradičním způsobem je vyhledávání fotografií na základě hrubého nákresu, který uživatel poskytne (viz obrázek 2.3). Takovéto aplikace zpravidla obsahují velice základní sadu kreslících nástrojů (paleta barev, různé velikosti štětce) a poskytují kreslící plochu, na kterou uživatel nakreslí zhruba to, co si přeje vyhledávat.

U této metody se spíše využívá segmentace obrazu a porovnávání barevnosti/histogramů, než metody strojového učení a detekce význačných bodů, protože na hrubém nákresu jsou zpravidla dobře viditelné segmenty, které uživatel ve fotografii vyžaduje a dle toho je možné vytvořit parametry vyhledávání. Oproti tomu hrubý nákres neobsahuje příliš mnoho kvalitních význačných bodů (resp. protože jde často o hranovou detekci, tak mohou být body zkresleny i jen nedokonalostí nákresu, viz bílé oblasti v obrázku 2.3).

Příkladem takového vyhledávače je aplikace *retrievr*[1].

[1] http://labs.systemone.at/retrievr/



Obrázek 2.3: Vyhledávání obrázku podle nákresu.

2.4 Klíčová slova

Vyhledávání na základě klíčových slov [21] předpokládá vstup v podobě textových řetězců reprezentujících významné vlastnosti fotografie (tzv. *tag*), pomocí kterých je poté z databáze vybrána množina fotografií, která těmto klíčovým slovům nejlépe odpovídá. Je vhodné mít ke každé fotografii co nejvíce specifickou množinu klíčových slov a zároveň vyhledávat využitím několika klíčových slov, aby se zamezilo omylům například ve formě homonym (např. klíčové slovo "kolo"je třeba dále specifikovat – "kolo horské", "kolo auto").

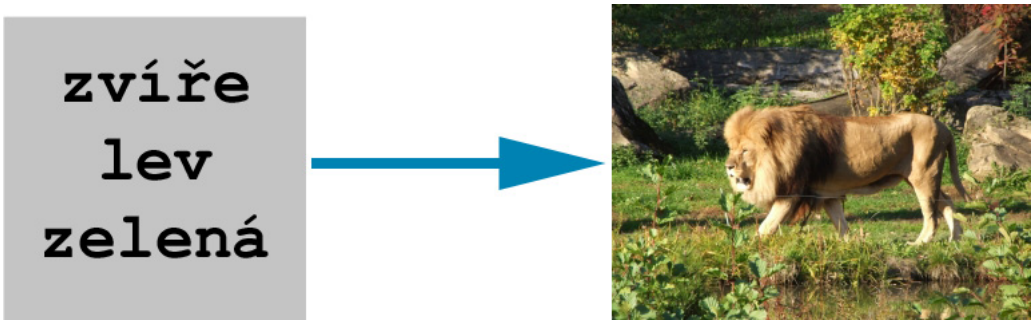
Pro výběr klíčových slov jsou dvě možnosti:

1. Klíčová slova jsou zadávána uživatelem.
2. Klíčová slova tvořena automaticky.

Přiřazení uživatelem

Tvorba a přiřazení klíčových slov uživatelem je nejjednodušší z hlediska implementace, protože je třeba řešit jen způsob uložení a následného vyhledávání klíčových slov. O zbytek (vložení fotografie, vybrání a zapsaní slov) se stará sám autor fotografie. Pokud ovšem nejde o člověka znalého daného databázového systému a principu vyhledávání, tak nevyhnutelně dojde k tvorbě šumu mezi klíčovými slovy. Uživatel může vkládat klíčová slova k nepodstatným detailům a zároveň opomenout podstatné části fotografie.

Dalším problémem je syntax klíčových slov. I drobná chyba způsobí, že fotografii nebude možné nalézt, navíc se mohou do systému zanést duplicity ve formě různého skloňování a jednotných/množných čísel.



Obrázek 2.4: Vyhledávání obrázků podle textového řetězce.

Přiřazení automaticky

Automatické přiřazování klíčových slov řeší předchozí problémy s rizikem mnohoznačnosti a duplicit, ale přináší problém v podobě správné detekce všech důležitých částí fotografie. Tento přístup je možné nalézt v aplikacích věnujících se strojovému učení a klasifikaci fotografií do kategorií, jejichž množina je zpravidla velmi omezená a jasně definovaná.

Pro klasifikaci fotografie do určité kategorie se používá metoda *Support vector machine* (SVM), což je úloha, při které je nad množinou příznaků optimálně rozdělena sada trénovacích dat (s využitím množiny negativních protipříkladů) na jednotlivé kategorie tak, aby bylo následně možné vstupní fotografie do některé kategorie přiřadit.

Kapitola 3

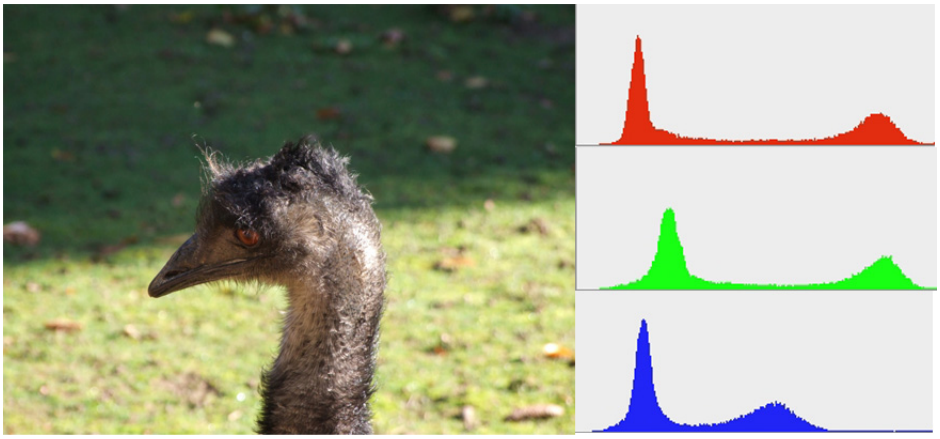
Metody vyhledávání v databázi fotografií

Tato kapitola pojednává o různých přístupech ke zpracování fotografie a získání informací potřebných k vyhledávání podobných vzorů nebo objektů v jiných fotografiích. Jde o záležitost specifickou pro způsob využití a typ hledaných dat, je třeba zvolit metodu, která nejlépe vyhovuje našim potřebám. Proto je vhodné v této kapitole rozebrat jednotlivé přístupy a možné aplikace těchto metod.

3.1 Porovnávání histogramů

Vyhledávání na základě porovnávání histogramů je relativně nejjednodušší a výpočetně nejrychlejší metodou pro určení podobnosti dvou fotografií. Tato metoda z každého obrázku získá vektor četností jednotlivých možných hodnot intenzity jasu, nebo jednotlivých barevných složek. Tento vektor poté slouží jako reprezentace daného obrázku pro další porovnávání.

Pro porovnávání podobnosti dvou histogramů existuje velké množství metod [6], uvedu tedy zde několik základních:



Obrázek 3.1: Histogramy barevných složek.

Běžný výpočet průniku histogramů

Jde o přímočarou metodu výpočtu míry podobnosti mezi dvěma histogramy. Uvažujme dva histogramy: histogram hledané fotografie H_M a histogram porovnávané fotografie H_T, které sestávají z n složek. Oba histogramy jsou normalizované, tedy $\sum_{i=1}^{n} h_M(i) = 1$ a $\sum_{i=1}^{n} h_T(i) = 1$. Průnik těchto histogramů je vypočítaný rovnicí

$$HI = \sum_{i=1}^{n} min(h_M(i), h_T(i)) \tag{3.1}$$

Výsledek spadá do intervalu $< 0, 1 >$ a vyjadřuje poměr počtu pixelů z hledané fotografie, které mají odpovídající obraz stejné barvy v porovnávané fotografii. Hodnota blížící se 1 označuje vysokou podobnost až barevnou ekvivalenci těchto dvou fotografií.

Gaussův vážený průnik histogramů

Předchozí metoda porovnávání je velmi jednoduchá, nicméně s sebou nese nevýhodu v podobě porovnávání absolutních hodnot počtu výskytů jednotlivých složek histogramu. I drobný šum v obraze nebo malá změna barvy rapidně změní hodnotu podobnosti těchto fotografií. Proto se metoda vylepšuje přidáním váhy pro jednotlivé složky tak, aby k míře podobnosti přispěly i dostatečně blízké hodnoty histogramu.

Funkce váženého průniku v mnohém vychází z běžného porovnávání a je definovaná následovně:

$$HI = \sum_{\vec{c_i} \in C_M} \sum_{\vec{c_j} \in C_T} min(h_M(\vec{c_i}), h_T(\vec{c_j})) \cdot exp\left(-\frac{||\vec{c_i} - \vec{c_j}||^2}{2\sigma^2}\right) \tag{3.2}$$
$$-3.3\sigma \leq ||\vec{c_i} - \vec{c_j}|| \leq 3.3\sigma$$

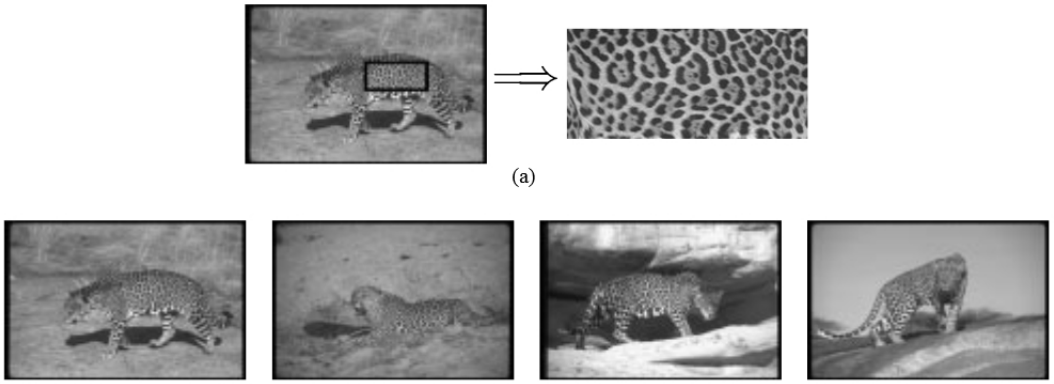
Vzdálenost dvou barev je definovaná jako

$$||\vec{c_i} - \vec{c_j}|| = \sqrt{(l_1 - l_2)^2 + (u_1 - u_2)^2 + (v_1 - v_2)^2} \tag{3.3}$$

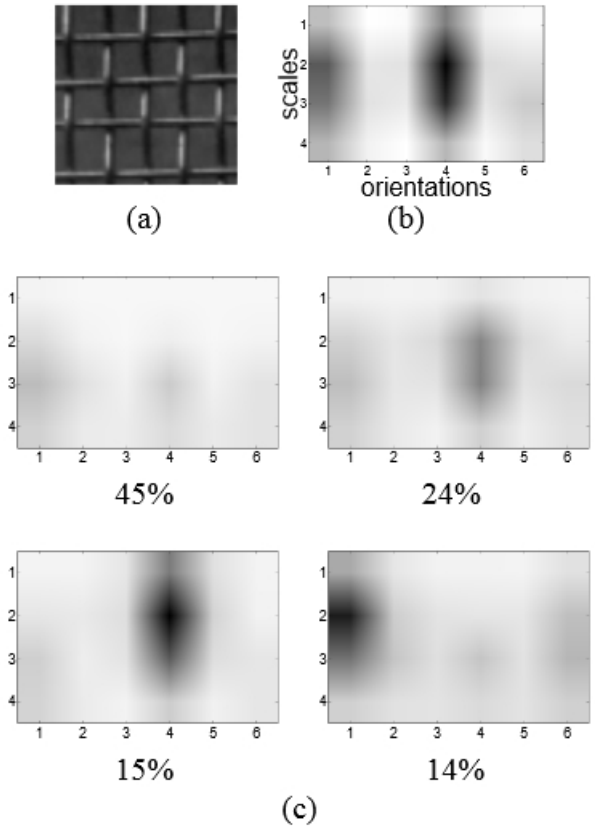
kde $\vec{c_1} = (l_1, u_1, v_1)$ a $\vec{c_2} = (l_2, u_2, v_2)$ jsou dvě barvy reprezentované v barevném modelu CIE LUV.

3.2 Vyhledávání podle textury

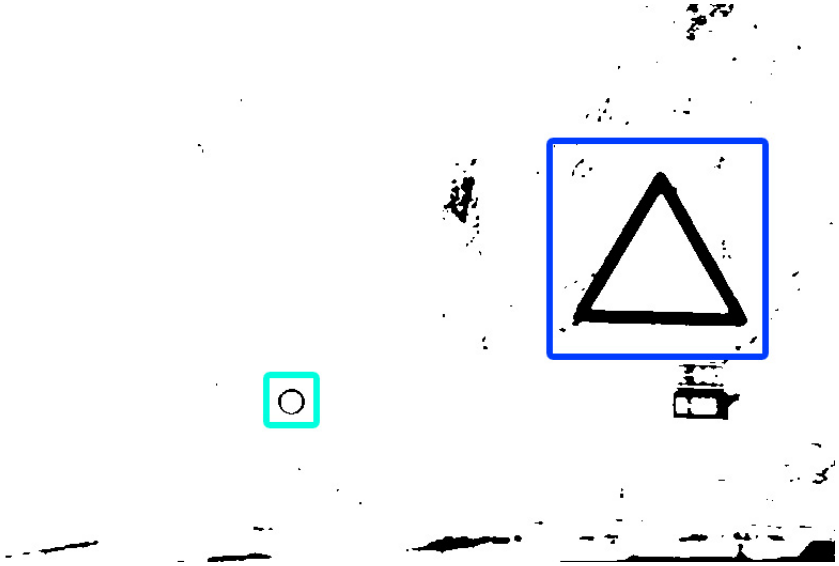
Vyhledávání na základě textury vyžaduje značnou míru porovnávání informace o rozmístění pixelů v prostoru, které nebylo u histogramu natolik potřeba. Textura se vyznačuje opakujícími se jasně ohraničenými vzory (nemusí se ovšem opakovat periodicky), takže do značné míry ulehčuje návrh segmentace obrazu, případně návrh reprezentace obrazu ve formě frekvence. V [18] je navržena detekce textur s využitím Gaborova filtru (lineární filtr používaný pro hranovou detekci a frekvenční analýzu), jejíž průběh je znázorněn na obrázku 3.3. V části (b) jsou vidět jednotlivé odezvy filtru (tmavší oblast značí silnější odezvu). Výstup samotného vyhledávacího algoritmu je vidět na obrázku 3.2.



Obrázek 3.2: Hledaná textura a nalezené výsledky. Obrázek převzat z [18].



Obrázek 3.3: (a) Analyzovaná textura. (b) Průměr všech příznaků textury. (c) Čtyři shluky příznaků spolu s jejich procentuální váhou. Obrázek převzat z [18].



Obrázek 3.4: Příklad segmentace obrazu pro detekci dopravních značek.

3.3 Vyhledávání tvarů

Termín "tvar"zde označuje výrazně ohraničený a dobře geometricky popsatelný objekt. Například může jít o vyhledání dopravní značky, detekci SPZ nebo i nalezení složitějších objektů jako třeba vozidlo.

Při vyhledávání tvarů se často používá segmentace obrazu, díky které z fotografie filtrujeme irelevantní informace a snažíme se, aby v obraze zůstaly jen pixely týkající se námi hledaného objektu (viz obrázek 3.4).

3.4 Detekce význačných bodů

Poslední zde zmíněnou metodou je detekce význačných bodů v obraze. Těmito body jsou zpravidla ostré hrany, které nesou dobrou informaci o obsahu dané fotografie. Tato práce je zaměřena především na tuto metodu, proto je jí věnována celá následující kapitola.

Kapitola 4

Detekce a zpracování význačných bodů

Prvním krokem při rozpoznávání a klasifikaci fotografie je redukce informace obsažené v této fotografii pouze na minimum bodů, které nejlépe charakterizují obsah dané fotografie. V této kapitole se budu věnovat metodám používaným pro získání množiny zájmových bodů a algoritmům pro popis regionů obrazu. Tyto dva prvky poté použijí při vytváření slovníku obrazových slov a následné klasifikaci vstupní fotografie v rámci určité databáze.

4.1 Keypoint

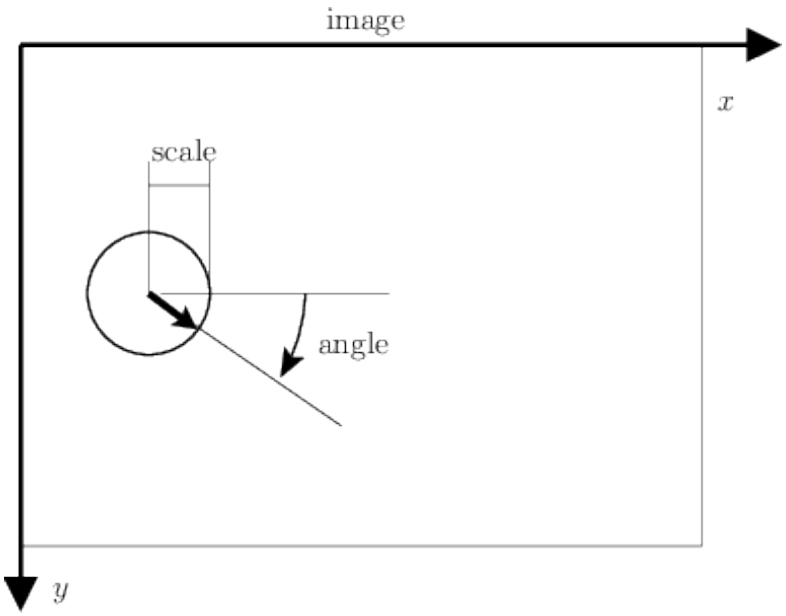
Význačný bod fotografie, neboli tzv. *keypoint*, označuje oblast, která nese největší množství informace o obsahu takové fotografie. Tyto oblasti jsou zpravidla lokální minima či maxima a jsou určeny některou metodou hranové detekce, která se liší podle daného algoritmu. Každý keypoint v závislosti na implementaci nese určitou informaci o svém okolí (detekce hrany nebo rohu), velikosti a směru. Keypoint je tedy nezávislý na změně pozice dané fotografie, fotografie posunutá o určitý počet pixelů bude mít stejné keypointy jako fotografie na původní pozici. Pro srovnání keypointů v oblastech fotografie se proto dále používají deskriptory.

4.2 Deskriptor

Keypointy samy o sobě nesou jen omezenou informaci o fotografii a pro naše potřeby nedostačují. Potřebujeme získat informaci o podobě většího segmentu obrazu. K tomu slouží deskriptory, které pracují jako suma významných vlastností určitého regionu fotografie. Může jít například o histogram intenzit pixelů nebo odezev některého filtru. Metoda popisu se liší podle implementace.

4.3 Existující metody

V této sekci jsou popsány běžně používané i zcela nové metody pro detekci keypointů a tvorbu jejich deskriptoru.



Obrázek 4.1: SIFT keypoint, obrázek převzat z [20].

4.3.1 SIFT

SIFT[9] je zkratkou pro Scale Invariant Feature Transforms. Algoritmus publikoval David Lowe v roce 1999.

Detekce

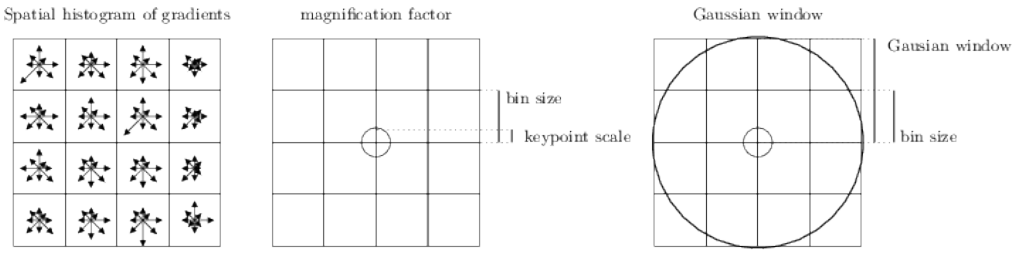
Detektor SIFT přiřazuje k bodům zájmu jejich velikost (ve formě kruhové plochy) a směr (resp. jejich úhel). Samotná detekce bodů zájmu probíhá v několika krocích. Nejprve je obraz převeden na hodnoty intenzity, výsledek je poté filtrován Gaussovým filtrem s postupně se zvětšujícími jádry [15]. Množina filtrovaných obrazů poté tvoří 3D matici o velikosti $I \times J \times K$, kde I a J jsou rozměry fotografie a K je počet filtrování obrázku. V této matici jsou poté nalezeny lokální extrémy. Pro určitou vstupní hodnotu je prohledán $3 \times 3 \times 3$ okolní blok. Vstupní hodnota je prohlášena za extrém, pokud jsou všechny hodnoty v tomto okolí buď menší, nebo větší.

Získali jsme tedy množinu kandidátních bodů, kterou je ale třeba dále redukovat. V každém bodu je vypočítán tenzor struktury obrazu dle rovnice 4.1 a poté jsou odstraněny kandidátní body v hladkých regionech, určených dle velikosti hodnoty tenzoru.

V rovnici 4.1 S_{ij} odpovídá tenzoru struktury obrazu na pozici (i, j), vypočítanému ve čtvercové oblasti o velikosti $(2D + 1) \times (2D + 1)$ kolem současné pozice. h_{mn} označuje odezvu horizontálního filtru na pozici (m, n) a v_{mn} označuje odezvu vertikálního filtru na této pozici. Nakonec w_{mn} označuje váhování dané vzdáleností od středu (i, j).

Bodům redukované kandidátní množiny je nakonec přiřazena hodnota amplitudy a směru lokálního gradientu.

$$S_{ij} = \sum_{m=i-D}^{i+D} \sum_{n=j-D}^{j+D} w_{mn} \begin{bmatrix} h_{mn}^2 & h_{mn}v_{mn} \\ h_{mn}v_{mn} & v_{mn}^2 \end{bmatrix} \tag{4.1}$$



Obrázek 4.2: SIFT deskriptor, obrázek převzat z [20].

Deskripce

Z předchozího kroku byly získány význačné body v obraze, které mají přiřazenu pozici, měřítko a orientaci a jsou tedy vůči těmto parametrům neměnné. Nyní je třeba získat popis regionu obrazu, který je neměnný i vůči změně světlosti či 3D pohledu.

Pro dosažení tohoto cíle vzorkujeme oblast kolem každého keypointu, ze které poté získáme menší pole deskriptorů (Lowe ve své práci uvádí region 16×16 vzorků, ze kterých je získáno pole 4×4). Měřítko daného keypointu udává parametry Gaussova filtrování a orientace keypointu ovlivňuje orientaci a koordináty deskriptoru. Následně je v regionu provedeno Gaussovské váhování na základě vzdálenosti vzorku od keypointu. Z jednotlivých hodnot je poté vytvořen histogram orientací vzorků, který je uložen do patřičné buňky pole deskriptorů.

4.3.2 SURF

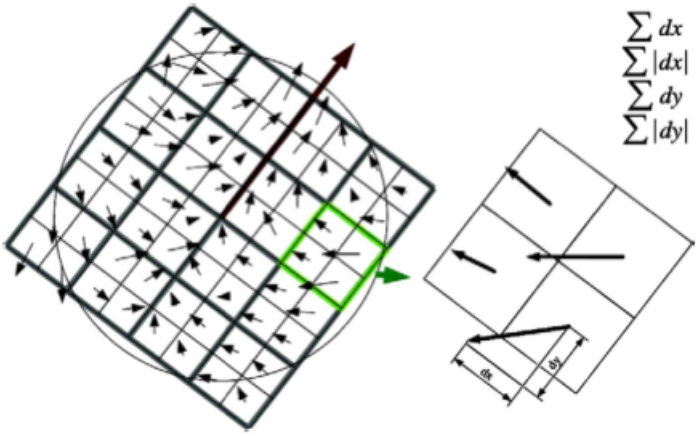
Algoritmus SURF (Speeded-Up Robust Features, viz [1]) publikoval Herbert Bay v roce 2006 jako vylepšení existujícího algoritmu SIFT. SURF urychluje výpočet tím, že nevytváří 3D reprezentaci několika měřítek různými Gaussovými filtry, ale pouze tento prostor aproximuje čtvercovými funkcemi, které mají konstantní časovou složitost (oproti logaritmické u Gaussova filtru). Značnou nevýhodou algoritmu podobně jako u SIFT je, že je patentově chráněný.

Zaznamenanou výhodou oproti algoritmu SIFT je větší robustnost (s výjimkou u změny perspektivy a různých zakřivení, vůči kterým není invariantní). SURF v průměru detekuje menší počet keypointů, což vychází ze snahy o zrychlení algoritmu. Dle měření[13] je SURF výhodný v případě, že potřebujeme rychle nalézt odpovídající body v obraze (naříklad u záznamu kamery v reálném čase). Pokud nám na výpočetním čase nezáleží, tak SIFT poskytuje lepší výsledky.

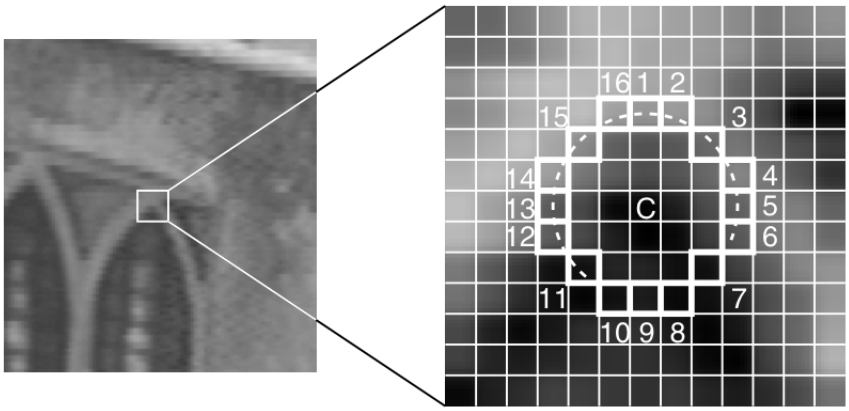
4.3.3 FAST

Features from Accelerated Segment Test[16] (FAST) je algoritmem pro detekci význačných bodů ve formě hranové detekce. Jeho název není zvolen náhodně, uváděnou výhodou algoritmu je opravdu výpočetní efektivnost oproti jiným běžným detektorům jako je například rozdíl Gaussových filtrů využívaný u algoritmu SIFT nebo Harrisova detektoru.

Způsob deskripce je ilustrován na obrázku 4.4. Algoritmus vybere kandidátní pixel C, kolem kterého poté kontroluje 16 pixelů uspořádaných na kružnici. Detektor vyhodnotí bod obrazu jako rohový, pokud má n souvisle jdoucích pixelů na kružnici větší hodnotu jasu, než kandidátní pixel I_p zvýšený o hodnotu prahu t, resp. menší hodnotu jasu než $I_p - t$. Hodnota n byla zvolena na 12 pixelů, který umožňuje aplikaci rychlého testování.



Obrázek 4.3: SURF deskriptor, obrázek převzat z [14].



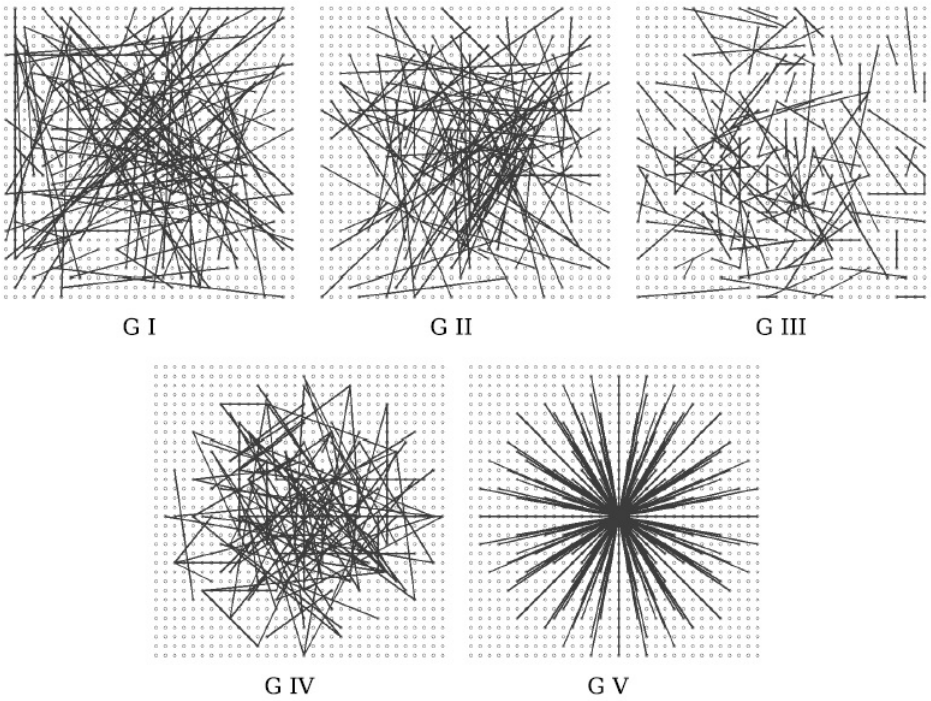
Obrázek 4.4: FAST detekce bodu, obrázek převzat z [14].

Rychlé testování nejprve kontroluje hodnoty jasu pixelů 1 a 9 (tedy pixely, které se nachází naproti sobě). Pokud oba body spadnou do rozsahu hodnot daných kandidátním pixelem I_p a prahem t, tak je bod vyřazen z dalšího testování (nejde o rohový bod). Jinak se pokračuje dále v testování a stejným způsobem jsou zkontrolovány pixely 5 a 13.

Jestliže projde i druhý test, tak zbývají čtyři trojice pixelů. Aby byl kandidátní bod vyhodnocen jako rohový, tak musí alespoň jedna trojice být jasnější než $I_p + t$, nebo tmavší než $I_p - t$. Pokud ani jedna podmínka není splněna, tak o rohový bod nejde.

Je vidět, že detektor může poměrně rychle odmítat nevyhovující body. Tento přístup má ale několik nevýhod:

- Pro $n < 12$ není testování tak efektivní, protože může jít o roh, pokud jsou i jen dva pixely tmavší nebo jasnější než I_p.
- Efektivita detektoru závisí na distribuci kandidátních bodů.
- Je vysoká pravděpodobnost detekce několika rohů vedle sebe v jedné oblasti.



Obrázek 4.5: Různé typy vzorkování BRIEF deskriptoru, obrázek převzat z [2].

4.3.4 BRIEF

Binary Robust Independent Elementary Features [2] (BRIEF) přináší nový přístup ohledně deskripce význačných bodů v podobě oproštění od datového typu plovoucí řádové čárky a přechodu k binární reprezentaci.

Deskriptor BRIEF je tvořen pomocí množiny testů τ intenzity dvou pixelů, jejichž výsledek tvoří patřičnou hodnotu 1 nebo 0 ve vektoru deskriptoru.

$$\tau(p; x, y) = \begin{cases} 1, & p(x) < p(y) \\ 0, & \text{jinak} \end{cases} \tag{4.2}$$

V rovnici $p(x)$ odpovídá intenzitě pixelu po aplikaci vyhlazovacího jádra v bodě $x = (u, v)^\top$. Zvolením n_d párů souřadnic (x, y) získáme množinu binárních testů, které následně tvoří deskriptor ve formě n_d-dimenzionálního bitového řetězce, definovaný jako

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{4.3}$$

Publikované dimenze tohoto řetězce jsou $n_d = 128$, 256 a 512, které se ukázaly jako dobrý kompromis mezi rychlostí, množstvím uchovávaných dat a přesností.

Zmíněné vyhlazovací jádro se používá z důvodu náchylnosti testů na šum. Protože se provádí testování pouze jednotlivých pixelů, tak případný šum zanesený do obrazu může způsobit neplatné porovnání a tvorbu nepřesného deskriptoru. Proto se vstupní obraz nejprve vyhladí a až poté se hledají hrany na úrovni pixelů.

Samotné testy jsou prováděny s různými prostorovými vzory, které jsou znázorněny na obrázku 4.5. Jednotlivé obrázky znázorňují následující vzorkování:

1. Rovnoměrné rozložení – Souřadnice (x_i, y_i) jsou rovnoměrně rozloženy v testované oblasti.

2. Gaussovo rozložení – Souřadnice jsou rozloženy podle Gaussovy distribuční funkce.

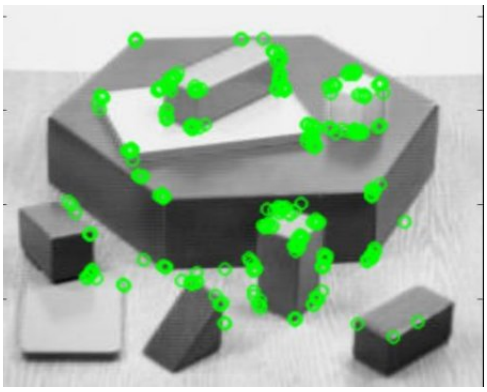
3. Dvoukrokové Gaussovo rozložení – Složeno ze dvou kroků. První souřadnice x_i je získána Gaussovou distribucí z počátku a následně je další souřadnice získána další Gaussovou distribuční funkcí vystředěnou na x_i.

4. Náhodné rozložení – Vzorky jsou získávány z náhodných polárních souřadnic.

5. Souvsilé rozložení – Vzorkuje všechny souvislé souřadnice v soustavě polárních souřadnic, které obsahují n_d bodů.

4.3.5 ORB

ORB [17] je kombinací FAST detektoru keypointů a BRIEF deskriptoru. Výkonností je na úrovni předchozích algoritmů a není patentově chráněn. Po detekci keypointů algoritmem FAST provede redukci bodů pomocí Harrisova detektoru rohů a vytvoří pyramidově model o několika měřítkách. Protože algoritmus FAST sám o sobě nevypočítává v bodech orientaci daného gradientu, tak autoři ORB vytvořili modifikaci, která vypočítá geometrický střed regionu na základě vážení intenzity a poté vytvoří vektor k tomuto geometrickému středu směrem od nalezeného rohového bodu (výstup Harrisova detektoru). Tento vektor se stane orientací keypointu.



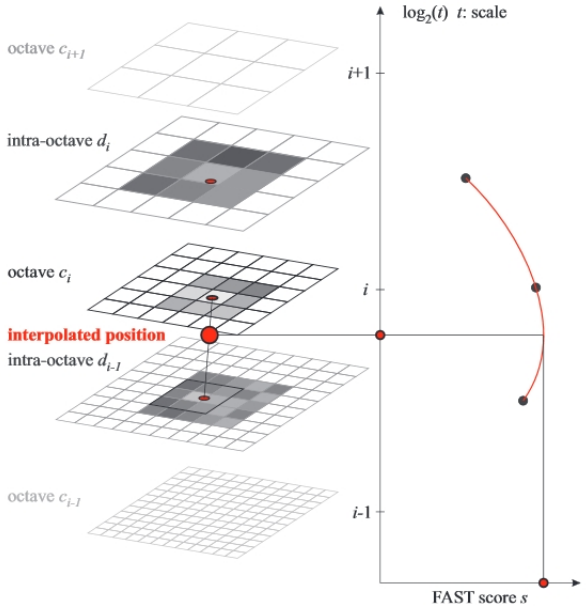
Obrázek 4.6: ORB detekce. Obrázek převzat z [12].

4.3.6 BRISK

Tento algoritmus publikovali v roce 2011 Stefan Leutenegger, Margarita Chli a Roland Siegwart [8]. Dle zveřejněné publikace by měl být až o řád rychlejší než algoritmus SURF díky použití FAST detektoru a porovnávání deskriptorů pomocí bitových operací.

Detekce

Při detekci keypointů algoritmus BRISK vychází z již zmíněného algoritmu FAST. Modifikací je vyhledávání lokálního maxima nejen v samotném obrázku, ale i v měřítkovém prostoru s využitím FAST hodnocení pro měření významnosti. Měřítkový prostor je opět

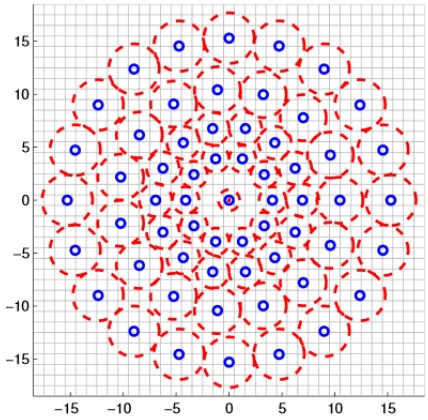


Obrázek 4.7: Měřítkový prostor algoritmu BRISK [8].

tvořen určitou "pyramidou"vstupního obrazu s různými měřítky podobně jako u předchozích algoritmů. BRISK využívá pro diskretizaci prostoru větší intervaly, pravé měřítko keypointu poté odhaduje. Tato interpolace je ilustrována na obrázku 4.7.

Deskripce

Po extrahování sady keypointů z obrazu je vytvořen deskriptor. BRISK deskriptor je unikátní díky tomu, že je tvořen binárním řetězcem. Opět je vzorkováno okolí každého keypointu, tentokrát ale kruhovým vzorem o N bodech, kde okolí každého vzorkovacího bodu je vyhlazeno Gaussovým jádrem. Toto vyhlazení slouží k váhování vlivu dané hodnoty na vzorek. Vzorkovací vzor je vidět na obrázku 4.8. Bitový vektor deskriptoru d_k je poté sestaven porovnáním intenzit párů vzorkovacích bodů (p_i, p_j), které spadají do určité definované



Obrázek 4.8: Vzorkovací vzor okolo keypointu u algoritmu BRISK [8].

minimální vzdálenosti.

$$b = \begin{cases} 1, & I(p_j, \sigma_j) > I(p_i, \sigma_i) \\ 0, & \text{jinak} \end{cases} \tag{4.4}$$

Funkce $I(p, \sigma)$ z rovnice 4.4 odpovídá intenzitě vzorkovacího bodu s Gaussovým vyhlazovacím jádrem σ.

Pro porovnání dvou deskriptorů stačí díky binární reprezentaci pouze vypočítání jejich Hammingovy vzdálenosti. Pomocí té získáme počet rozdílných bitů, který vypovídá o rozdílnosti dvou deskriptorů. Takové porovnání sestává pouze z operace XOR nad dvěma deskriptory a následné spočítání bitů, což může být provedeno rychle.

4.3.7 MSER

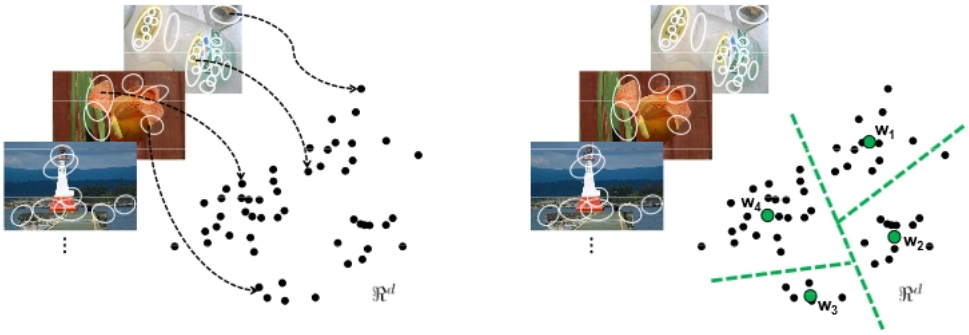
MSER je zkratkou pro *Maximally stable extremal regions* a oproti předchozím metodám je použitelný pro detekci celých význačných regionů místo jednotlivých bodů a jejich bezprostředního okolí. Jeho použití je tedy poměrně zajímavé ohledně tvorby prostorové informace o vstupním obrázku a vytvoření seznamu shluků, které by obrázek mohly charakterizovat.

Konceptem je nalezení extrémních regionů, které jsou extrémní svojí souvislou intenzitou, která se výrazně liší oproti svému okolí. Algoritmus se často využívá pro detekci textu v obraze, protože právě text výborně splňuje tyto parametry.

Obrázek 4.9: Příklad MSER detekce.

4.4 Bag of Words/Features

Model Bag of Words původně vznikl pro potřeby porovnávání textových dokumentů. Dokument byl brán jako množina (histogram) v něm obsažených slov a jejich frekvencí bez ohledu na gramatiku nebo slovosled. Model je používán například pro detekci spamu, kde máme vytvořený histogram spamových slov a histogram běžného e-mailu a pomocí Bag of



Obrázek 4.10: Ilustrace shlukování BoW deskriptorů.

Words modelu zjišťujeme, zda histogram vstupního e-mailu odpovídá spíše spamu, nebo jde o nezávadný text.

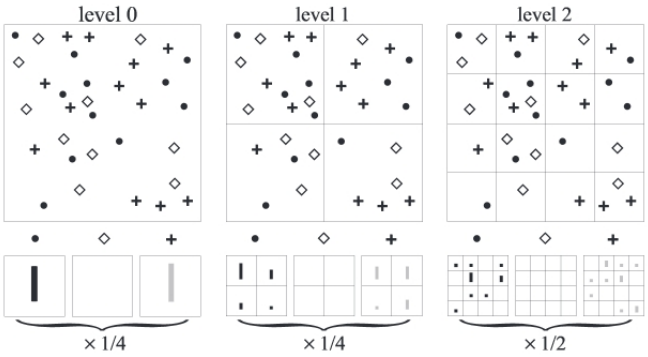
Tento princip byl následně převzat oborem počítačového vidění pro podobné použití při vyhledávání a klasifikaci fotografií [4]. Dokumentem je zde daný obrázek, ve kterém vyhledáváme význačné body či regiony, které poté popíšeme deskriptorem. Deskriptory získáme některým z již zmíněných algoritmů a používáme je pro vytvoření slovníku vizuálních slov. Tento slovník nejprve sestává z množiny nalezených deskriptorů a nad touto množinou je poté proveden některý algoritmus shlukování(viz obr. 4.10). Tím je vytvořena množina předem určeného počtu vizuálních slov.

Algorithm 1 Bag of Features

1: Inicializace detektoru a extraktoru deskriptorů.
2: Inicializace matice M, která bude obsahovat data ke shlukování.
3: **for** i in range 0 .. počet fotografií **do**
4: Načti obrázek i.
5: Nalezni pomocí detektoru množinu význačných bodů.
6: Proveď extrakci příznaků.
7: Přidej příznaky do matice M.
8: **end for**
9: Proveď nad maticí M k-means shlukování. Produktem je slovník.
10: **for** i in range 0 .. počet fotografií **do**
11: Vypočítej Bag of Words deskriptor pro fotografii i.
12: **end for**

Shlukování se provádí například algoritmem *k-means* [11]. K-means dělí prostor s deskriptory na n shluků tak, že nejprve náhodně rozmístí n těžišť a poté přiřadí jednotlivé deskriptory ke svému nejbližšímu těžišti. Tím získáme n shluků dat, ve kterých je jejich těžiště přepočítáno a body jsou opět přiřazeny nejbližšímu těžišti. Algoritmus ideálně běží do chvíle, kdy již není žádné těžiště přesunuto, nicméně k takové ideální konvergenci může dojít až za velmi dlouho. Proto běh algoritmu v praxi omezujeme buďto na maximální počet iterací, nebo na minimální vzdálenost mezi těžišti, jejímž dosažení je již výsledek prohlášen za konečný. Běh algoritmu je popsán rovnicí 4.5.

$$D(X, M) = \sum_{i=0}^{n} \sum_{x_j \in C_i} (x_j - m_i)^2 \tag{4.5}$$



Obrázek 4.11: Spatial Pyramid Matching se třemi úrovněmi.

4.5 Spatial Pyramid Matching

Pokud použijeme výše uvedený postup, tak získáme slovník vizuálních slov bez informace o jejich umístění v prostoru. Může se tedy stát, že získáme deskriptory odpovídající určitému objektu (například budova), poté spustíme vyhledávání nad nějakou fotografií a v této fotografii budou nalezeny podobné deskriptory, ale půjde o naprosto rozdílný objekt, který měl pouze podobné vlastnosti. Význačné deskriptory nám tedy vrátí velkou podobnost, ale prostorové rozmístění těchto deskriptorů neodpovídá. Proto může být zlepšním uvažovat nějakým způsobem i umístění deskriptorů.

K tomuto cíli slouží algoritmus Spatial Pyramid Matching (viz [7]). Tento algoritmus postupně rozděluje vstupní fotografii na síť buněk s postupně se zvyšující hustotou a vypočítává histogramy výskytu slov pro každou buňku (viz rovnice 4.6).

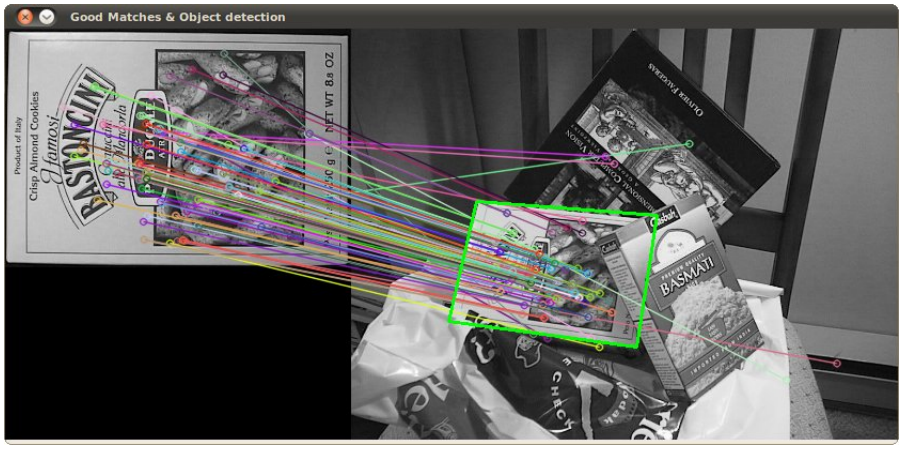
$$K^L(X,Y) = \sum_{m=1}^{M} k^L(X_m, Y_m) \tag{4.6}$$

Vzorec 4.6 říká, že Histogram K úrovně L získáme jako sumu výskytů vizuálních slov m (z celkového počtu M), které se vyskytují v dané úrovni L sítě buněk. Pokud by platilo $L = 0$, tak by byl algoritmus zredukován na prosté Bag of Visual Words. Na jednotlivé úrovně sítě je aplikována váha tak, že stejné výskyty v hustší síti mají větší vliv než hodnoty v méně husté síti (viz obr. 4.11).

4.6 Homografie

Spatial Pyramid Matching je prováděn ještě nad deskriptory na relativně nízké úrovni v procesu vyhledávání fotografie, ale je možné vyhledání zpřesnit až když máme kandidáty vybrané a seřazené. K tomuto lze využít homografii. Proces homografie ve zpracování obrazu se pokouší nalézt mapování keypointů jedné fotografie na nějaké keypointy druhé fotografie tak, aby byla zachována jejich sousednost. Algoritmus nejprve nalezne podobné deskriptory například algoritmem *FLANN* (Fast Approximate Nearest Neighbor Search) a následně se pokusí mapovat pozice odpovídajících bodů vstupní fotografie na pozice fotografie z databáze (viz obr. 4.12). Fotografie s dobrou shodou jsou poté ve výstupu upřednostněny před ostatními.

Pro potřeby homografie je používán algoritmus *RANSAC* (Random Sample Consensus), který publikovali Fischler a Bolles v roce 1981 (článek dostupný na [3]). Oproti jiným



Obrázek 4.12: Homografie pomocí funkcí OpenCV. Zdroj dokumentace OpenCV.

metodám, které začínají na co největší množině dat a postupně se snaží tuto množinu redukovat pouze na relevantní body, RANSAC nejprve ze vstupních dat vytvoří minimální množinu, kterou poté rozšiřuje podle potřeby.

RANSAC je iterativní algoritmus, který pracuje ve dvou krocích(viz [22]):

- *Hypotéza.* Algoritmus vybere minimální sadu vzorků, ze kterých vypočítá parametry hledaného modelu. Kardinalita sady odpovídá minimu pro stanovení těchto parametrů.

- *Testování.* Algoritmus testuje vytvořenou sadu vzorků na vstupních datech a ověřuje, zda je hypotéza aplikovatelná. Pokud nalezne dostatek blízkých bodů, tak použije tuto informaci k zpřesnění parametrů a pokud pravděpodobnost nalezení lepší sady neklesla pod určitou mez, tak pokračuje v další iteraci.

Příkladem z literatury je mapování kružnice na sadu dvoudimenzionálních bodů. RANSAC nejprve vybere minimální množinu tří bodů (které tvoří odhad středu a poloměru, tedy parametrů dané kružnice) a poté vyhledává takové body, které splňují určitou minimální vzdálenost k této kružnici. Pokud je nalezen dostatek blízkých bodů, tak RANSAC aplikuje některou vyhlazovací techniku (např. *least squares*) a vypočítá nové přesnější parametry kružnice.

Použití homografie se mi v aplikaci zatím osvědčilo, je ale třeba ji aplikovat na již redukovanou sadu nejpravděpodobnějších kandidátů, protože proces mapování je výpočetně náročný.

Kapitola 5

Využití databází pro indexování fotografií

Další kapitolou, které se budu věnovat, je způsob uložení již získaných dat. Za získáním keypointů, deskriptorů a vytvořením slovníku Bag of Words je velké množství výpočetního výkonu a zároveň i času. Tyto data je ovšem třeba vypočítat pouze jednou, pro další práci již potřebujeme jen Bag of Words deskriptor přiřazený ke každé fotografii z databáze. Je tedy vhodné data nějakým způsobem ukládat pro pozdější použití. Je možné matice deskriptorů ukládat přímo na disk do nějakého souboru, nicméně tím se připravujeme o možnost rychlého vyhledání dat pomocí indexování, případně vyhledávání podobnosti a rozsahu v množině redukované právě indexováním. Tím se dostáváme k rozboru vhodnosti použití některé databáze pro potřeby implementované aplikace.

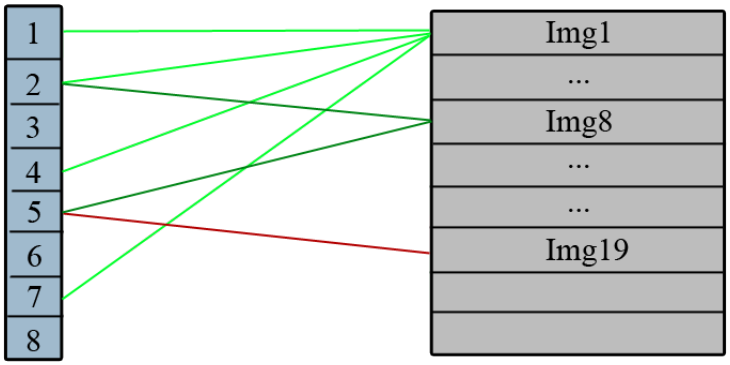
5.1 Indexování

Tvorba indexu nad daty je známým konceptem a výhodou využití databáze pro ukládání dat. Bez využití indexování bychom museli při vyhledávání fotografie sekvenčně procházet všechny data v databázi a postupně porovnávat deskriptory. S využitím indexování můžeme potřebný čas k nalezení záznamu redukovat, protože záznamy, nad kterými vytvoříme index, jsou zpracovány a seřazeny ve vyhrazené struktuře pro umožnění použití vyhledávacího algoritmu (např. binární vyhledávání). Problémem je, že klasické indexy pracují zpravidla nad jedinou konkrétní hodnotou (například *ID* fotografie). My ale ID hledané fotografie neznáme, vyhledáváme na základě podobnosti. Proto nám klasické indexování k užitku příliš nebude a musíme využít jiné metody. Zde je dobré popsat tzv. *invertované indexování*.

5.1.1 Invertovaný index

Invertovaný index vznikl pro potřeby *fulltextového* vyhledávání. Umožňuje vyhledávání na základě výskytu určitých slov v textu a nevrací konkrétní záznam, ale množinu záznamů, kde se daná slova vyskytují. Vystavění inverzního indexu je náročnější než u klasického indexu, nicméně je třeba jej vytvořit jen při přidání nových dat, což je u galerie fotografií méně obvyklé, než samotné vyhledávání.

Invertovaný index je vystavěn tak, že je nad daty v databázi vytvořen slovník slov, ke kterým je zároveň připojena informace o jejich výskytu v určitých záznamech. Při dotazování nad nějakou větou je poté opět získán slovník slov této věty a pro jednotlivá slova jsou



Obrázek 5.1: Princip dotazu s využitím invertovaného indexu.

získány množiny výskytů v databázi. Index poté vrátí množinu pravděpodobných záznamů jako průnik jednotlivých množin výskytů (ilustrováno na obrázku 5.1).

Tento model je aplikovatelný na zde řešenou aplikaci, protože vyhledávání fotografií dle obsahu pracuje na stejném principu. Každá fotografie v tabulce sestává z množiny vizuálních slov a my potřebujeme nalézt takové záznamy, u kterých se vyskytují podobná vizuální slova.

5.2 PostgreSQL

Jde o open source objektově relační databázový systém, který klade důraz na rozšiřitelnost a dodržování standardů. Výhodou PostgreSQL v porovnání s například MySQL je důraz na výkon při komplexnějším vyhledávání a již zmíněná rozšiřitelnost. Také jsem nalezl několik implementací práce s histogramy, které se svým zaměřením blížily řešenému problému [19]. Proto jsem PostgreSQL zvolil jako cílovou databázi pro ukládání informací o fotografiích.

5.2.1 Indexování v PostgreSQL

PostgreSQL podporuje dva typy indexů pro fulltextové vyhledávání a vhodnost daného typu závisí na způsobu použití dat v databázi. Využití jednoho z těchto indexů značně urychluje fulltextové vyhledávání a proto je výhodné adaptovat informace o fotografiích tak, aby bylo možné tyto indexy použít i pro indexování a vyhledávání obrázků.

GiST

Generalized Search Tree (GiST) je ztrátovým indexem, což znamená že může vracet falešné záznamy a jejich korektnost je třeba zkontrolovat přístupem k danému řádku v databázi. Ztrátovost je způsobena typem reprezentace každého dokumentu, který je reprezentován svým podpisem pevné délky, získaným hash funkcí provedenou nad každým slovem. Každé slovo v n-bitovém řetězci je konvertováno na 1 bit a tyto bity jsou následně logickým OR spojeny do výsledného podpisu dokumentu. Využití hash funkce a redukcí dokumentu do mnohem menšího počtu bitů tedy implikuje možné kolize. Tyto kolize zpomalují vyhledávání, protože každá kolize musí být zkontrolována přímo v daném řádku databáze. Riziko kolize se snižuje s vyšším počtem unikátních slov, čili záleží na datech, která budou v databázi obsažena.

Vytvoření GiST indexu nad nějakým sloupcem je velmi jednoduché, stačí použít následující PostgreSQL příkaz:

```
CREATE INDEX name ON table USING gist(column);
```

GIN

Generalized Inverted Index (GIN) implementuje již zmíněný invertovaný index, tedy ke každému databázovému záznamu udržuje seznam lexémů (slov v textu, případně hodnot histogramu v našem případě), které se v tomto záznamu vyskytují a umožňuje vyhledávání nejpodobnějšího záznamu na základě těchto lexémů. Vlastnosti GIN jsou:

- Vyhledávání v GIN indexu je přibližně 3x rychlejší než v GiST.
- Vybudování GIN indexu trvá přibližně 3x déle než vybudování GiST indexu.
- Aktualizace dat v GIN indexu je pomalejší než v GiST indexu.
- GIN indexy zabírají dvakrát až třikrát více prostoru než GiST indexy.

Vytvoření GIN indexu je stejně snadné jako u GiST, k vytvoření dojde následujícím příkazem:

```
CREATE INDEX name ON table USING gin(column);
```

5.2.2 PostgreSQL-IE

PostgreSQL-IE[5] je zkratkou pro *PostgreSQL with Image-handling Extension* a jde o rozšíření databáze o nástroje a funkce pro operace nad obrazovými daty. V současnosti obsahuje dvě procedury pro vyhledání podobnosti pomocí algoritmu *k-nearest neighbours* a *range search*. Dále obsahuje dvanáct procedur pro extrakci příznaků, z toho jednu na základě histogramu barev a zbylých jedenáct na základě struktury.

PostgreSQL-IE zavádí nový datový typ *PGImage*, který zapouzdřuje ukazatel do tabulky, která slouží k ukládání množiny atributů, jako jsou

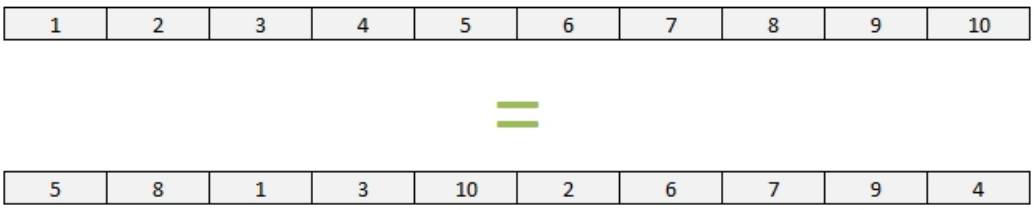
- Binární forma fotografie ve struktuře BLOB
- Rozměry fotografie
- Formát fotografie

Dále zavádí množinu nových SQL dotazů označených jako *SQL-IE*, která obsahuje 16 metod pro extrakci příznaků, tvorbu deskriptoru, vložení nové fotografie, přiřazení deskriptoru dané fotografií a vyhledání na základě podobnosti. Funkce pro vyhledání podobnosti jsou integrovány jako poddotazy dotazu WHERE a návrat dotazu je již seřazený seznam podobných záznamů. Získané záznamy je možné ovlivňovat pomocí explicitní specifikace hodnocení funkcí *Score* přímo v dotazu SELECT.

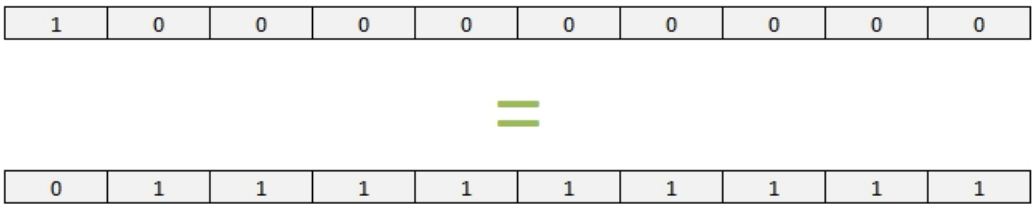
Problémem tohoto rozšíření je špatná dostupnost dokumentace, která dle mých informací není dostupná v angličtině. Proto není mnoho popsaných aplikací, které toto rozšíření využívají a seznámení s funkčností je obtížné.

5.2.3 Smlar

Smlar není přímo knihovnou, jde spíše o rozšíření PostgreSQL o funkce pro porovnávání podobnosti dvou polí hodnot a tvorbou indexu nad touto množinou. Podporuje jak GIST, tak GIN indexování a je tedy vhodný pro širokou škálu aplikací. Volání funkce je součástí dotazu SELECT a vrací float hodnotu mezi 0 a 1, označující do jaké míry si jsou dvě pole podobná. Nevýhodou této funkce je, že nijak nekontroluje pořadí a počet hodnot a porovnává pouze jednotlivé unikátní hodnoty. Pro použití pro indexování Bag of Words je tedy třeba pole hodnot určitým způsobem modifikovat, o čemž bude pojednávat kapitola zabývající se implementací.



Obrázek 5.2: Nevýhoda porovnávání ohledně pořadí.



Obrázek 5.3: Nevýhoda porovnávání ohledně duplicitních hodnot.

Rozšíření Smlar je možné také volat pomocí binárního operátoru %, který odpovídá volání funkce smlar se základním nastavením a vrací hodnotu True/False, pokud provedení funkce na daném databázovém záznamu vrátilo hodnotu, která byla stejná či vyšší než nastavený práh. Tento práh je možné nastavit nastavením hodnoty `smlar.threshold` v konfiguračním souboru `postgresql.conf`.

Je tedy možné získat hodnotu podobnosti dvou polí pomocí příkazu:

```
SELECT smlar('{1,2,3}'::int[], '{3,4,5}' );
```

Případně se dotazovat na True/False výsledek s nastaveným prahem podobnosti:

```
SELECT '{1,2,3}'::int[] % '{3,4,5}'::int[] as similar;
```

Funkce smlar poté ještě přijímá třetí parametr, kterým je možné specifikovat funkci, podle které dojde k porovnání hodnot. K tomuto účelu je v rozšíření implementováno několik interních proměnných, kterými jsou:

- N.i – Počet společných prvků v obou polích.
- N.a – Počet unikátních prvků v prvním poli.

- N.b – Počet unikátních prvků v druhém poli.

S těmito proměnnými je poté možné specifikovat konkrétní metriku porovnávání:

```
SELECT smlar('{1,4,6}'::int[], '{5,4,6}', 'N.i / sqrt(N.a * N.b)' );
```

5.3 Vyhledávání

Zkoumal jsem několik algoritmů, respektive aplikací, které jsou specializované na vyhledávání v nějaké množině dat za pomoci klíčových slov. Důležitou otázkou je, zda je možné některé rozšíření využít pro potřeby této práce. V následujících odstavcích uvedu relevantní aplikace a shrnu dosavadní poznatky.

5.3.1 Lucene

Apache Lucene je zdarma šířená open source knihovna sloužící k indexování textu a rychlému vyhledání požadovaných dat. Knihovna byla původně napsána v jazyku Java, ale dnes už je možné ji získat i pro jazyky Delphi, Perl, C#, C++, Python, Ruby a PHP a spolupracuje s velkým počtem databázových systémů včetně PostgreSQL. Lucene samo o sobě se zabývá především vyhledáváním textu, přinejlepším alfanumerickými znaky, proto jej momentálně pro další vývoj neuvažuji.

Zajímavou možností pro implementaci je rozšíření Lucene o vyhledávání obrazových dat s názvem LIRe (Lucene Image REtrieval, viz [10]). Jde o knihovnu, napsanou pro jazyk Java, která rozšiřuje Lucene o metody pro extrakci příznaků z fotografií a indexování získaných dat za pomocí algoritmů Lucene.

5.3.2 Sphinx

Sphinx je další zdarma šířený multiplatformní open source vyhledávač, který také pracuje nad textem. Je napsaný v jazyce C++ a taktéž umožňuje práci s rozšířenými databázemi jako MySQL a PostgreSQL. Oproti Lucene jsem ale nenalezl žádné rozšíření pro práci nad obrazovými daty, deskriptory, nebo histogramy.

Kapitola 6

Návrh aplikace pro vyhledávání fotografií

V této kapitole rozeberu návrh samotné aplikace. Kapitola je věnována využitým algoritmům a důvodům, které vedly k jejich zvolení. Dále je zde popsán návrh funkčnosti aplikace a způsobu běhu a interakce s uživatelem.

6.1 Cíle

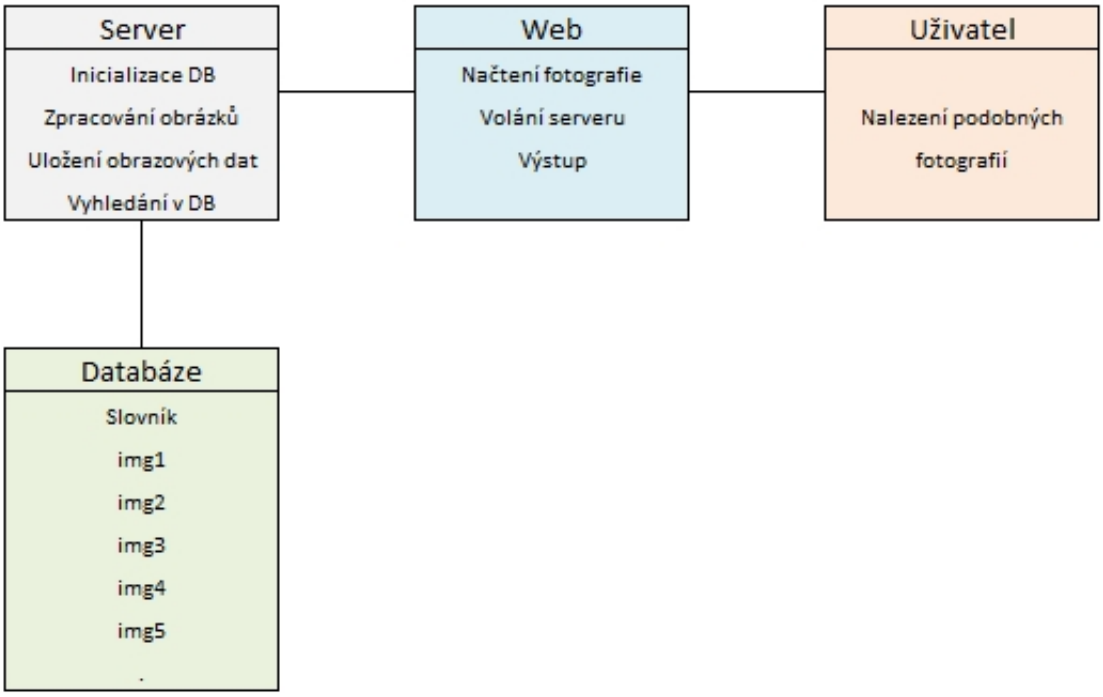
Vyhledávání fotografií podle obsahu je poměrně široké téma, u kterého se lze soustředit na mnoho aspektů. Je možné vytvořit aplikaci cílenou na vysokou přesnost a porovnávání fotografií na celé řadě parametrů, případně cílenou na rychlost a velké databáze trénovacích dat. Plánovaná aplikace spadá do druhé kategorie, cílem je vytvoření škálovatelného serverového procesu, který je schopný vytvořit databázi vizuálních slov z velké množiny fotografií (tisíce až stovky tisíc, potenciálně až miliony) a nad touto množinou fotografií poté provádět rychlé dotazy ohledně podobnosti.

Je tedy třeba vybrat vhodný databázový systém, který bude schopný pracovat s takovým objemem dat a bude buďto obsahovat vhodné nástroje, nebo bude možné jej rozšířit o požadovanou funkčnost. Nad tímto databázovým systémem bude implementována aplikace, která zpracuje galerii fotografií a připraví ji pro následné uživatelské dotazy, které by měla být schopna zpracovat v řádu sekund.

6.2 Schéma funkčnosti

Na obrázku 6.1 je ilustrováno schéma návrhu aplikace. Aplikace je rozdělena na několik částí, kterými jsou:

- Server – Jádro aplikace, implementované v jazyce C++. Zde jsou prováděny všechny základní operace, jako je inicializace databáze, načtení a zpracování vstupní složky s fotografiemi, vypočítání slovníku vizuálních slov a přiřazení Bag of Words deskriptorů patřičným obrázkům. Serverový proces jako jediný komunikuje s databází a zpracovává její výstupy.
- Databáze – Implementovaná v databázovém systému PostgreSQL s využitím rozšíření Smlar. Do databáze se postupně ukládá slovník vizuálních slov a jednotlivé záznamy



Obrázek 6.1: Schéma návrhu aplikace.

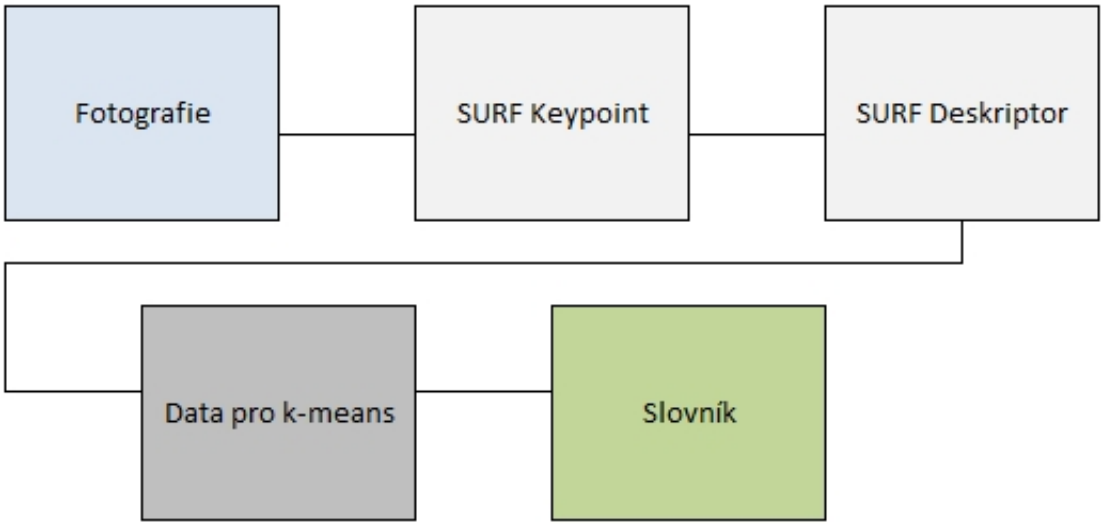
fotografií. Způsob uložení bude dále rozebrán v kapitole Implementace. Nad fotografiemi je vytvořen invertovaný index, urychlující následné vyhledávání. Rozšíření Smlar se stará o vyhledávání na základě podobnosti dvou vektorů hodnot a vrací předem definovaný počet nejlepších shod serverovému procesu pro další zpracování.

- Web – Slouží jako uživatelské rozhraní pro ovládání aplikace. Zde uživatel může nahrát fotografii, jejíž varianty chce v databázi nalézt. WEbová část je implementovaná využitím jazyků PHP, HTML, Javascript a CSS a provádí přímé volání serverové aplikace, která načte vstup, zpracuje výstup z databáze a vrátí fotografie s nejlepší shodou.
- Uživatel – Uživatel musí aplikaci poskytnout vzorovou fotografii skrze webový formulář, kterým bude fotografie nahrána do dočasného úložiště na serveru a následně zpracována serverovým procesem. Skrze webový formulář poté uživatel dostane galerii nejlepších shod se vstupní fotografií.

6.3 Metody a algoritmy

Nalezení význačných bodů a extrakce deskriptorů je navržena s využitím algoritmu SURF. Důvodem je především snaha o ušetření prostoru v paměti, protože algoritmus SURF popisuje keypointy pomocí poloviního počtu hodnot oproti algoritmu SIFT (64 oproti 128). Tím je možné provádět shlukování na dvojnásobném počtu deskriptorů a buďto navýšit přesnost, nebo navýšit počet fotografií v databázi.

Shlukování využivá algoritmu k-means, který byl popsán již dříve. Výhodou algoritmu je lineární časová složitost. Navíc také nemá žádné větší požadavky ohledně dostupné paměti



Obrázek 6.2: Schéma tvorby slovníku.

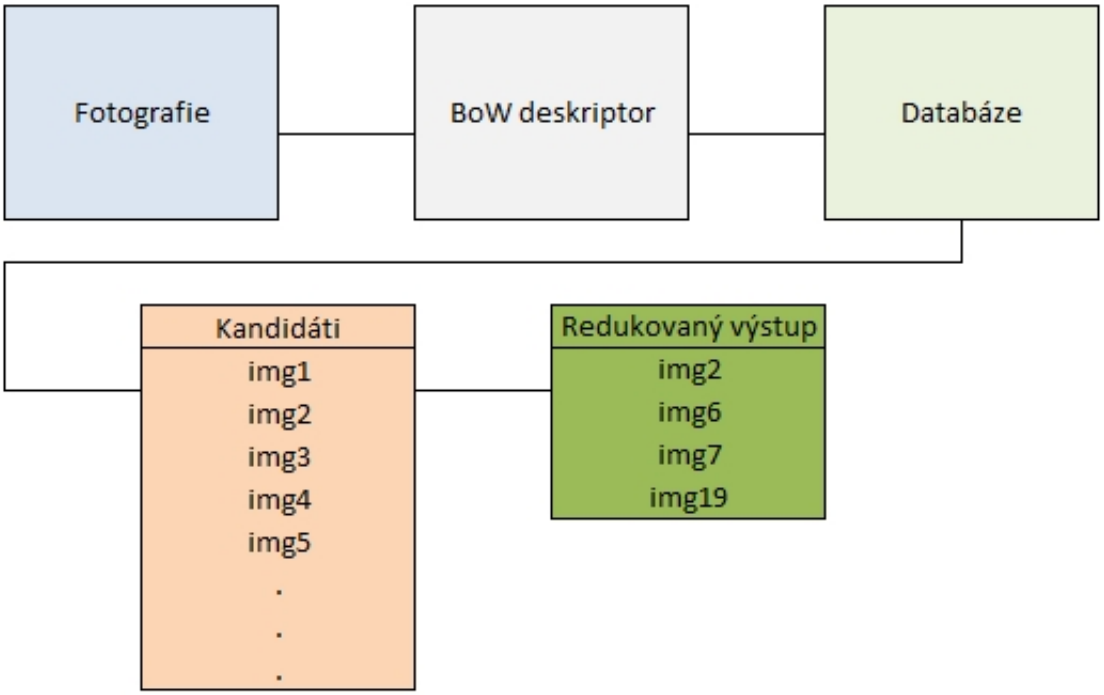
kromě nutnosti přítomnosti všech deskriptorů ke shlukování. Protože vstupní deskriptory samy o sobě zabírají obrovské množství prostoru, tak je nízká až téměř nulová následná režie zajisté velkou výhodou.

6.4 Rozhraní a běh

Rozhraní aplikace je navrženo jako webová stránka, kde pomocí formuláře uživatel nahraje na dočasné úložiště svoji vybranou fotografii, kterou následně implementovaná aplikace zpracuje. Zpracování je podobné procesu tvorby slovníku, sestává z kroků:

- Načtení fotografie z dočasného úložiště.
- Nalezení význačných bodů ve fotografii.
- Extrakce příznaků z bodů.
- Načtení uloženého slovníku z databáze.
- Vytvoření Bag of Words deskriptoru, charakterizujícího obrázek.
- Vložení prostorových dat a úprava formátu deskriptoru.
- Dotaz do databáze ohledně nejlepší shody.

Z databáze je vrácena stále poměrně velká množina fotografií, kterou aplikace dále redukuje s využitím homografie a porovnávání blízkosti význačných bodů.



Obrázek 6.3: Schéma vyhledání fotografie.

Kapitola 7

Implementace

V této kapitole se budu věnovat detailům samotné implementace aplikace a všem aspektům s tím souvisejícím. Rozeberu zde vytyčené cíle, ke kterým by tato práce měla dospět a postup, kterým těchto cílů bylo dosaženo, případně ve které části projektu vyvstaly nějaké problémy.

Aplikace byla vyvíjena iterativně, v každé verzi byla vždy přidávána, resp. upravována funkčnost podle získaných poznatků a na základě provedených experimentů.

Při implementaci podobných aplikací je důležité rozložení algoritmů mezi časovou a prostorovou složitost. Ideálně by celá aplikace ukládala a čerpala data pouze z RAM, tím bychom dosáhli výborné rychlosti. V prvních iteracích jsem testoval tuto možnost a zvládal jsem pracovat jen řádově se stovkami fotografií (nutno říci, že bez větších optimalizací). Tato možnost je tedy vhodná jen pro velmi omezený počet dat. V dalších iteracích jsem vyzkoušel ukládání dat do xml souboru na disk. Zde je již prostor pro takřka neomezené množství dat, nicméně nevýhodou je v podstatě sekvenční povaha přístupu k takovým záznamům. Tímto přístupem tedy získáme hodně paměťového prostoru, ale ztratíme na rychlosti vyhledávání. Je tedy zřejmé, že pro aplikaci na úrovni rychlého vyhledávání v databázi desítek až stovek tisíců fotografií bylo nutné zvolit jiný přístup, a to využití relační databáze. Tato reprezentace umožňuje značné množství optimalizací z hlediska přístupu k záznamům, kterému se budu věnovat dále.

Serverová část aplikace je implementována v jazyce C++ a využívá databáze PostgreSQL s rozšířením Smlar. Klientská část je implementována jako PHP rozhraní v podobě webové stránky, kterému klient bude poskytovat dotaz v podobě vlastní fotografie a výstupem získá množinu nejpodobnějších fotografií z databáze na serveru.

Serverová aplikace je spouštěna v jednom ze dvou režimů podle požadované funkce.

7.1 Režim učení

Aplikace očekává cestu ke galerii fotografií, které bude následně zpracovávat. Zpracovávání probíhá tak, že aplikace nejprve vloží do databáze jednotlivé fotografie na základě jejich názvu a cesty a nad každou fotografií spočítá pole deskriptorů, které si drží v paměti. Toto pole deskriptorů následně použije při tvorbě slovníku vizuálních slov pomocí k-means shlukování.

(a) Detekce SIFT. (b) Detekce SURF.

Obrázek 7.1: Srovnání detekce význačných bodů SIFT a SURF.

Algoritmus	Počet nalezených bodů
SIFT	8741
SURF	2900

Tabulka 7.1: Počty nalezených bodů

Extrakce příznaků

Pro extrakci keypointů a deskriptorů využívám algoritmus SURF, který se běžně používá ve výpočtech prováděných v reálném čase (zpracování videa). Chtěl jsem využít teoreticky vyšší rychlost, ale především mi vyhovoval menší počet detekovaných keypointů. Vzhledem ke škálovatelnosti řešení je možné využít pomalejší a paměťově náročnější řešení, pokud budou k dispozici dostatečné systémové zdroje, nicméně já byl v tomto ohledu poměrně omezený (implementace byla prováděna na stroji o jednom jádru CPU a 2GB RAM). Srovnání detekce je možné vidět na obrázku 7.1.

Je vidět, že jsou detekovány keypointy ve stejných místech, jen SIFT k tomu přidává velké množství "nadbytečných"bodů, které jsou už v prostorové informaci obsaženy. Počet detekovaných bodů u algoritmu SIFT je přibližně 3x větší, což je vzhledem k důrazu na velkou databázi nepřijatelné (detaily v tabulce 7.1). Algoritmus SURF navíc pracuje s vektory deskriptorů o délce 64 hodnot, oproti deskriptoru SIFT s délkou 128, což přináší další zmenšení objemu dat nutně uloženého v paměti.

Redukce počtu příznaků

Počet bodů byl volbou vhodného algoritmu značně redukován, nicméně pro hardwarové omezení je toto řešení stále nedostačující. K zaplnění dostupné paměti (na referenčním

počítači) by došlo již při přibližně 30000 fotografiích. Cílem je dosažení hodnoty 100000 fotografií a poté slovník dle potřeby segmentovat. Proto jsem přistoupil k další redukci počtu bodů pomocí adaptivního algoritmu, který vstupní bitmapu rozdělí na buňky, ve kterých poté z množiny nalezených význačných bodů vybírá ty nejsilnější. Algoritmus se pokouší o rovnoměrné rozložení bodů a zároveň jejich počet omezuje, takže se dá lépe odhadnout paměťová náročnost. U vzorového obrázku byl počet bodů redukován z 2900 na 466 nejsilnějších bodů a tím dosažena požadovaná paměťová náročnost procesu přípravy dat ke k-means shlukování. Výsledek je znázorněn na obrázku 7.2.

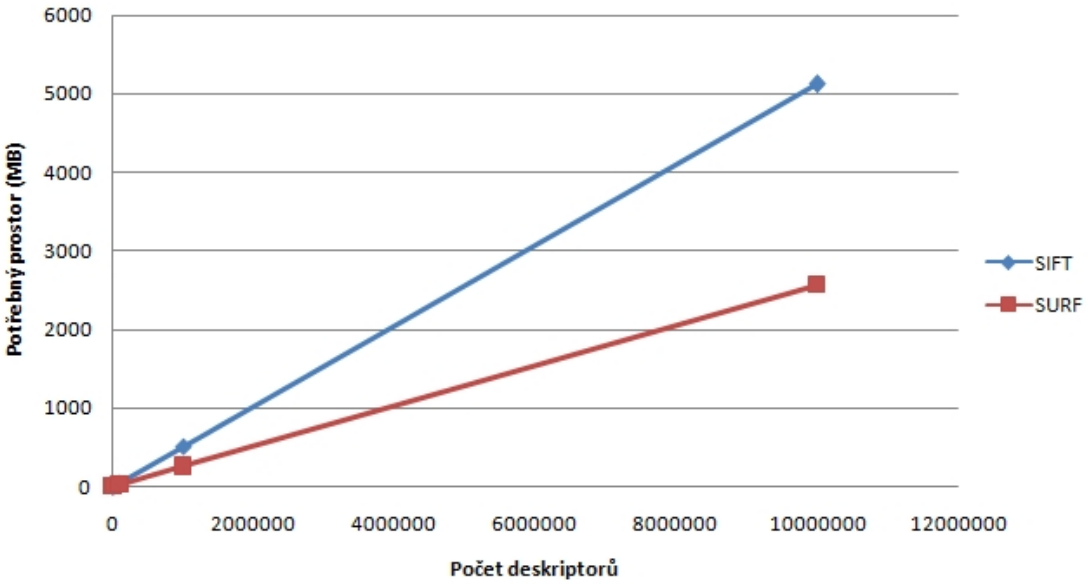
Obrázek 7.2: Redukce pomocí adaptivního algoritmu.

Tento postup samozřejmě snižuje výslednou přesnost vyhledávání, dle mých měření došlo ke ztrátě přibližně 30% přesnosti (hledaná fotografie byla v základní neoptimalizované metrice o 30% blíže následujícímu nerelevantnímu obrázku), vyhledání ale bylo stále úspěšné a redukce měla výrazný vliv na rychlost následného k-means shlukování (při 2000 fotografiích došlo k zrychlení o 80% z minuty na několik sekund.

Pro další urychlení následného shlukování jsem před samotným vyhledáním význačných bodů byl nucen provést zmenšení vstupního obrázku, protože i redukovaných 500 bodů na obrázek dává obrovské množství dat při databázi 100000 fotografií. Redukce na maximální šířku 300px redukovala průměrný počet bodů na obrázek na 50, čímž jsme se již dostali do prostorově přijatelných hodnot.

Algoritmus	Průměrný počet bodů/obrázek	Datová náročnost
SIFT	6500	332800 MB
SURF	2300	58880 MB
Adaptive SURF	450	11520 MB
Downsampled Adaptive SURF	50	1280 MB

Tabulka 7.2: Metriky neshlukovaných dat pro databázi 100000 obrázků.



Obrázek 7.3: Srovnání prostorové náročnosti SIFT a SURF.

Důvodem pro snahu o co největší redukci počtu zpracovávaných bodů je fakt, že před shlukováním potřebujeme mít všechny deskriptory načtené v paměti a nad celou touto množinou teprve provádíme samotné shlukování. Alternativou může být tvorba nového slovníků pro následující část vstupních dat vždy, když vyčerpáme paměť, nicméně podle pokusů není problém vést slovníky pro přibližně 100000 – 200000 obrázků bez velké ztráty na přesnosti následného vyhledávání. Podrobnosti ohledně prostorové náročnosti jsou v tabulce 7.2. Jak je vidět, tak bez drastické redukce průměrného počtu bodů v každé fotografii není tvorba slovníku vizuálních slov nad tak velkou množinou fotografií vůbec reálná.

Tvorba slovníku

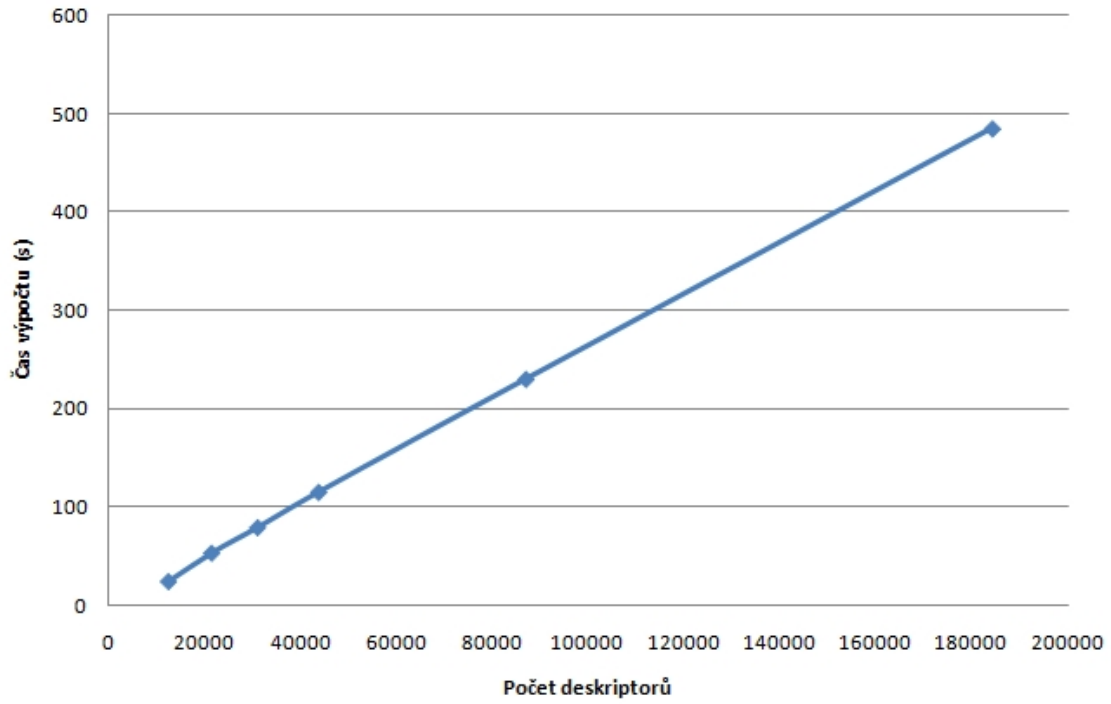
Předchozí kroky vedly k vytvoření matice deskriptorů všech význačných bodů nalezených v každém obrázku z databáze. Nad touto množinou dat je následně provedeno k-means shlukování, jehož produktem je slovník 100000 vizuálních slov.

Tento slovník (Bag of Words) je důležitou položkou, která je následně uložena do databáze a jejíž pomocí je aktualizována samotná databáze fotografií, kde je ke každé fotografii přiřazen její Bag of Words deskriptor. Proces shlukování je časově velmi náročný, pro 100000 slov nad 5000000 deskriptory ze 100000 obrázků proces shlukování trval 48 hodin (1x 3.8 GHz CPU). Srovnání vlivu počtu deskriptorů na časovou náročnost k-means algoritmu je ilustrováno v tabulce 7.3 a v grafu 7.4. Je vidět, že algoritmus v této implementaci má lineární časovou složitost, což odpovídá časové složitosti, popsané v literatuře. Zdvojnásobení počtu deskriptorů způsobí zdvojnásobení délky běhu algoritmu. Musíme ale pamatovat na fakt, že podobný vliv má i počet slov ve slovníku, takže efektivní zvyšování přesnosti má ve výsledku složitost n^2.

Metrika týkající se přesnosti byla naměřena pomocí dvou obrázků, jednoho pozitivního (v databázi se vyskytuje podobná fotografie) a jednoho negativního (razantně se liší od všech databázových obrázků). Procentuálně je vyjádřen rozdíl C_Δ mezi korelační hodnotou pozitivního a referenčního obrázku C_p a mezi korelační hodnotou negativního a referenčního

Počet deskriptorů	Délka výpočtu k-means	Relativní přesnost
12604	24 s	100%
21592	53 s	89%
31183	79 s	83%
43890	115 s	75%
87087	230 s	62%
184277	485 s	48%

Tabulka 7.3: Srovnání doby výpočtu a přesnosti v závislosti na počtu deskriptorů. Počet slov je konstantních 10000.

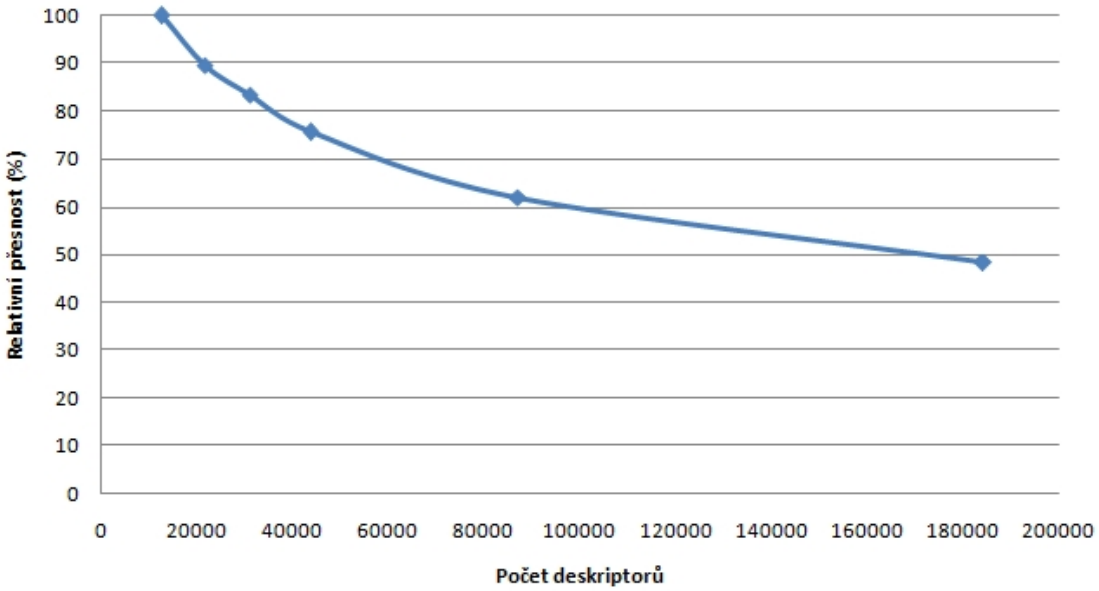


Obrázek 7.4: Srovnání vlivu počtu deskriptorů na délku výpočtu k-means při 10000 slovech.

obrázku C_n (viz rovnice 7.1) Hodnoty $H_r(x)$ a $H_p(x)$ odpovídají patřičným hodnotám buněk jednotlivých deskriptorů.

$$\begin{aligned} C_p &= \frac{\sum_i (H_r(i) - \overline{H_r})(H_p(i) - \overline{H_p})}{\sqrt{\sum_i (H_r(i) - \overline{H_r})^2 \sum_i (H_p(i) - \overline{H_p})^2}} \\ C_n &= \frac{\sum_i (H_r(i) - \overline{H_r})(H_n(i) - \overline{H_n})}{\sqrt{\sum_i (H_r(i) - \overline{H_r})^2 \sum_i (H_n(i) - \overline{H_n})^2}} \\ C_\Delta &= C_p - C_n \end{aligned} \tag{7.1}$$

Čím nižší je hodnota tohoto rozdílu, tím vyšší je riziko, že bude obrázek vyhodnocen nesprávně. Je vidět, že s přibývajícími deskriptory a stále konstantním počtem slov se relativní přesnost snižuje. To je zapříčiněno tím, že čím dál více deskriptorů je přiřazeno do stejných slov, čímž ztrácíme na rozlišení jednotlivých fotografií.



Obrázek 7.5: Srovnání vlivu počtu deskriptorů na přesnost výpočtu k-means při 10000 slovech.

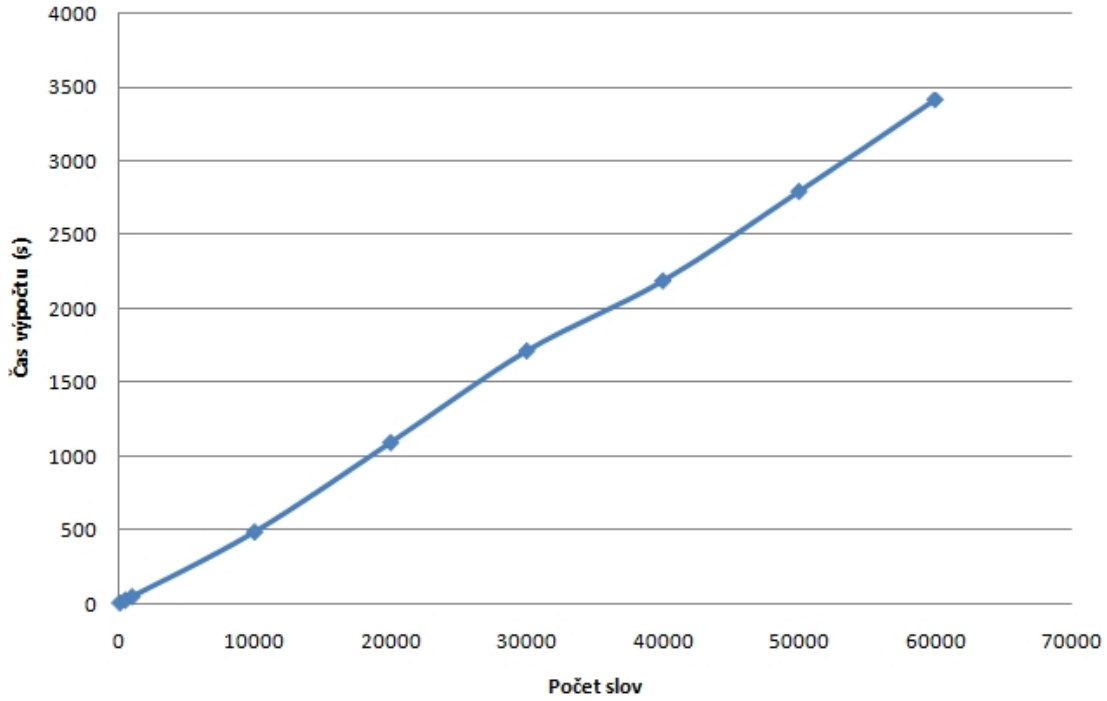
Volba počtu vizuálních slov má vliv na výslednou přesnost vyhledávání. Cílem je, aby ke každému vizuálnímu slovu byl pokud možno přiřazen počet deskriptorů blížící se 1 (resp. 0, pokud slovo v obrázku obsaženo není). Pokud k jednomu slovu připadne příliš mnoho deskriptorů, tak se ochuzujeme o potenciál rozlišení dvou obrázků na základě rozdílných slov. Naopak pokud ke slovu připadne příliš málo deskriptorů, tak se může stát, že dva podobné obrázky budou mít značně odlišné Bag of Words deskriptory. Problém lze shrnout následovně:

- Příliš mnoho vizuálních slov – I drobné rozdíly povedou k velmi rozdílným deskriptorům.
- Příliš málo vizuálních slov – I značně rozdílné obrázky budou vyhodnoceny jako podobné.

Počet vizuálních slov má také vliv na délku výpočtu, ale závislost je podobně jako u deskriptorů lineární, jak je vidět v grafu 7.6 a tabulce 7.4. Jakmile je dokončen slovník

Počet vizuálních slov	Délka výpočtu k-means
10000	485 s
20000	1091 s
30000	1712 s
40000	2186 s
50000	2792 s
60000	3414 s

Tabulka 7.4: Srovnání vlivu počtu vizuálních slov na délku výpočtu k-means při 180000 deskriptorech.

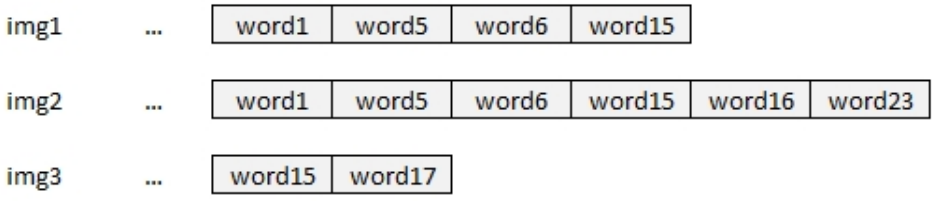


Obrázek 7.6: Srovnání vlivu počtu vizuálních slov na délku výpočtu k-means při 180000 deskriptorech.

vizuálních slov, tak aplikace prochází jednotlivé fotografie v databázi a ke každé vypočítá a přiřadí Bag of Words deskriptor, který je po úpravě uložen do databáze.

Tvorba položky v databázi

Protože při dobře zvoleném počtu vizuálních slov je vektor BoW deskriptoru relativně řídký (a to především u následně hledaných fotografií), tak je do databáze ukládána především informace o samotném výskytu daného vizuálního slova. To vede k značnému ušetření prostoru a rychlosti následného vyhledávání, díky kterému není potřeba porovnávat rozsahy hodnot, ale jen vyhledat samotné hodnoty.



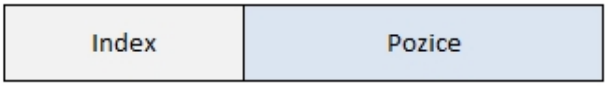
Obrázek 7.7: Ilustrace vektoru příznaků v databázi.

Přidání dalších heuristik ohledně počtu daných příznaků by vylepšilo databázové výsledky za cenu výpočetní náročnosti. V implementaci jsem zvolil variantu velice jednoduchého způsobu porovnávání, který by mohl být vykonán velmi rychle a vrátil by relativně nepřesnou množinu výsledků, která by ale i tak byla dostatečně malá na efektivní aplikaci dalšího zpřesnění na úrovni programu.

Zakomponování prostorových dat

V implementaci jsem testoval různé způsoby vyjádření vzájemné pozice vizuálních slov. Testoval jsem variantu s detekcí shluků detektorem MSER, ale nebyl jsem spokojený se stálostí těchto výsledků. Experimenty vracely různé shluky při různých úrovních rozlišení a bylo složité vytvořit na tomto základě spolehlivý model, pomocí kterého by se daly jednoznačně vyjádřit prostorové závislosti vizuálních slov.

Nakonec jsem využil obdobu Spatial Pyramid Matching pouze s jednou úrovní, na základě které ukládám ke každému slovu jeho polohu v mřížce 3x3. Porovnávání kvůli tomuto má problém s okrajovými sektory při větší transformaci rotace, nicméně u dobře orientovaných fotografií nese dobré výsledky.



Obrázek 7.8: Ilustrace kombinovaného příznaku v databázi.

7.2 Režim vyhledávání

V režimu vyhledávání aplikace očekává jako parametr cestu k fotografii, podle které se bude vyhledávat nejpodobnější množina v databázi. Pro testování je možné spouštět aplikaci přímo, návratovou hodnotou jsou seřazené absolutní cesty k fotografiím na serveru.

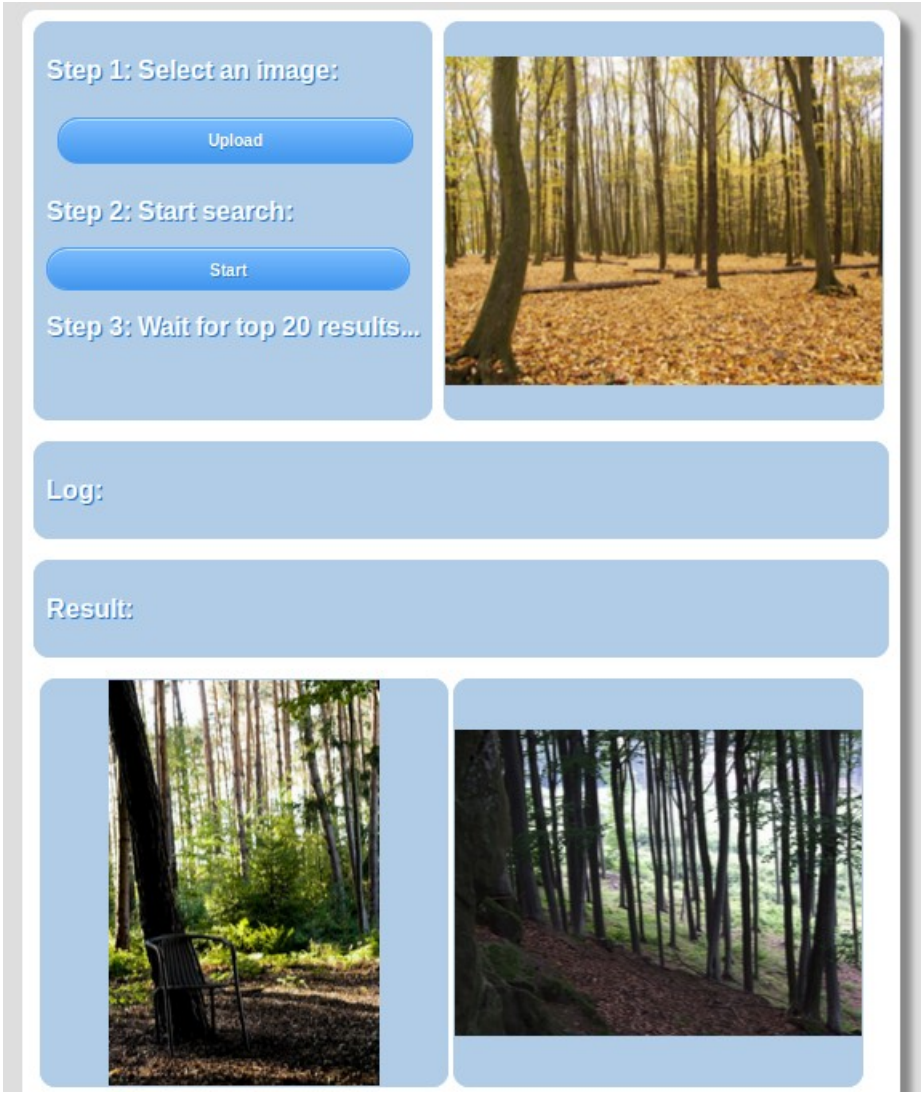
Aplikace je primárně určena pro souběžný běh s webovým serverem, který slouží jako uživatelské rozhraní. Uživatel webovému formuláři poskytne fotografii ze své lokální sbírky a tato fotografie bude nahrána na server do dočasného úložiště. Cesta k této fotografii je následně poskytnuta aplikaci v parametru a je spuštěno vyhledávání. Během vyhledávání jsou extrahovány příznaky ze vstupní fotografie, vytvořen příslušný Bag of Words deskriptor dle dříve zmíněných pravidel a s tímto Bag of Words deskriptorem je vytvořen dotaz do databáze na míru podobnosti.

Délka vyhledávání v databázi je optimalizována využitím invertovaného indexu, nicméně narůstající počet fotografií má měřitelný dopad na délku hledání. V tabulce 8.3 v experimentech je znázorněna doba vyhledávání podle počtu záznamů. K rozšíření databáze došlo duplikováním již uložených záznamů, takže na zdánlivou rychlost výpočtu může mít vliv efektivita invertovaného indexu a při stovkách tisíc opravdu unikátních Bag of Words deskriptorů by mohl být výsledný čas horší, nicméně pro odhad časové složitosti a škálovatelnosti řešení toto měření postačuje. Jak již bylo psáno v kapitole věnující se invertovanému indexu, tak vyhledávání v tomto indexu je rychlé, ale vkládání dat představuje problém, který je znázorněn v třetím sloupci tabulky. Zapsaný čas označuje počet minut, potřebných k navýšení počtu záznamů z předchozí na současnou hodnotu. Jak je vidět, tak i když objem přidaných dat zůstává konstantní, tak se rychlost přidávání dat značně zpomaluje kvůli přebudovávání invertovaného indexu.

7.3 Webové rozhraní

Jednoduché webové rozhraní je implementováno v jazyce PHP a poskytuje uživateli formulář k nahrání fotografie, která je následně uložena do dočasného úložiště na serveru. PHP

rozhraní poté přímo volá serverový proces, kterému parametrem předává cestu ke zvolenému souboru a čeká na návrat adresářových cest k nalezeným fotografiím, které následně zobrazí na stránce. Část webového rozhraní je vidět na obrázku 7.9.



Obrázek 7.9: Webové rozhraní pro vyhledávání fotografií.

7.4 Použité knihovny

- OpenCV – Jádro aplikace. Obsahuje třídy a metody pro práci s fotografiemi.
- pqxx – Knihovna sloužící k propojení aplikace s PostgreSQL databází.
- Smlar – Rozšíření databáze PostgreSQL o funkce porovnání dvou polí hodnot a vyhledávání polí hodnot pomocí GiST a GIN indexů.

7.5 Další vývoj

Tato práce má velký prostor pro další rozvoj. Prvním faktorem by bylo další rozšiřování databáze (blížící se k milionu fotografií) a zakomponování prvku automatické segmentace slovníku. S tímto rozšířením by se poté daly provádět další experimenty ohledně rychlosti vyhledávání a ověření dříve naměřené rychlosti a hypotézy ohledně efektivity invertovaného indexu na duplicitních datech.

Druhý směr dalšího vývoje se týká samotného vyhledávání, kde je velký prostor pro další optimalizaci a rozšíření prostorové informace, testování nových způsobů vyhledávání význačných bodů a jejich deskripce a tvorby Bag of Words deskriptoru.

V práci bylo zmíněno několik různých algoritmů, ze kterých byly nakonec ve finální implementaci testovány 2 (SIFT a SURF), takže je zde prostor pro změnu implementace a vystavění aplikace nad jiným způsobem popisu bodů (například binární řetězec).

Kapitola 8

Experimenty

V této kapitole je popsáno několik experimentů s výslednou aplikací, které se týkají rychlosti a přesnosti vyhledávání. První část se věnuje časové složitosti dotazů do databáze a následně je shrnuta přesnost vyhledávání s využitím pouze BoW deskriptoru a přesnost vyhledávání následným řazením na základě vzdálenosti deskriptorů a homografie.

8.1 Údaje o experimentálním modelu

Pro lepší představu o naměřených výsledcích je v tabulce 8.1 uveden souhrn údajů, které charakterizují implementovanou aplikaci pro vyhledávání fotografií.

Frekvence procesoru	3.7 GHz
Počet fotografií	87552
Potřebný prostor na disku	7.62 GB
Potřebný prostor v databázi	1.1 GB
Počet vizuálních slov	100000
Velikost kandidátní množiny	600
Počet výstupních fotografií	20

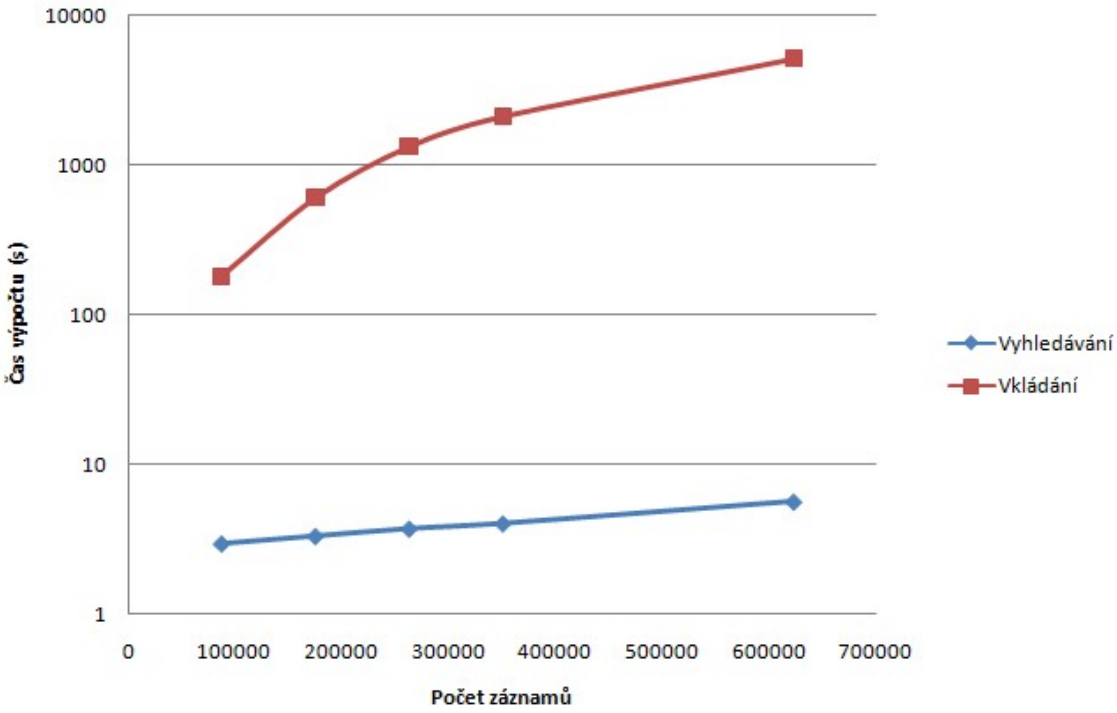
Tabulka 8.1: Parametry testované aplikace.

8.2 Experimenty s databází

V tabulce 8.1 a grafu 8.1 je vidět srovnání nárustu náročnosti výpočtu na základě počtu záznamů v databázi. K rozšíření počtu záznamů došlo duplikováním již existujících záznamů, takže je složitost potenciálně menší díky optimálnosti GIN indexu, nicméně vyhledávací složitost dle měření vychází se složitostí $O(n)$. Složitost vkládání dat je oproti tomu $O(n^2)$, což odpovídá dokumentovaným vlastnostem GIN indexu.

Počet záznamů	Délka vyhledávání	Potřebný čas k vložení záznamů
87552	2,9 s	3 minuty
175104	3,3 s	10 minut
262656	3,69 s	22 minut
350208	4,01 s	35 minut
621746	5,622 s	85 minut

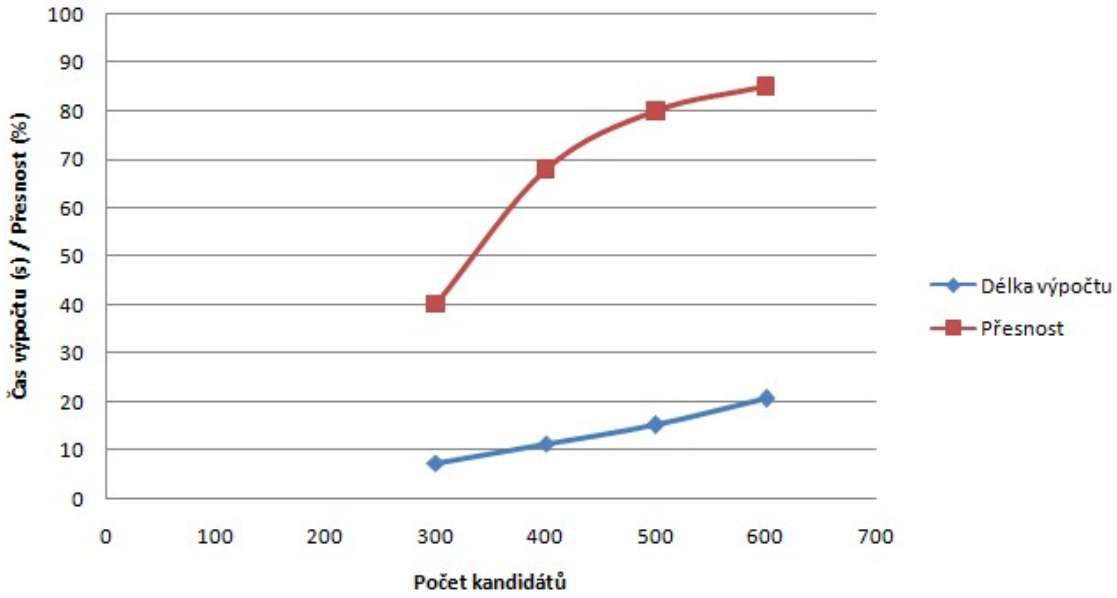
Tabulka 8.2: Srovnání délky vyhledávání podle počtu záznamů.



Obrázek 8.1: Graf srovnání délky vyhledávání a vkládání podle počtu záznamů.

Počet kandidátů	Délka vyhledávání	Procento relevantních fotografií
600	20,2 s	85%
500	15,192 s	80%
400	11,22 s	65%
300	7,433 s	40%

Tabulka 8.3: Srovnání vlivu počtu kandidátů na rychlost a přesnost.



Obrázek 8.2: Graf vlivu počtu kandidátů na rychlost a přesnost.

8.3 Experimenty s vyhledáváním

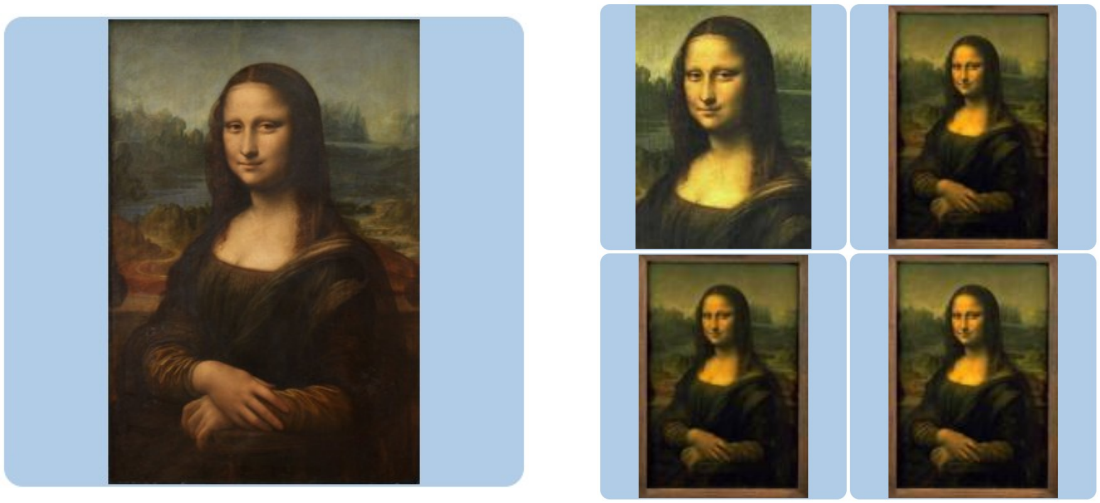
V této části jsou srovnány výsledky vyhledávání při použití dalšího zpřesnění a seřazení kandidátní množiny a výsledky bez tohoto řazení (tedy 20 nejlepších výsledků jen na základě podobnosti BoW deskriptoru). Vliv tohoto zpřesnění je vidět hlavně na vzorku 1 na obrázcích 8.3 a 8.4, kde bez zpřesnění není množina výsledků velmi kvalitní. Druhý vzorek na obrázcích 8.5 a 8.6 díky velké podobnosti s databázovými fotografiemi již nesl kvalitní výsledky i bez řazení.

(a) Hledaná fotografie. (b) Výstup (4 nejlepší fotografie).

Obrázek 8.3: Vzorek 1: Výstup po redukci kandidátních fotografií.

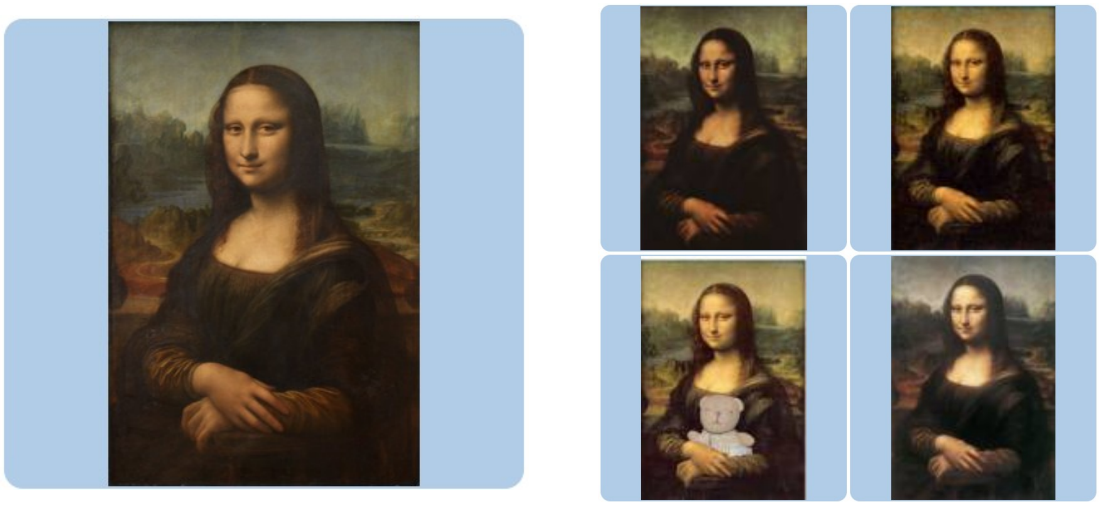
(a) Hledaná fotografie. (b) Výstup (4 nejlepší fotografie).

Obrázek 8.4: Vzorek 1: Výstup bez redukce kandidátních fotografií.



(a) Hledaná fotografie. (b) Výstup (4 nejlepší fotografie).

Obrázek 8.5: Vzorek 2: Výstup po redukci kandidátních fotografií.



(a) Hledaná fotografie. (b) Výstup (4 nejlepší fotografie).

Obrázek 8.6: Vzorek 2: Výstup bez redukce kandidátních fotografií.

Kapitola 9

Závěr

Jak je patrné z experimentů, tak cíl práce byl splněn. Výsledná aplikace dokáže vyhledat podobnou fotografii během 20 sekund z množiny 87552 fotografií. Vyhledávání vyžaduje další upřesnění, protože na značně členitých fotografiích není výsledek příliš kvalitní. Tato následná korekce je na algoritmu časově nejnáročnější.

Aplikace může sloužit jako základ pro další experimentování, případně může být rovnou nasazena na nějakém serveru. Další vývoj může být věnován testování různých zmíněných algoritmů, především týkajících se binární reprezentace deskriptoru a efektivních bitových operací s těmito deskriptory. Další možností je již již zmíněné rozšíření prostorových informací například na detekci pozadí fotografie a přiřazení vizuálních slov do kategorie popředí a pozadí.

Samotná práce obsahuje dále použitelné informace ohledně jednotlivých algoritmů, jejichž popis může sloužit jako podklad pro další studium zpracování fotografií a detekce význačných bodů.

Literatura

[1] Bay, H.; Ess, A.; Tuytelaars, T.; aj.: Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, ročník 110, č. 3, Červen 2008: s. 346–359, ISSN 1077-3142.

[2] Calonder, M.; Lepetit, V.; Strecha, C.; aj.: BRIEF: Binary Robust Independent Elementary Features. In *Computer Vision - ECCV 2010*, *Lecture Notes in Computer Science*, ročník 6314, editace K. Daniilidis; P. Maragos; N. Paragios, Springer Berlin Heidelberg, 2010, ISBN 978-3-642-15560-4, s. 778–792, doi:10.1007/978-3-642-15561-1"˙56.

[3] Fischler, M. A.; Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, ročník 24, č. 6, Červen 1981: s. 381–395, ISSN 0001-0782.

[4] Grauman, K.; Leibe, B.: *Visual Object Recognition*. Synthesis lectures on artificial intelligence and machine learning, Morgan & Claypool Publishers, 2011, ISBN 9781598299687.

[5] Guliato, D.; Melo, E.; Rangayyan, R.; aj.: POSTGRESQL-IE: An Image-handling Extension for PostgreSQL. *Journal of Digital Imaging*, ročník 22, č. 2, 2009: s. 149–165, ISSN 0897-1889, doi:10.1007/s10278-007-9097-5.

[6] Jia, W.; Zhang, H.; He, X.; aj.: A Comparison on Histogram Based Image Matching Methods. In *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, Nov 2006, s. 97–97, doi:10.1109/AVSS.2006.5.

[7] Lazebnik, S.; Schmid, C.; Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, Washington, DC, USA: IEEE Computer Society, 2006, ISBN 0-7695-2597-0, s. 2169–2178.

[8] Leutenegger, S.; Chli, M.; Siegwart, R.: BRISK: Binary Robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference*, 2011, ISSN 1550-5499, s. 2548–2555, doi:10.1109/ICCV.2011.6126542.

[9] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, ročník 60, č. 2, Listopad 2004: s. 91–110, ISSN 0920-5691.

[10] Lux, M.; Chatzichristofis, S. A.: Lire: Lucene Image Retrieval: An Extensible Java CBIR Library. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-303-7, s. 1085–1088.

[11] Muja, M.; Lowe, D. G.: Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, 2009, s. 331–340.

[12] OpenCV: ORB [online]. `http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_orb/py_orb.html`.

[13] Oyallon, E.; Rabin, J.: An analysis and implementation of the SURF method, and its comparison to SIFT [online]. `http://www.ipol.im/pub/pre/69/preprint.pdf`.

[14] pixHawk.ethz.ch: Fast natural features [online]. `https://pixhawk.ethz.ch/software/computer_vision/surf_features`.

[15] Prince, S. J. D.: *Computer Vision: Models, Learning, and Inference*. New York, NY, USA: Cambridge University Press, první vydání, 2012, ISBN 1107011795, 9781107011793.

[16] Rosten, E.; Porter, R.; Drummond, T.: Faster and Better: A Machine Learning Approach to Corner Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 32, č. 1, Jan 2010: s. 105–119, ISSN 0162-8828, doi:10.1109/TPAMI.2008.275.

[17] Rublee, E.; Rabaud, V.; Konolige, K.; aj.: ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, Washington, DC, USA: IEEE Computer Society, 2011, ISBN 978-1-4577-1101-5, s. 2564–2571.

[18] Rubner, Y.; Tomasi, C.: Texture-based image retrieval without segmentation. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, ročník 2, 1999, s. 1018–1024 vol.2, doi:10.1109/ICCV.1999.790380.

[19] Sutherland, D.; Carlson, R.: Evaluating Multidimensional Histograms in PostgreSQL [online]. `http://www.cs.cmu.edu/ rcarlson/docs/RyanCarlson_databases.pdf`.

[20] VLFeat.org: Scale Invariant Feature Transform (SIFT) [online]. `http://www.vlfeat.org/api/sift.html`.

[21] Wang, M.; Yang, K.; Hua, X.-S.; aj.: Towards a Relevant and Diverse Search of Social Images. *Multimedia, IEEE Transactions on*, ročník 12, č. 8, Dec 2010: s. 829–842, ISSN 1520-9210, doi:10.1109/TMM.2010.2055045.

[22] Zuliani, M.; Kenney, C. S.; Manjunath, B. S.: The multiRANSAC algorithm and its application to detect planar homographies. In *Image Processing*, ročník 3, IEEE International Conference, 2005, ISBN 0-7803-9134-9, s. 153–156.