**České vysoké učení technické v Praze**
**Fakulta elektrotechnická**

**Katedra kybernetiky**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

**Student:** Bc. Ondřej Kopřiva

**Studijní program:** Otevřená informatika (magisterský)

**Obor:** Počítačové vidění a digitální obraz

**Název tématu:** Fotoaparát s historickou pamětí

## Pokyny pro vypracování:

Cílem práce je vytvořit prototyp systému, který bude podporovat rozpoznání příslušnosti nové fotografie objektu k jednomu z již existujících souborů fotografií architektonických objektů. Obrazy téhož objektu v databázi mohou pocházet z různých historických období. Vstupní obraz může být doplněn GPS geolokací, která rozpoznání usnadní. Historická a částečně i moderní vstupní data budou dodána.

1. Seznamte se s metodami vhodnými pro popis obrazů architektonických scén, pro párování mezi obrazy a pro rozpoznání příslušnosti obrazu k souboru fotografií téhož objektu.
2. Zjistěte dostupnost implementací jednotlivých metod a rozhodněte, které převezmete do svého systému.
3. S použitím dostupných metod, případně s vlastními implementacemi vytvořte podporu pro následující automatické procesy:
   a) popis souboru snímků jedné scény na základě *bag of visual words* nebo podobném.
   b) rozpoznání příslušnosti nového obrazu k souboru fotografií téhož objektu a vyhledání nejpodobnějšího pohledu.
   c) přibližné slícování moderní a historické fotografie téhož objektu, fotografovaných z podobného místa.
4. Demonstrujte vhodnost vytvořené podpory pro následující aplikaci: Turista na své procházce Prahou vyfotografuje historický objekt a následně se v jeho mobilním zařízení zobrazí historické fotografie téhož objektu z podobného místa.
5. Kriticky zhodnoťte dosažené výsledky.

## Seznam odborné literatury:

[1] H. Bay, T. Tuytelaars, L. van Gool: SURF: Speeded up Robust Features. In Proceedings European Conference on Computer Vision, 2006.
[2] J. Matas, Š. Obdržálek, O. Chum: Local Affine Frames for Wide-Baseline Stereo. In Proceedings of International Conference on Pattern Recognition, 2002.
[3] J. Sivic and A. Zisserman: Video Google: A Text Retrieval Approach to Object Matching in Videos. In Proceedings IEEE International Conference on Computer Vision, 2003.

**Vedoucí diplomové práce:** Mgr. Roman Sejkot

**Platnost zadání:** do konce zimního semestru 2014/2015

doc. Dr. Ing. Jan Kybic
**vedoucí katedry**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

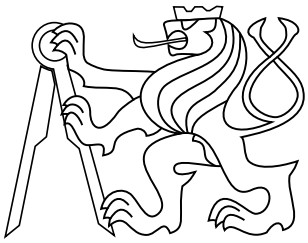
prof. Ing. Pavel Ripka, CSc.
**děkan**

V Praze dne 16. 9. 2013

diplomová práce

# Fotoaparát s historickou pamětí

*Bc. Ondřej Kopřiva*



Leden 2014

Vedoucí: Mgr. Roman Sejkot

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

# Poděkování

Chtěl bych poděkovat svému vedoucímu Mgr. Romanu Sejkotovi a také Doc. Ing. Radimu Šárovi, Ph.D. za pomoc a cenné rady při zpracování této práce. Dále pak Mgr. Pavlu Scheuflerovi za poskytnutí historických fotografií a Ing. Danielu Večerkovi za serverovou část. A také rodině a přátelům za podporu a trpělivost.

## Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 2.1.2014 .....

.....

Podpis autora práce

# Abstrakt

Tato práce se zabývá vývojem prototypu mobilního systému, který umožňuje uživateli vyfotografovat zajímavou historickou budovu fotoaparátem ve svém mobilním zařízením a místo pořízené fotografie si zobrazit odpovídající historickou.

Systém se skládá ze dvou částí. Klíčovou částí je server, který zpracuje fotografie, vyhledá nejpodobnější budovy z databáze současných fotografií, vybere odpovídající historický obrázek a pokud je to možné, upraví ho do úhlu pohledu, ze kterého byla pořízena uživatelská fotografie. Vyhledávání fotografií je založeno na metodě *bag of words* a následném geometrickém ověření výsledků. Druhou součástí je jednoduchá mobilní aplikace pro zařízení s operačním systémem Android, která umožňuje uživateli pořídit fotografie a prostřednictvím serveru k nim zobrazit odpovídající historické.

Systém byl otestován na sadě 200 fotografií 13 pražských historických a moderních budov. Průměrná úspěšnost rozpoznání správné budovy je 80%.

## Klíčová slova

Počítačové vidění, Historické fotografie, Mobilní aplikace, Vyhledávání obrázků

# Abstrakt

This work deals with the development of a prototype mobile service that allows a user to photograph a historical object of interest by his mobile device's camera and display an old historical image of the object instead of the current one.

The system consists of two major parts: The core one is a server that processes the input image, finds the most similar image of the building in a database of modern images, fetches the corresponding historical image, and, if possible, transforms the historical image to the user viewpoint. The image matching is based on the Bag of Words method followed by geometric correspondence verification. The second part of the system is an application for an Android mobile device, that allows user to take photographs, communicate with the server, and display the result.

The system has been tested on a suite of 202 images of 13 historical and modern buildings in Prague. The average success rate of recognizing the correct building is 80%.



## Keywords

Computer vision, Mobile Application, Historical Photography, Image retrieval, Mobile Imaging

# Obsah

**1. Úvod** **1**

**2. Návrh řešení** **5**
2.1. Zpracování obrazu . . . . . 6
2.1.1. Detekce bodů zájmu . . . . . 7
2.1.2. Měřítko . . . . . 9
2.1.3. Afinní normalizace . . . . . 10
2.1.4. Deskriptory . . . . . 11
2.2. Vytvoření databáze . . . . . 11
2.2.1. Datová sada . . . . . 12
2.2.2. Snížení dimensionality . . . . . 12
2.2.3. Shlukování . . . . . 12
2.2.4. Pytel slov . . . . . 16
2.2.5. Vylepšení databáze . . . . . 16
2.3. Zpracování dotazu . . . . . 18
2.3.1. Postup vyhledávání . . . . . 18
2.3.2. Rozšířený dotaz . . . . . 21

**3. Implementace** **23**
3.1. Mobilní aplikace – Klient . . . . . 24
3.1.1. Grafické uživatelské rozhraní . . . . . 25
3.1.2. Informace o poloze . . . . . 28
3.1.3. Komunikace . . . . . 28
3.2. Server . . . . . 30
3.2.1. Rozhraní . . . . . 30
3.2.2. Zpracování obrázku . . . . . 30
3.2.3. Komunikační rozhraní . . . . . 33

**4. Experimenty** **35**
4.1. Testovací data . . . . . 35
4.2. Hodnocení výsledků . . . . . 35
4.3. Výsledky . . . . . 37

**5. Závěr** **41**

**Reference** **43**

**Přílohy**

**A. Obsah přiloženého CD** **47**

# Seznam obrázků


1. Pražské dominanty v historii . . . . . 2
2. Příklad srovnání fotografií . . . . . 3

3. Příklad významné změny stavby . . . . . 5
4. Příklad bodů zájmu nalezených metodou lokálních maxim Hessiánu 8
5. Příklad bodů zájmu nalezených Harrisovým detektorem rohů . . . 9
6. Objekty v databázi . . . . . 13
7. Různé pohledy na jeden objekt . . . . . 14
8. Vliv počtu použitých dimenzí PCA . . . . . 15
9. Příklady popisů obrázků metodou pytel slov. . . . . 17
10. Předběžné korespondence . . . . . 20
11. Ověřené korespondence . . . . . 21

12. Návrh systému . . . . . 23
13. Vzhled uživatelského rozhraní . . . . . 26
14. Uživatelské rozhraní okna s nastaveními. . . . . 27

# 1. Úvod

Touha graficky zachytit podobu věcí, na kterých nám záleží pro budoucí generace, provází lidstvo již desítky tisíc let, od dob kdy naši předci stvořili první jeskyní malby. Lidé očividně rádi srovnávají staré s novým, a proto máme celá oddělení knihoven a knihkupectví plná publikací srovnávajících staré a nové podoby všeho možného a zejména historických památek [1], [2], obr. 2.

Lidé vnímají významné historické stavby jako něco stálého a neměnného. Jako ozvěnu dávných časů, která zde stojí v nezměněné podobě již stovky či tisíce let. Proto by je velice překvapilo, kdyby zjistili, jak moc se jejich město proměnilo jenom v průběhu posledních sto let. A to včetně těchto domněle nedotknutelných památek (obr. 1).

Člověk je tvor od přírody zvědavý a historie je pro mnoho lidí fascinující téma skýtající nekonečné množství zajímavostí a tajemství, která možná nikdy nedokážeme odhalit. Zároveň většina lidí ráda fotografuje a díky vývoji moderní techniky má u sebe stále více lidí mobilní telefony se stále kvalitnějšími fotoaparáty vždy připravenými zachytit cokoliv, co uživatele zaujme. A s rozvojem rychlých datových přenosů v mobilních zařízeních se otevírají nepřeberné možnosti jejich využití.

Ohromný nárůst ekonomicky dostupného výkonu počítačů a s ním spojený dramatický rozvoj informačních technologií v posledních desetiletích přinesl obrovské možnosti. Jednou z oblastí, prožívajících překotný vývoj, je výzkum počítačového vidění. Vidění tak jak ho chápe člověk je nesmírně komplikovaný problém, protože zahrnuje vyšší pochopení scény a souvislostí mezi objekty a jevy, jež zachycuje, získané z předchozích zkušeností, kterého počítače zatím nejsou schopny.

Počítačové vidění tedy zatím řeší jednodušší a úzce specifikované problémy. Příkladem může být zpracování dat v medicíně. Moderní zobrazovací přístroje produkují závratná množství obrazových dat a metody počítačového vidění mohou lékaři pomoci při hledání anomálií. Významné jsou také úzce specializované aplikace v průmyslu, např. na kontrolu kvality. Stále rozšířenější jsou v současnosti všemožné více či méně autonomní stroje. Od parkovacích asistentů a systému prevence kolizí u luxusních automobilů přes bezpilotní letouny až po sondy na cizích planetách. Obzvláště v tomto případě je důležité, když sonda umí řešit některé problémy, jako nalezení cesty, sama protože řídící centrum je nejméně několik hodin daleko.

Počítače mají ovšem své přednosti v rychlosti a přesnosti, či možnosti vidět v částech spektra, které lidské oko nevnímá. Proto se jim v některých specializovaných úlohách počítačového vidění nemůže člověk rovnat.

Jednou z těchto oblastí jsou metody pro vyhledávání obrázků stejných objektů v databázi obrázků. Tyto metody sice neumožňují rozpoznat obrázky tak přesně a podrobně jako to dokáží lidé, ale zato jich dokáží rozpoznat rychle obrovské

**Obrázek 1.** Příklad historických podob několika pražských dominant. Nahoře předchůdce mostu Legií, most císaře Františka I. s Pražským hradem v pozadí (Neznámý autor, kolem 1894)a jezírko před dnešním Hlavním nádražím (Karel Bellman, kolem 1893), dole pak Chrám sv. Víta (František Krátký, 1891-1892) a Karlovo náměstí (František Fridrich, kolem 1866). Převzato z [3]

množství. Příkladem takové technologie je aplikace Google Goggles. Teoreticky umožňuje vyhledávání čehokoliv a čtení textů z fotografie pořízené z mobilního telefonu. Aplikace sice umožňuje např rozpoznávání QR kódů a do jisté míry i čtení textů, ale problémem u fotografií je, že aplikace si klade za cíl vyhledávat bez omezení cokoliv a má tedy tak obrovský záběr, že málokdy najde skutečně odpovídající obrázky.

Tyto moderní technologie nám tak umožňují podívat se na starodávnou myšlenku z nového úhlu, který snad více zaujme uživatele a vtáhne ho do historie více než listování obrázkovou knihou v teple domova.

Tento projekt spojuje tyto prvky v interaktivní možnost ukojit naši zvědavost a podívat se do historie. I když pouze prostřednictvím obrázků. Cílem projektu je vytvořit aplikaci, která umožní uživateli vyfotografovat si historickou stavbu, která ho zaujala a téměř okamžitě mu ukáže vývoj její podoby v průběhu věků. Navíc bychom mu mohli nabídnout stručný text o historii objektu, zajímavosti a tak dále.

Vezměme si jako příklad Chrám Svatého Víta. Pravděpodobně všichni jsme

**(a)** Emauzy, 2013 **(b)** Emauzy, 1933, J. Bruner-Dvořák

**Obrázek 2.** Příklad srovnání aktuální a historické fotografie. Historická fotografie převzata z [3]

se učili ve škole, že tuto dominantu Pražského hradu nechal vybudovat již Karel IV. a většina lidí je tak přesvědčena, že Karel IV. se díval na chrám ve stejné podobě jako my nyní. Málokdo však ví, že jeho současná podoba vznikla až na přelomu 19. a 20. století. Naštěstí existují fotografie z doby před přestavbou (obr. 1).

Neznalého turistu tak zaujme jiný a často nečekaný pohled na zajímavou stavbu, která se mu zalíbila. Možná ještě zajímavější však bude výsledek pro uživatele, který je zde doma. Okolo stavby nesčetněkrát prošel a je přesvědčen, že ji zná jako své boty, že vždycky vypadala tak, jak ji zná dnes a navždy tak vypadat bude. Bude pak dost překvapen, když uvidí, jak dramaticky se podoba změnila a třeba se v něm probudí zvědavost, a začne se zajímat o historii svého okolí. Už nebude pouze znuděně procházet kolem tisíckrát viděných budov, ale začne přemýšlet o tom, jak to vše vypadalo dřív.

Aplikace se však nemusí zaměřovat výhradně na historické stavby. Zajímavé mohou být i moderní budovy nebo spíše co na jejich místě bylo před nimi.

Zobrazit vývoj v průběhu věku je samozřejmě utopický cíl, protože historických vyobrazení kýženého objektu je pouze omezené množství, roztroušené mezi sběrateli, archivy a muzei, nebo dokonce žádné neexistují. Proto se musíme spokojit se zobrazením alespoň nějakých dostupných vyobrazení z historie některých význačných památek.

Pokud chceme dosáhnout spolehlivějších výsledků než obecné vyhledávání Google Goggles musíme úlohu poněkud upřesnit. V tomto případě je problémem rozpoznávání staveb. Nicméně ne jakýchkoliv, ale pouze vybraných zajímavých staveb, pro které navíc existují historické fotografie. Výhodou toho je, že vyhledáváme v připravené databázi pouze s obrázky vybraných staveb. Navíc můžeme využít známou pozici, ze které byla fotografie pořízena k významnému omezení počtu prohledávaných obrázků a zpřesnění výsledků.

# 2. Návrh řešení

Cílem práce tedy je ukázat proměny známých budov a potažmo města v průběhu času. Abychom toho mohli dosáhnout, potřebujeme přiřadit historické obrázky k současným uživatelským, v ideálním případě pouze na základě srovnání těchto obrázků. S historickými obrázky je však spojeno mnoho problémů. Je jich pouze omezené množství, jsou špatně dostupné a jsou většinou poměrně nekvalitní. U fotografií dostatečně starých na to, aby zachycovaly zajímavé změny, je velkým problémem špatná kvalita, vyblednutí barvy a poškození. Pokud chceme ukázat starší změny vzhledu staveb, musíme se, vzhledem k tomu, že fotografie byly vynalezeny teprve před necelými dvěma stoletími, spokojit s kresbami či rytinami. Ty ovšem většinou zachycují objekty velice nepřesně a zkresleně v závislosti na uměleckém stylu, který byl zrovna v módě.

Avšak ještě vážnější překážkou pro přiřazení historických obrázků k těm aktuálním je změna samotných staveb, o kterou nám jde. Chceme totiž ukázat zajímavé změny klasických památek a významných staveb. Takové změny ovšem většinou bývají natolik radikální a zásadní, že aktuální a historický vzhled stavby mají společné pouze hrubé rysy a často ani to ne (obr. 3). Proto tyto problémy vyřešíme použitím aktuálních fotografií staveb. Pro každou historickou fotografii se pokusíme pořídit v rámci možností co nejpodobnější aktuální fotografii a všechno

**Obrázek 3.** Příklad významné změny vzhledu stavby. Chrám svatého Víta na Pražském hradě z III. nádvoří. Vlevo současný fotografie upravená, aby odpovídala historické, protože historické fotoaparáty měli širší úhel záběru. Vpravo fotografie Františka Krátkého pořízená kolem roku 1891. Historická fotografie převzata z [3].

další zpracování se bude provádět na nich a až po zpracování budou ve výsledcích nahrazeny původními historickými fotografiemi.

Počítačové vidění a rozpoznávání obrazu je velice mladý obor zažívající překotný rozvoj s nepřeberným množstvím různých metod a přístupů. Hlavní vlastností metod počítačového vidění je to, že na problémy nejsou žádné jasné odpovědi a bezchybná řešení. Existují metody pro víceméně úspěšné řešení omezených problémů, ale rozsah zvládnutých problémů se stále mění s probíhajícím vývojem a výzkumem. Je tedy nutné problém omezit na některou ze standardních úloh, které jsou již dobře prozkoumány.

V této práci se proto omezíme problém na vyhledávání v databázi obrázků. Zřejmě nejrozšířenější metodou v této oblasti je metoda *Bag of Words* (Pytel slov) [4]. V každém obrázku v databázi jsou nalezeny výrazné oblasti a z charakteristických vlastnosti každé oblasti se utvoří její popis, takzvaný deskriptor. Z těchto deskriptorů ze všech obrázků se statistickými metodami automaticky extrahují charakteristické třídy (tzv. vizuální slova) blízce si podobných deskriptorů. Deskriptory z každého obrázku se roztřídí do těchto připravených tříd a normalizovaný vektor četností těchto tříd potom reprezentuje obrázek. Pak lze velice efektivně porovnat podobnost obrázků metrikou založenou na skalárním součinu jejich reprezentací.

Výsledkem je ovšem pouze podobnost reprezentací obrázků, která nemusí nezbytně odpovídat podobnosti objektů na nich zachycených. Pokud jsou si obrázky skutečně podobné, pak jsou odpovídající si body na nich vázány nějakou geometrickou transformací, kterou se můžeme pokusit odhadnout a ověřit. Pokud najdeme odpovídající transformaci, obrázek je pravděpodobně skutečně podobný hledanému.

## 2.1. Zpracování obrazu

Na začátku zpracování máme fotografii pořízenou pravděpodobně mobilním zařízením. Vzhledem k tomu, že je pořízena vlastní mobilní aplikací, můžeme předpokládat, že nebyla vyfotografována s použitím zoomu a přisvětlení.

Fotografie se nejdříve převede do 256 odstínů šedi. Většina metod pracuje zejména s jasovou složkou obrazu, ve které jsou uloženy klíčové informace o struktuře objektu. Barevné složky nepřidávají tolik informace, aby to vyvážilo vyšší náročnost na implementaci a výpočetní a paměťovou náročnost. Navíc mobilní zařízení nemají vesměs příliš kvalitní fotoaparáty a mají velký barevný šum.

Jedním z problémů fotoaparátů v mobilních telefonech je špatný dynamický kontrast, to ve spojeni s tmavou barvou některých staveb jako Prašná brána či Staroměstská radnice a nutností fotografovat takové vysoké stavby proti obloze vede ke špatné rozpoznatelnosti fotografie. V moderních zařízeních by mohlo být řešením použití HDR režimu, který je součástí nejmodernějších verzí operačního systému Android, ale na většině starších zařízení chybí.

Proto byly otestovány dvě metody vylepšení kontrastu. První metodou je ekvalizace histogramu. V té se histogram obrázku transformuje, tak aby se rozdělení jasu maximálně blížilo rovnoměrnému. Došlo ke znatelnému vylepšení rozpozná-

vání problematických staveb, ale na úkor všech ostatních, takže celkovým výsledkem bylo zhoršení rozpoznávání na testovací sadě o 10%.

Druhou metodou bylo odstranění oblohy na základě analýzy histogramu. Pokud se v histogramu zjistí přítomnost silné jasné jasové složky, o které lze předpokládat, že jde o oblohu, tak se odstraní a zbytek jasové složky obrázku se roztáhne na celou šířku pásma.

Následně se obrázek zmenší na takové rozlišení, aby delší strana obrázku měla 800 px, což vychází jako vhodný kompromis mezi dostatečnou přesností a výpočetní náročností. Navíc se vytvoří obrázek poloviční velikosti zmenšeného, ve kterém se později budou vypočítávat deskriptory.

### 2.1.1. Detekce bodů zájmu

Dalším krokem je najít ve fotografii výrazné rysy, o kterých se dá předpokládat, že budou vidět a budou výrazné i na dalších fotografiích stejného objektu. Velké plochy obrázku obsahují jen velice málo užitečné informace. Například obloha nebo jednobarevná zeď. A protože zpracování celého obrazu je velice náročné, musíme vybrat v obrázku místa, která obsahují hodně informace a dále pracovat s těmi.

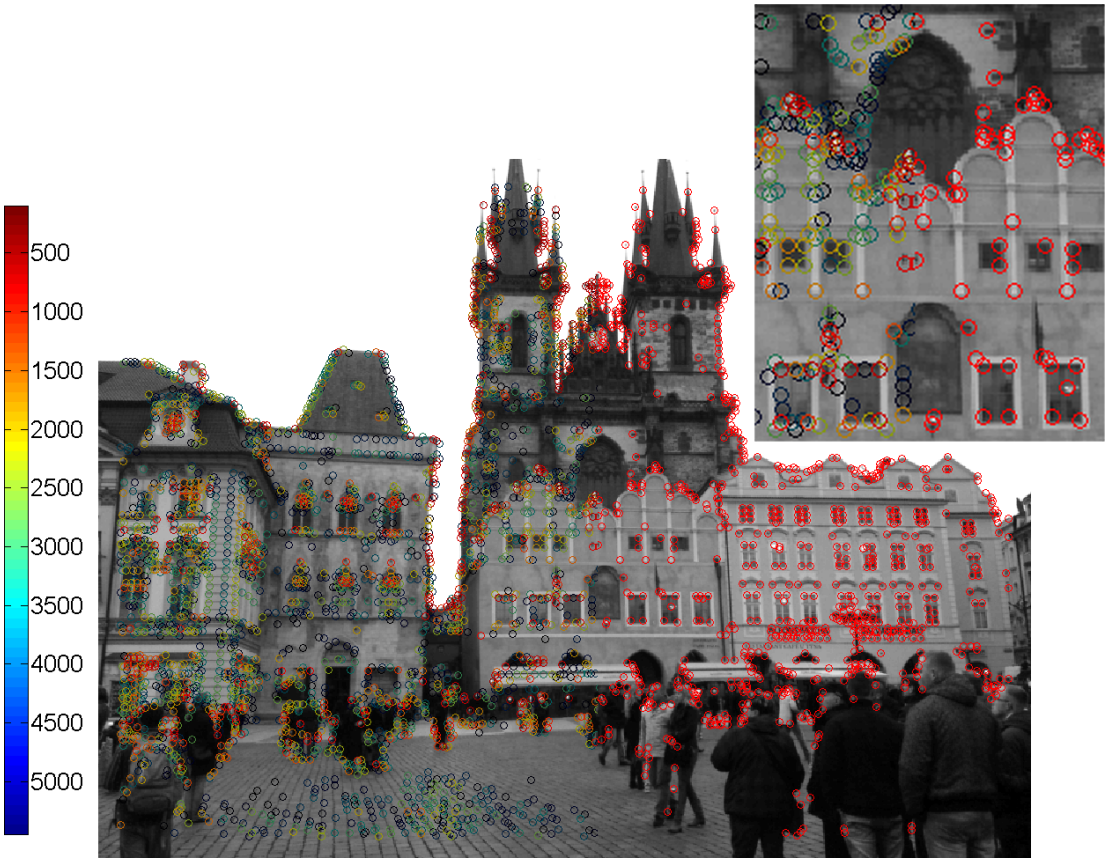
K tomu slouží detektory takzvaných bodů zájmu. Body zájmu jsou taková místa v obrázku, které jsou jednoznačně lokalizovatelná. Nicméně různé detektory mají různá kritéria pro to co je a není bodem zájmu. Na čisté bílé zdi například nemůžeme jednoznačně vybrat žádný jeden bod odlišný od okolí. Ale bude-li ve zdi například okno můžeme jednoznačně vybrat jeho rohy nebo střed.

Široce používanou je metoda MSER (*Maximally Stable Extremal Regions*) [5], ta považuje za body zájmu oblasti, které mají jednotné vlastnosti (jas, barvu,...) a výrazně odlišné od svého okolí a jejich okraje. Další metodou, která je podle výzkumů podobná zpracování obrazu v mozcích savců, je metoda lokálních maxim Hessiánu (determinant z Hessovy matice), ta považuje za body zájmu přibližné středy výrazných skvrn dobře odlišitelných od svého okolí. Často používaný je také Harrisův-Stephensův detektor rohů [6], body zájmu jsou v tomto případě v místech přesné lokalizovatelnosti v obraze.

Metoda MSER vychází ze srovnání většinou nejlepší [7], ale zároveň nalézá výrazně méně bodů než zbylé metody, a hůře se vyrovnává s rozmazáním obrazu, což je jev, se kterým se u fotografií z mobilních zařízení musí počítat. Navíc vzhledem ke statistické povaze metody pytel slov je lepší mít bodů zájmu více. Proto byla metoda MSER zavržena již v počátku projektu.

Lokální extrémy Hessiánu se vyskytují v oblastech, které silně kontrastují s jejich okolím. Detektor má tedy silnou odezvu například uprostřed tabulek oken (obr. 4).

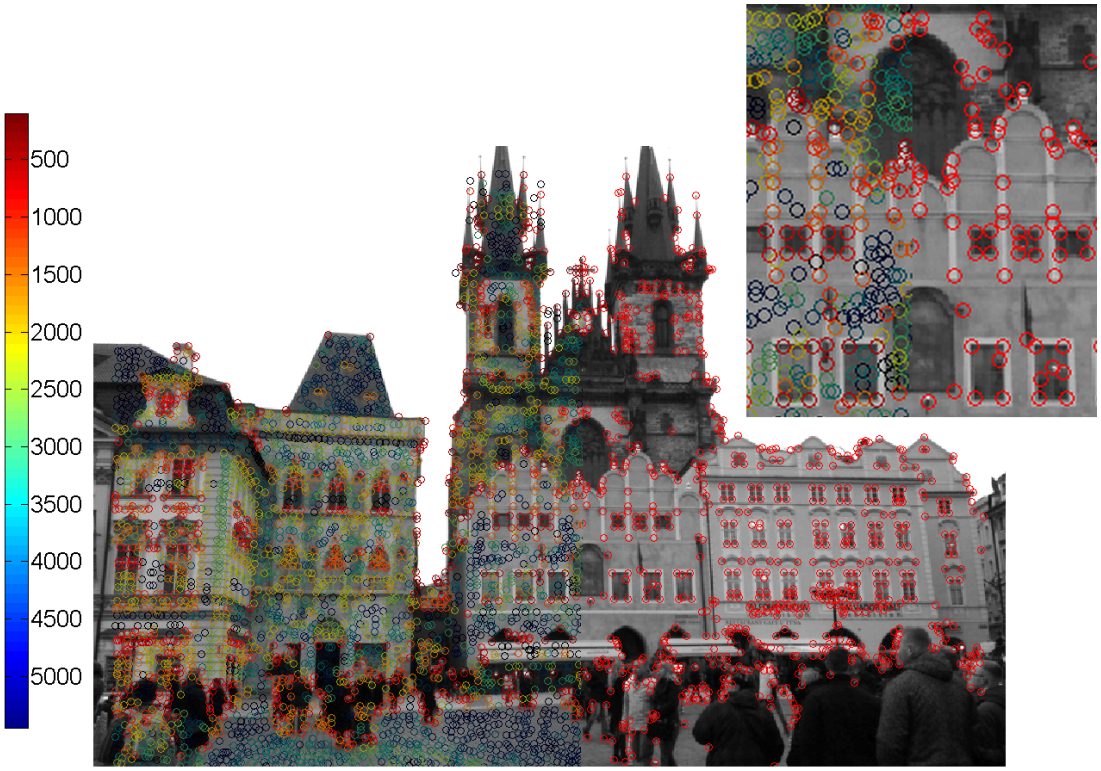
Harrisův detektor rohů, má největší odezvu v dobře lokalizovatelných místech v obrázku, jako jsou například rohy (obr. 5). Tedy v oblastech obrázku, kde jsou obě derivace intenzity podle obou os velké. Vzhledem k tomu, že objekty, jimiž se tato práce zabývá, jsou převážně historické budovy a tedy objekty bohaté na ostré linie a rohy, zdá se Harrisův detektor býti lepší volbou.




**Obrázek 4.** Příklad bodů zájmu nalezených metodou lokálních maxim Hessiánu. V levé polovině obrázku prvních 5500 nejlepších bodů zájmu. Barva kroužku značí pořadí bodu zájmu. Body s nejlepší odezvou jsou červené a s nejhorší modré. V pravé polovině 1500 nejlepších bodů. Vpravo nahoře detailní střední části obrázku. Body detekovány v rozlišení $800 \times 600$ px při $\sigma_f = 1$.

Hlavním parametrem těchto metod je rozptyl filtračního gausiánu, který určuje měřítko, to znamená, z jak velké oblasti obrázku se vypočítá odezva funkce. Při předběžných experimentech se při použití lokálních maxim Hessiánu dosahovalo nejlepších výsledků při nastavení rozptylu derivačního gausiánu na $\sigma_f = 3,25$. A u Harrisova detektoru při rozptylu filtračního gausiánu $\sigma_f = 1$. V textech se většinou doporučuje rozptyl integračního gausiánu jako $\sigma_i = 0.7\sigma_d$, ale při testování se lepších výsledků dosahovalo s $\sigma_i = \sigma_f$.

Často se u těchto detektorů vybírají všechny body zájmu, jejichž odezva překročí nastavenou hodnotu prahu. Při předběžném testování se dosahovalo lepších výsledků s pevným počtem nejlepších odezev než s výběrem na základě prahování. Pro oba detektory se tak z obrázku vybere maximálně 1500 největších lokálních extrémů odpovídající funkce. Což pro většinu obrázků představovalo hranici, při které ještě šlo převážně o zajímavé body a ne o náhodný šum. Z jednoho obrázku se v průměru vyprodukuje 5000 kandidátů na body zájmu.




**Obrázek 5.** Příklad bodů zájmu nalezených Harrisovým detektorem rohů. V levé polovině obrázku prvních 5500 nejlepších bodů zájmu. Barva kroužku značí pořadí bodu zájmu. Body s nejlepší odezvou jsou červené a s nejhorší modré. V pravé polovině 1500 nejlepších bodů. Vpravo nahoře detailní střední části obrázku. Body detekovány v rozlišení $800 \times 600$ px při $\sigma_f = \sigma_i = 1$.

### 2.1.2. Měřítko

V obrázku scény jsou pravděpodobně body zájmu v různých úrovních detailů. Například ozdoby na fasádách jsou nejmenší detaily, okna pak větší detaily a největší rysy budou rohy budovy. Pokud hledáme body zájmu pouze v jednom měřítku, najdeme jenom jednu úroveň výrazných rysů objektu.

Proto se používá takzvaný prostor měřítek. Ten je tvořen několika kopiemi obrázku, kde každá úroveň je filtrována gausiánem s větším rozptylem než předchozí. První úroveň se například filtruje gausiánem s rozptylem $\sigma_{i=1} = 1$ a další například $\sigma_{i+1} = 1.2\,\sigma_i$. To umožňuje najít nejenom různě veliké rysy v obraze, ale navíc lépe srovnávat s obrázky pořízenými z různých vzdálenosti, protože obvyklé deskriptory jsou poměrně citlivé na změnu měřítka.

Abychom nebyli zahlcení velkým množstvím stejných bodů zájmu v různých měřítcích, potřebujeme najít lokální maxima v prostoru měřítek. Ta se najdou srovnáním odezvy v bodu zájmu v daném měřítku s odezvou v obrázku na nižší a vyšší sousedící úrovni měřítka. Pro metodu lokálních maxim Hessiánu můžeme použít pro srovnání samotnou odezvu funkce normalizovanou $\sigma^2$. Odezva Harrisova detektoru rohů je ovšem špatně porovnatelná přes různá měřítka a proto se

pro srovnání používá normalizovaný Laplacián gausiánu

$$\nabla^2 L(x, \sigma_i) = \mid \sigma_i^2 (L_{xx}(x, \sigma_i) + L_{yy}(x, \sigma_i)) \mid .$$

### 2.1.3. Afinní normalizace

Okolí jednoho zájmového bodu v obrázcích stejného objektu z různých úhlů jsou zkreslena různou projektivní transformací (homografií). To komplikuje následné přiřazení vzájemně si odpovídajících bodů zájmu. Pro malé oblasti můžeme tuto transformaci aproximovat pomocí afinní transformace. Když odhadneme tuto afinní transformaci, můžeme oblast zpětně transformovat a dosáhnout vyšší invariance vůči změně úhlu pohledu.

Podle výsledků studie [7] dosahují afinně invariantní verze Harrisova detektoru i detektoru založeném na metodě lokálních maxim Hessiánu podobných výsledků. Metoda lokálních maxim Hessiánu sice v porovnání dosahuje o něco lepších výsledků, ale vzhledem k tomu, že afinní normalizace vychází z matice druhých momentů, která se musí pro metodu lokálních maxim Hessiánu vypočítat navíc, zdála se být afinně invariantní verze Harrisova detektoru [8] lepší volbou pro testování.

Další metoda afinní normalizace je založena na lokálních maximech Hessiánu [9]. Podle výsledků srovnání má pro městskou scénu srovnatelné výsledky s Harrisovým detektorem, ale je méně výpočetně náročná. Tato metoda by mohla být vhodným budoucím vylepšením, nicméně pro účely ověření přínosu afinně invariantních bodů zájmu jsem se rozhodl implementovat prověřenější metodu založenou na Harrisových bodech zájmu.

Pro každý nalezený bod zájmu známe jeho počáteční polohu $x_0 \in \mathbb{R}^2$, měřítko $\sigma_0$, ve kterém byl nalezen a matici druhých momentů $\mu_0 \in \mathbb{R}^{2,2}$. Z matice druhých momentů vypočteme transformační matici

$$u_0 = \mu_0^{-\frac{1}{2}} \in \mathbb{R}^{2,2},$$

která normalizuje rozložení gradientů v oblasti tak, aby bylo rovnoměrné.

Nalezení stabilní transformace probíhá iterativně. Nejdříve transformujeme okolí bodu zájmu transformací

$$U_i = \prod_{j=0}^{i} u_j.$$

V transformovaném okolí bodu zájmu najdeme novou polohu $x_{i+1}$ bodu zájmu, tedy maximum odezvy Harrisova detektoru. Při tom získáme také novou matici druhých momentů $\mu_{i+1}$. Kritériem pro zastavení iterací je nalezení stabilní transformace, která se již významně nemění nebo dosažení nastaveného limitu počtu iterací. Afinní normalizaci okolí považujeme za stabilní pokud je poměru vlastních čísel matice $\mu_{i+1}$ větší než nastavený práh

$$\frac{\lambda_{min}(\mu_{i+1})}{\lambda_{max}(\mu_{i+1})} > c,$$

kde používanou hodnotou $c$ je $c = 0.96$ viz. [8].

Afinní a vůči měřítku invariantní detektory jsou ovšem výrazně výpočetně náročnější než původní jednoduché detektory. A navíc produkují menší množství bodů zájmu na obrázek, což je nevýhodné pro metodu pytel slov.

Protože k dalšímu zpracování (kap. 2.3.1) je potřeba informace o afinní transformaci každého regionu, je tato hrubě extrahována z matice druhých momentů i pro první dva detektory, ale není iterativně vylepšována pro dosažení stabilních regionů jako u standardních implementací těchto metod.

### 2.1.4. Deskriptory

Abychom poznali, že dvě oblasti ukazují stejný bod zájmu i za přítomnosti zkreslení, šumu a dalších možných komplikací, potřebujeme oblasti nějak popsat. Nejzákladnější reprezentaci oblasti jsou samotné pixely, ale je velice náročné takové reprezentace porovnávat a navíc jsou hodnoty pixelů velice citlivé na zkreslení a změny. Proto se používají deskriptory, ty vyextrahují z výřezu důležité informace a potlačí závislost na geometrických a fotometrických deformacích a uloží je ve snáze porovnatelné formě.

Existuje nepřeberné množství deskriptorů, ale zřejmě nejpoužívanějším z nich je SIFT (*Scale-Invariant Feature Transform*) [10]. Je velice dobře popsán a otestován na mnoha problémech. Proto je dobrou první volbou, protože lze srovnávat výsledky s literaturou. Zde je použit v klasické 128-dimenzionální verzi. Vzhledem k zaměření práce na budovy je zajímavým deskriptorem také Local Symmetry Features [11]. Ale zdá se zaměřený především na výrazně odlišné znázornění stejných budov, jako například denní a noční fotografie.

Deskriptor zde popisuje oblast o velikosti $16 \times 16$ px se středem v bodu zájmu, ale získanou z obrázku zmenšeného podle měřítka, ve kterém byl bod zájmu nalezen, takže v praxi odpovídají pro základní Harrisův detektor výřezu o velikosti $32 \times 32$ px v původním obrázku.

Protože mohou být stejné bodů zájmu v různých obrázcích vzájemně pootočeny, je výhodné provést na výřezech rotační normalizaci. V původním SIFTu [10] se používá k určení orientace dominantní úhel. Ze všech vektorů obrazových gradientů vypočítaných metodou se vytvoří histogram úhlů a nejčastěji se vyskytující úhel je označen za dominantní a výřez se o něj pootočí.

Region se rozdělí na $4 \times 4$ sektory a pro každý sektor se spočítá histogram úhlů gradientu upravených o nalezenou dominantní orientaci s osmi intervaly. Výsledný histogram se filtruje konvolucí s jádrem [0.2, 0.6, 0.2]. Což je jednoduchá metoda jak potlačit kvantizační chybu a vliv malého počtu měření na kvalitu histogramu ve smyslu podobnosti skutečnému (neznámému) rozdělení.

## 2.2. Vytvoření databáze

Když přijmeme neznámý obrázek, je naším cílem určit, zda na něm vidíme jeden z nám známých objektů. K tomu potřebujeme mít informace o tom jak vypadají. Proto pro každý objekt pořídíme sadu fotografií, které ho dobře reprezentují. Abychom nemuseli každý hledaný obrázek náročně porovnávat se všemi ob-

rázky, které jsme nasbírali, musíme si vytvořit vyhledávací strukturu, která nám to usnadní. K tomu slouží metoda pytel slov, která nám z obrázků vyextrahuje jejich jednoduché popisy, mezi kterými můžeme snadno a rychle vyhledávat.

Ve všech vybraných obrázcích se najdou body zájmu a z jejich okolí se vypočtou deskriptory. Samotné obrázky pak již pro další zpracování nebudou potřeba, ale bude se pracovat pouze s jejich reprezentacemi v podobě bodů zájmu. Pro další snížení výpočetní náročnosti ještě zredukujeme deskriptory pomocí statistických metod na menší velikost, zachovávající většinu informace. Pak již použijeme metodu pytel slov pro vytvoření efektivní vyhledávací struktury.

### 2.2.1. Datová sada

Pro účely vývoje bylo vybráno 16 různých výrazných staveb a dominant v centru Prahy (obr. 6). Protože díky zvolené metodě řešení není pro vyhledávání obrázků podstatný jejich historický vzhled, nebyly objekty vybírány primárně podle jejich historické podoby a dostupných historických fotografií, ale spíše s důrazem na jejich dobrou přístupnost, možnost fotografování a odlišnost stylů, aby výsledná aplikace nebyla příliš zaměřená na jeden styl staveb. Proto jsou, vedle notoricky známých pražských dominant, součástí také moderní stavby.

Pro každý objekt byla pořízena sada fotografií z různých míst a pokud možno i za různých podmínek, tak aby mělo vyhledávání k dispozici pokud možno kompletní aktuální vzhled budovy. Naštěstí jsou v husté pražské zástavbě poměrně omezené možnosti výběru místa pro fotografování. Příklad jak vypadá sada fotografií pro jeden objekt je na obr. 7. Fotografie byly pořízeny fotoaparátem v mobilním telefonu, aby byly technicky co nejpodobnější fotografiím, se kterými se budou následně srovnávat.

### 2.2.2. Snížení dimensionality

Vzhledem k náročnosti výpočtu se 128-dimenzionálními vektory SIFT a tomu, že klíčovým krokem metody je kvantizace deskriptorů do tříd, kde se ztratí značné množství jejich informace, můžeme bez významného dopadu na výsledek použít metodu PCA (*Principal Component Analysis*) [12] na snížení dimensionality. Podle předběžných experimentů by měly být deskriptory redukované na 32 dimenzí přibližně stejně účinné jako deskriptory v plné dimenzi (obr. 8).

### 2.2.3. Shlukování

Klíčovým krokem metody pytel slov je nalezení tříd podobných deskriptorů (vizuálních slov). Vzhledem k tomu, že nemáme informace o tom jak by třídy měly vypadat, používají se metody shlukování. V literatuře [4] je nejčastěji úspěšně používán algoritmus K-středů. Jeho výsledkem je pro předem nastavený počet středů shluků takové přiřazení všech bodů, že každá změna přiřazení bodů k prototypům (středům) zvýší součet vzdáleností bodů od jim přiřazených středů. To je ovšem pouze lokální minimum, což se obchází vícenásobným spuštěním algoritmu.

**(a)** Týnská ulička

**(b)** Městský soud, Karlovo náměstí

**(c)** Chrám Matky Boží před Týnem

**(d)** Městská knihovna

**(e)** Kostel sv. Vojtěcha

**(f)** Prašná brána

**(g)** Rytířská, čp.29

**(h)** budova ČVUT, Karlovo náměstí

**(i)** OD Kotva

**(j)** Obecní dům

**(k)** OC Paladium

**(l)** Kostel sv. Mikuláše

**(m)** Sousoší Františka Palackého

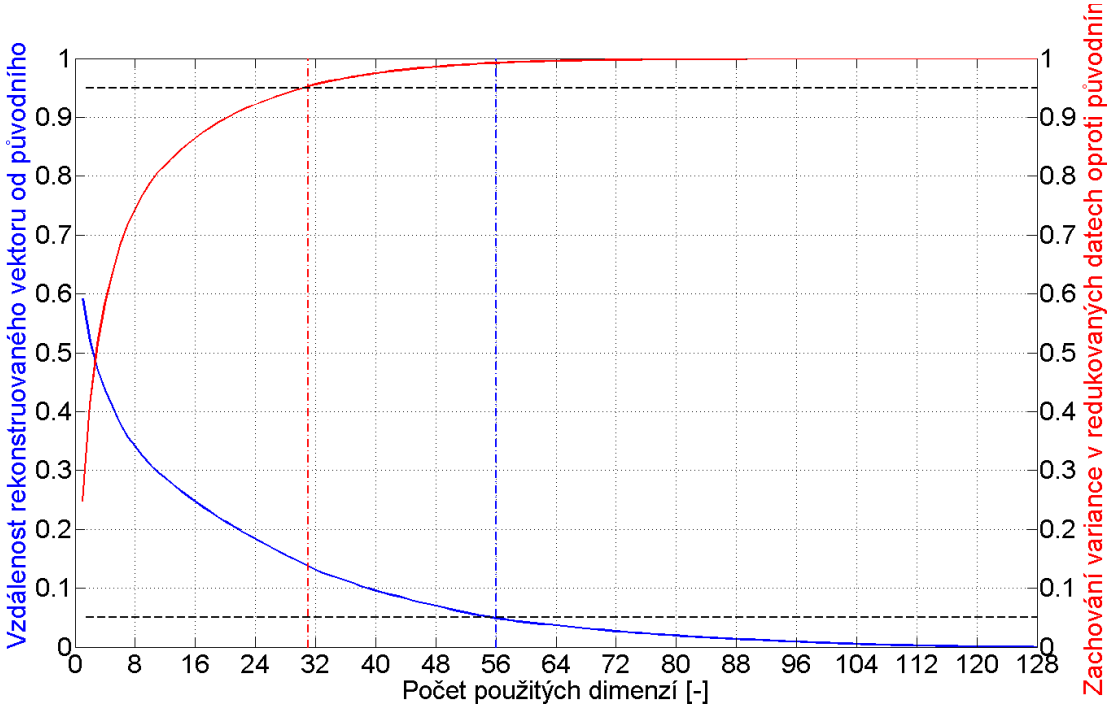
**(n)** Škola Vojtěšská

**(o)** Staroměstská radnice

**(p)** Novoměstská radnice

**Obrázek 6.** Objekty v databázi

**Obrázek 7.** Různé pohledy na jeden objekt




**Obrázek 8.** Vliv počtu použitých dimenzí PCA na zachování variance dat (*červená*) z velikosti vlastních čísel a průměrnou odchylku zpětně rekonstruovaných vektorů od původních na celé datové sadě (*modrá*). Vyznačeny počty dimenzí potřebných pro zachování 95% podobnosti s původním vektorem.

Nejdůležitějším parametrem metody je počet shluků. Příliš mnoho shluků znamená příliš těsné třídy a stejné prvky v různých obrázcích pak budou klasifikovány do různých tříd, naopak příliš málo shluků vede na to, že všechny prvky začnou splývat. Při předběžném testování se dosahovalo nejlepších výsledků s nastavením počtu shluků tak, aby do průměrného shluku spadalo přibližně 50-200 deskriptorů. V literatuře jsou používány podobné hodnoty [4],[13].

Protože každý obrázek v databázi je reprezentován až 1500 deskriptory, je již při stovkách obrázků celkový počet deskriptorů příliš velký na efektivní použití standardních metod shlukování. Proto se používají přibližné varianty metody K-středů. Existuje mnoho implementací algoritmu a zde byla implementována hierarchická verze, podobná metodě [14]. Další variantou je přibližné shlukování na základě nejbližšího souseda [13], která dosahuje o něco lepších výsledků, ale cílem této práce není vytvořit databázi s miliony obrázků, na které by se zlepšení výrazněji projevilo, a proto se spokojíme s jednodušší verzí a verzi na základě nejbližšího souseda necháme pro případná budoucí vylepšení.

Použijeme tedy hierarchický algoritmus K-středů s eukleidovskou vzdáleností. Data se nejdříve rekurzivně dělí pomocí algoritmu K-středů vždy na několik (3-10) velkých shluků dokud počet deskriptorů v některém shluku neklesne pod práh. Ten je nastaven na 5000, což bylo určeno jako maximum, při kterém shlukování ještě probíhá přijatelnou rychlostí. Pro každý z těchto velkých shluků se vypočte, kolik by měl obsahovat v průměru vizuálních slov a rozdělí se opět metodou K-středů. Oproti zmíněné podobné metodě by se tím měl omezit vznik chyb na

okrajích shluků.

Ze středů shluků se pak vytvoří k-D strom, který bude sloužit ke klasifikaci deskriptorů v dotazu.

### 2.2.4. Pytel slov

Středy shluků tedy tvoří vizuální slova metody pytel slov. Z nalezeného přiřazení deskriptorů ke středům shluků, tedy vizuálním slovům, se vypočte počet jednotlivých vizuálních slov v obrázku. Četnost všech vizuálních slov se uloží do vektoru, v podstatě histogramu výskytu vizuálních slov v obrázku. Máme-li $k$ vizuálních slov, pak pro obrázek $j$ vznikne vektor četností $\vec{n}_j = (n_1, ..., n_i, ..., n_k)^\top$. Vizuální slova v něm mají náhodné, avšak pevně určené pořadí vzniklé v průběhu shlukování.

Vektor se normalizuje standardní metodou *tf-idf* (*term frequency-inverse document frequency*) [4], která diskriminuje často se opakující slova nastavením malé váhy a velkou vahou upřednostňuje výjimečně se vyskytující slova. Místo prostých četností $n_i$ se pro každé vizuální slovo $w_i$ vypočte vážená četnost $t_i$. Vektorovou reprezentaci (obr. 9) obrázku $j$ pak tvoří $\vec{t}_j = (t_1, ..., t_i, ..., t_k)^\top$, kde

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i} \tag{1}$$

a kde $n_{id}$ je počet výskytů vizuálního slova $i$ v obrázku $d$, $n_d$ je počet slov v obrázku $d$, $n_i$ je počet tříd, v jejichž obrázcích se vizuální slovo $i$ vyskytuje. $N$ je pak počet tříd obrázků v databázi.

Nejčastěji se vyskytujícím slovům se navíc nastaví nulová váha. Protože se vyskytují ve velkém množství tříd, nepřispívají příliš k odlišitelnosti obrázků. Podle původního článku [4] se osvědčilo vynulovat četnost 10% nejčastějších vizuálních slov. Protože pracujeme s budovami, na kterých se totožné prvky velice často mnohokrát opakují, snížili jsme tuto hodnotu na 5%. Důležité je následně normalizovat délku vektoru $t_i$ na jednotkovou, aby následným skalárním součinem s dotazem vyšla přímo podobnost dotazu od obrázku. Je-li $a^\top \cdot b$ skalární součin jednotkových vektorů $a$,$b$, potom $1 - a^\top \cdot b$ je metrika.

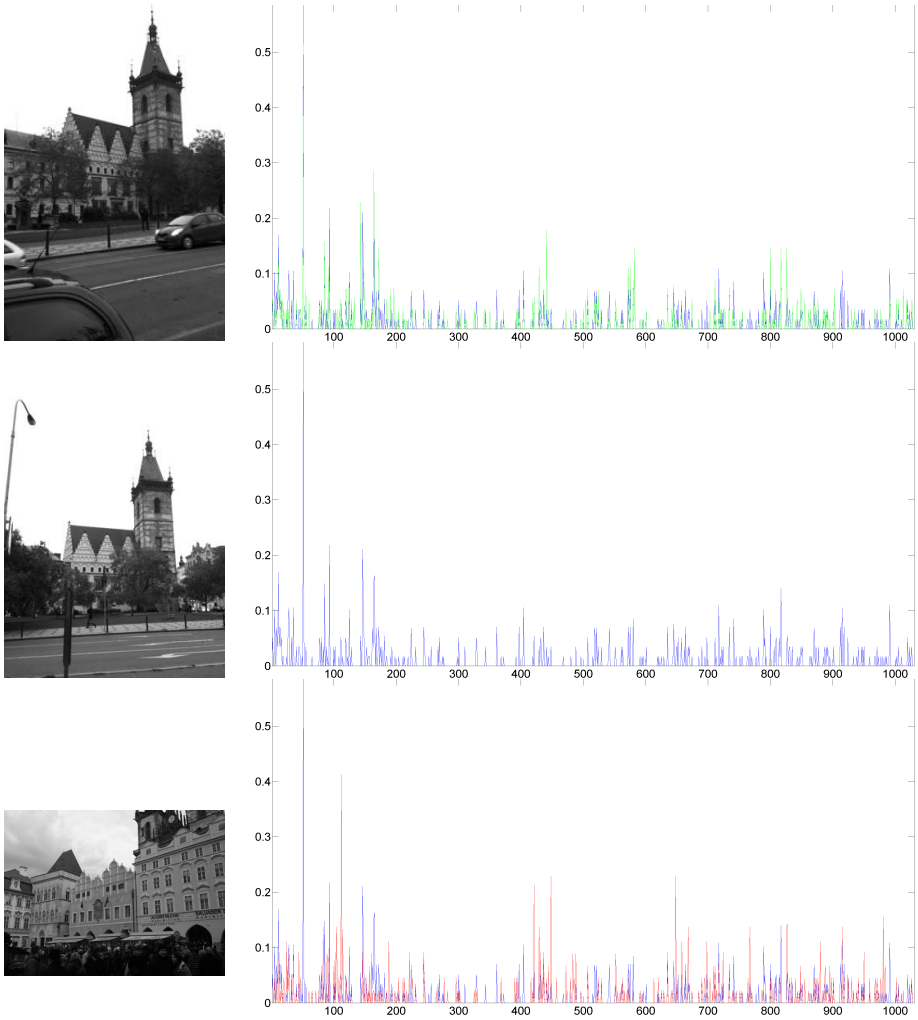
Poskládáním vektorů ze všech obrázků se následně vytvoří matice $\mathbf{B}$ reprezentující všechny obrázky v databázi sloužící k následnému vyhledávání. Máme-li $m$ obrázků, pak matice $\mathbf{B}$ bude mít podobu

$$\mathbf{B} = [\vec{t}_1, ..., \vec{t}_m] \qquad \mathbf{B} \in \mathbb{R}^{k,m}.$$

### 2.2.5. Vylepšení databáze

Existuje mnoho navrhovaných vylepšení metody pytel slov. Vzhledem k malé velikosti a úzkému zaměření naší databáze, pravděpodobně nevyváží přínos těchto vylepšení čas nutný na jejich implementaci. Mohly by však být zajímavým směrem pro další vývoj.

Základní metoda pytel slov porovnává popisy obrázků pouze na základě vizuálních slov přítomných v obou obrázcích. Nebere tak v potaz absenci vizuálního




**Obrázek 9.** Příklady *tf-idf* popisů $\bar{t}$ obrázků metodou pytel slov. Uprostřed je dotaz, nahoře je podobný obrázek a dole nepodobný. Vpravo pak je vizualizace popisu obrázku ve formě histogramu. Na osách x jsou vizuální slova a na osách y jejich váhy v daném obrázku. Problém je, že pořadí na ose x je sice pevně dané, ale nenese žádnou informaci o blízkosti sousedního vizuálního slova, takže nelze graf rozumně zpřehlednit. Je ale vidět že modrý a zelený graf mají velké špičky na stejných místech, zatímco červený má špičky jinde. Modře je popis obrázku dotazu, zeleně popis podobného obrázku a červeně popis nepodobného obrázku. Podobnost podobného obrázku vychází na **0.7455** a nepodobného na **0.3187**, kde 0 je absolutní nepodobnost a 1 je naprostá shoda.

slova v obou obrázcích. V článku [15] navrhují řešení odečtením střední hodnoty pytle slov od každého jejího vektoru a ukazují vylepšení výsledků. Upravená podoba pytle slova pak bude

$$\mathbf{B} = [\vec{t_1} - \alpha\,\bar{t}, ..., \vec{t_m} - \alpha\,\bar{t}] \qquad \mathbf{B} \in \mathbb{R}^{k,m},$$

kde

$$\bar{t} = \frac{\sum_{i=1}^{m} \vec{t_i}}{m} \qquad a \qquad \alpha \in \langle 0, 1 \rangle.$$

Pro $\alpha = 0$ jde pak o standardní metodu pytel slov.

Další zajímavou metodou [16] je odstranění takových vizuálních slov z popisu obrázku, která nepatří objektu, který má obrázek zobrazovat, například lidé na fotografii budovy. Po vytvoření databáze pytle slov se v ní každý obrázek vyhledá a na obrázcích, které vyjdou jako podobné, se geometricky ověří korespondence vizuálních slov. Vizuálním slovům, pro které se podaří najít kvalitní korespondence v dalších obrázcích se přiřadí váha podle počtu dalších obrázků, ve kterých byla ověřena. Vizuální slova, pro která se nepodařilo ověřit korespondence se z popisu obrázku vynechají.

## 2.3. Zpracování dotazu

Spolu s obrázkem v dotazu navíc dostaneme, pokud jsou dostupné, informace ze senzorů mobilního zařízení. Především údaje o poloze, měření magnetometru a akcelerometru. Magnetometr měří velikost magnetického pole a jeho směr ve třech dimenzích. Z toho lze vypočítat orientaci zařízení vůči zemskému magnetickému poli. Akcelerometr pak měří ve třech dimenzích zrychlení působící na mobilní zařízení.

Pokud je tedy zařízení v klidu, působí na něj pouze gravitační zrychlení a můžeme tak vypočítat natočení telefonu. Spojením údajů z magnetometru a akcelerometru můžeme přibližně určit, kterým směrem fotoaparát zařízení směřuje. Z údajů o poloze získané buď z GPS, nebo mobilní sítě, můžeme s větší či menší přesností určit, odkud je fotografie pořízena. Můžeme tedy při vyhledávání vynechat budovy, které uživatel nemůže vidět.

Obrázek může být předzpracován v mobilním zařízení. Následně se v něm najdou body zájmu a okolní zajímavé oblasti se popíší deskriptory. Následně se sníží dimenze deskriptorů pomocí PCA vektorů uložených v databázi. Výsledný popis se zašle na server. Tímto způsobem snížíme požadavky na kvalitu připojení a zároveň anonymizujeme obraz, protože z deskriptorů nelze obraz zpětně složit, což může být pro někoho důležité.

### 2.3.1. Postup vyhledávání

Vyhledávání probíhá v několika navazujících fázích. Postupuje se od nejrychlejších a nejméně přesných metod k metodám náročnějším a přesnějším. Každá úroveň tak zmenší množinu obrázků, které se musí prohledat na ty nejlepší. Na ty se

aplikuje další náročnější metoda, která opět zmenší počet kandidátů a tak dále. Tento přístup nám umožní mít větší databázi, zpracovat dotaz rychleji a použít náročnější metody pro zhodnocení podobnosti obrázků, protože je budeme aplikovat pouze na několik málo nejlepších kandidátů.

První fází je metoda pytel slov, ta nám umožní velice rychle, ale hrubě, vybrat nejlepší kandidáty na správnou odpověď. V další fázi spárujeme stejná vizuální slova v obrázcích nejpravděpodobnějších kandidátů s těmi v obrázku dotazu. Pro každý pár obrázků nám vyjde množina dvojic stejných vizuálních slov a jejich poloh v obrázcích. Z poloh bodů se můžeme pokusit odhadnout transformaci mezi obrázky. Protože pokud není podobnost obrázků náhodná, lze nalézt přibližnou transformaci souřadnic obrazů stejných bodů na skutečném objektu mezi obrázky. Pokud podoba náhodná je, nalézt dobrou transformaci se nepodaří.

Nejdříve se pokusíme transformaci hrubě odhadnout pomocí afinní transformace. U párů obrázků, pro které se podařilo najít transformace, které nejlépe popisují vzájemnou polohu bodů se pokusíme najít náročnější metodou přesnější transformaci, homografii. Pokud se podaří nalézt přijatelnou homografii mezi dvěma obrázky, můžeme ji použít k deformaci obrázku, který pošleme uživateli jako výsledek, tak aby přibližně vypadala jako z jeho úhlu pohledu.

**První fáze** Pro všechny deskriptory získané z obrázku dotazu se pomocí vyhledávání v k-D stromu najde nejbližší vizuální slovo. Četností všech vizuálních slov v obrázku pak metodou *tf-idf* podle (1) utvoří vektor $\vec{t}_d$ vážených četností vizuálních slov v obrázku. Výsledný vektor se vynásobí maticí $\mathbf{B}$ vizuálních slov databáze

$$\vec{p}_d = \mathbf{B}^\top \vec{t}_d.$$

Výsledkem je vektor podobností mezi dotazem a všemi obrázky v databázi $\vec{p}_d$. Z výsledků vybereme deset obrázků s největší podobností s obrázkem dotazu a ty postoupí do druhé fáze zpřesňování výsledků.

**Předběžné korespondence** Nejdříve ovšem potřebujeme v obou obrázcích nalézt odpovídající si vizuální slova. Protože je pravděpodobné, že stejné vizuální slovo se vyskytuje v obrázku vícekrát na různých místech nevíme, které s kterým párovat mezi obrázky. Proto musíme pracovat se všemi možnými páry, takzvanými předběžnými korespondencemi (obr. 10).

Tím ovšem může vzniknout značné množství chybných předběžných korespondencí. Na druhou stranu většina budov má například mnoho totožných oken, která by měla generovat stejná vizuální slova. Proto je nastaven práh pro celkový počet předběžných korespondencí na 3000. Pokud je předběžných korespondencí příliš, vynechávají se třídy s největším počtem vzájemných předběžných korespondencí, tak aby celkový počet nepřesáhl práh. Nicméně v průměrném páru se vytvoří pouze kolem dvou tisíc korespondencí.

**Druhá fáze** Protože máme více či méně přesný odhad afinní transformace normalizující okolí jednotlivých bodů, můžeme vzájemnou transformaci mezi obrázky

**Obrázek 10.** Příklad předběžných korespondencí mezi dvěma obrázky. Červené úsečky spojují jednotlivé páry potencionálně korespondujících vizuálních slov v obou obrázcích. Pro přehlednost náhodně vybráno 150 předběžných korespondencí z celkového počtu 2668.

v malém okolí hrubě aproximovat afinní transformací, kterou odhadneme z afinních transformací oblastí. Předběžné korespondence jsou tvořeny dvěma body zájmu ve dvou obrázcích normalizovaných pomocí afinních transformací $\mathbf{A}_q \in \mathbb{R}^{3,3}$ pro bod zájmu v obrázku dotazu a $\mathbf{A}_d$ pro odpovídající bod zájmu v obrázku v databázi. Odhad transformace z obrázku dotazu do obrázku v databázi tedy získáme jako

$$\mathbf{A} = \mathbf{A}_q \mathbf{A}_d^{-1}.$$

Tuto transformaci vypočteme pro každou předběžné korespondenci a transformujeme jí souřadnice všech slov v obrázku dotazu do obrázku v databázi

$$\bar{u}_i = \mathbf{A} u_i,$$

kde $u_i = (x_i, y_i, 1)^\top$ je homogenní vektor reprezentující polohu $i$-té předběžné korespondence v obrázku dotazu a $\bar{u}_i = \lambda_i(\bar{x}_i, \bar{y}_i, 1)^\top$ $\lambda_i \neq 0$ poloha transformovaná do obrázku v databázi.

Vypočteme druhou mocninu eukleidovské vzdálenosti transformovaných bodů $\bar{u}_i$ od bodů $u_i$ v obrázku z databáze a sečteme počet dvojic, které mají nižší vzdálenost než je nastavený práh tzv. podporu. Ze všech vypočtených transformací pro páry korespondencí vybereme tu s největší podporou a ta se stane novým ohodnocením podobnosti obrázků, podle kterého je řadíme. Tento postup je ekvivalentní vyčerpávajícímu hledání nejpravděpodobnější transformace mezi obrázky. K dalšímu zpřesnění výsledků vybereme pět obrázku s nejvyšším skóre a ty postoupí do další fáze upřesnění výsledků.

**Třetí fáze** Afinní transformaci lze vypočítat rychle, ale vzhledem k tomu, že neumožňuje modelovat projektivní zkreslení, tak až na výjimky příliš dobře nepopisuje vztah dvou obrázků. Proto se pokusíme model transformace upřesnit nalezením homografie. Pro rovinu, například fasádu budovy, zachycenou z různých úhlů můžeme najít homografii, která přesně transformuje body z jednoho obrázku fasády na jim odpovídající body na druhém obrázku.

**Obrázek 11.** Příklad výsledku třetí fáze vyhledávání. Nalezená transformace mezi dvěma fotografiemi z předchozího obrázku. Vlevo obrázek dotazu a vpravo obrázek z databáze transformovaný do úhlu pohledu dotazu. Zelené úsečky spojují páry korespondencí. Nalezená homografie je podporována 41 korespondencemi.

K nalezení homgrafie použijeme metodu RANSAC (*RANdom SAmple Consensus*) [17]. Metoda RANSAC robustně hledá model, který by dobře vystihoval vstupní data. Prohledat všechny možné kombinace vstupních dat pro nalezení globálně nejlepšího modelu, je vzhledem k tomu, že počet kombinaci roste s faktoriálem velikosti vstupní množiny prakticky nemožné. Hledání tedy probíhá tak, že metoda opakovaně vybírá z množiny vstupních dat náhodně potřebný počet vzorků, pro vypočtení modelu. Pro každý vypočtený model se stanoví na celé množině vstupních dat jeho podpora. Model s nejlepší podporou je výsledkem.

Modelem je v tomto případě homografie a vstupními daty jsou páry korespondencí, resp. páry poloh vizuálních slov ve dvou obrázcích. K vypočtení homografie jsou potřeba čtyři páry korespondencí. V každé iteraci algoritmu se tak náhodně vyberou čtyři páry korespondencí a vypočte se z nich homografie. Touto homografií se transformují polohy bodů všech korespondenci z jednoho obrázku do druhého a vypočte se jejich eukleidovská vzdálenost. Sečteme páry, které jsou si blíže, než je nastavený práh a výsledek tvoří podporu modelu. Jako nalezenou transformaci vybereme homografii s největší podporou.

Vedlejším produktem zpřesnění výsledku tak je transformace mezi obrázkem v databázi a uživatelovou fotografií (obr. 11). Protože můžeme ručně stanovit skutečné korespondence mezi historickou fotografií budovy a jejími současnými fotografiemi, které máme v databázi, můžeme si pro každý obrázek objektu v databázi vypočítat homografii, které transformuje historickou fotografii, tak aby byla podobná té aktuální. A tyto dvě homografie pak můžeme složit, abychom mohli uživateli odeslat historickou fotografii transformovanou přibližně do jeho úhlu pohledu.

### 2.3.2. Rozšířený dotaz

Kvalitní výsledky vyhledávání můžeme použít k dalšímu vylepšení vyhledávání metodou rozšíření dotazu [18]. Z nejlepších nalezených výsledků vybereme vizu-

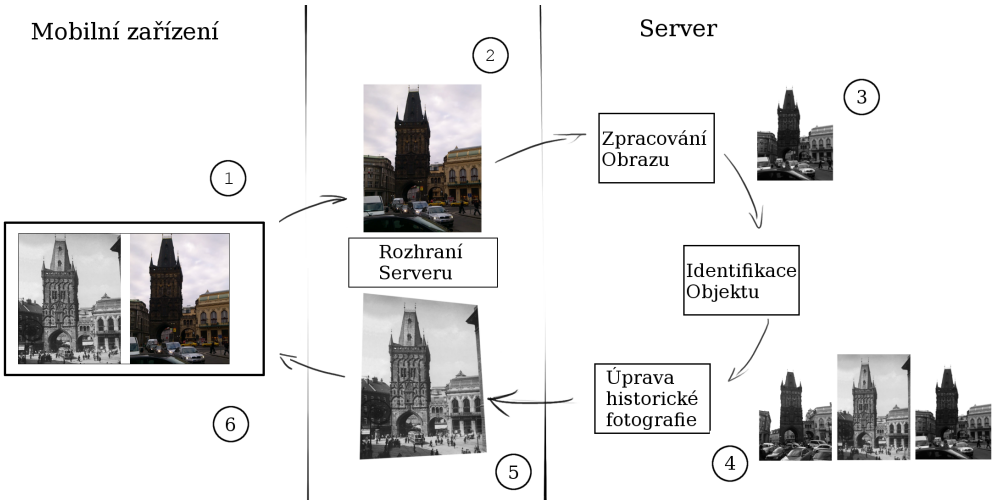
ální slova, která se v dotazu nenachází a transformujeme je do prostoru dotazu. Pak vytvoříme nový dotaz sloučením popisu obrázku dotazu s těmito transformovanými vizuálními slovy. Tím můžeme obohatit dotaz o části hledaného objektu, které například nejsou dobře vidět. Metoda na našich datech pravděpodobně nepřinese dostatečné zlepšení, aby vyvážila čas potřebný na její implementaci.

# 3. Implementace

Protože vyhledávání obrázků je datově a výpočetně velice náročné, byla aplikace navržena s architekturou klient-server. Klient tedy pořídí snímek, případně ho předzpracuje nakolik mu stačí výkon a pošle ho na server k vyhodnocení. Na serveru se provede výpočetně náročné vyhledávání a odpověď se pošle zpět klientovi. Nasbíraná data jsou zatím zaměřena pouze na velká města s velkou koncentrací turistů a památek, kde jsou dostupné moderní rychlé datové sítě takže není velký problém s prodlevou v komunikací mezi klientem a serverem.

Aby byla mobilní aplikace kompatibilní s co největším počtem i slabých zařízení, neprovádí se v ní žádné náročné zpracování a všechny výpočetně náročné operace obstará server. Klient tedy pouze pořídí snímek a odešle ho na server ke zpracování. Veškeré zpracování obrazu se pak provádí na serveru.

Komunikační rozhraní serveru implementované v PHP a Perlu vytvořil Ing. Daniel Večerka. Komunikace mezi klientem a serverem probíhá pomocí protokolu HTTP. Vnější rozhraní je založené na PHP a HTML. Je tedy možné se serverem interagovat např. z webového prohlížeče.




**Obrázek 12.** Schématické znázornění funkce systému. V bodě jedna uživatel pořídí snímek. Ten se bez úprav odešle po síti na server (bod 2), kde se obrázek zpracuje (bod 3), identifikuje se jestli zda je na něm známá stavba. V bodě 4 se pak odhadne transformace mezi obrazem dotazu a historickou fotografií. V bodě 5 se výsledek odešle zpět uživateli a zobrazí se mu na obrazovce (bod 6).

## 3.1. Mobilní aplikace – Klient

V současnosti jsou nejrozšířenějšími mobilními platformami operační systémy Android, iOS a Windows. V Evropě a v České Republice je jednoznačně nejrozšířenější operační systém Android a ostatní systémy jsou ve výrazné menšině.

Mobilní zařízení s operačním systémem Windows jsou mezi uživateli spíše raritou a nemá cenu se jimi tedy příliš zabývat. Operační systém iOS je velice uzavřený, omezený a oficiálně podporované vývojové prostředí pro něj je dostupné pouze pro operační systém Mac OS. Uzavřenost platformy má výhodu ve vyšší rychlosti systému a jednoduchosti ladění aplikací. Stačí odladit aplikaci pro několik málo konfigurací, na rozdíl od systému Android s nepřeberným množstvím všemožných konfigurací.

Pro implementaci klienta byl zvolen operační systém Android, protože je na rozdíl od konkurence otevřenější, masově rozšířený a vývojářské nástroje, jsou volně dostupné na běžné operační systémy.

Aplikace je napsána v Javě s Android SDK (*Software Developlment Kit*) [19]. Operační systém Android je od základu navržen tak, aby byl přístupný pro co nejširší spektrum uživatelů i vývojářů. Je zde kladen velký důraz na jednoduchost použití aplikací a s tím související omezené možnosti jejich nastavování. Velká pozornost byla při návrhu operačního systému věnována také bezpečnosti. Aplikace musí mít uživatelovo povolení k přístupu k jakýmkoliv důležitým systémovým a hardwarovým funkcím a žádat o ně systém. Kdykoliv se aplikace pokusí provést něco, na co nedostala povolení, systém ji ukončí.

Další důležitou vlastností systému je práce s pamětí vycházející z ideologie Javy. Většina aplikací se nikdy úplně nevypne. Pouze v případě, že systém nemá dostatek volné paměti, začne ukládat déle nepoužívaná okna aplikací a odstraňovat je z operační paměti. V systému navíc na pozadí stále běží mnoho procesů přijímajících a odesílajících data přes internet, pokud je dostupný. To klade velké nároky na správu procesu a zhoršuje odezvu systému. Pro pohodlné používání tak je potřeba alespoň dvoujádrový procesor.

Nevýhodou otevřenosti platformy je nepřeberné množství verzí hardwaru a softwaru, na kterém by aplikace měla pracovat. Pro každou implementovanou funkci se tedy musí ověřit, zda je na daném zařízení a verzi operačního systému dostupná. Pro kritické funkce, bez kterých by aplikace nemohla fungovat, lze určit minimální požadavky na verzi operačního systému a hardwarových funkcí, které musí být splněny, aby šlo aplikaci nainstalovat. V tomto případě nemá smysl aplikaci instalovat na zařízení bez fotoaparátu, či možnosti připojení na internet.

Mobilní zařízení mají oproti stolním počítačům navíc mnoho omezení, která se promítají do návrhu operačního systému a vývoje aplikací pro něj. Nejvýraznějším rozdílem je malý displej. Nebudeme-li uvažovat tablety, které jsou na fotografování při procházce městem velice nepraktické, nemají mobilní zařízení zobrazovacího prostoru nazbyt, jako operační systémy klasických počítačů. Mobilní operační systémy jsou tedy většinou navrženy tak, aby celou obrazovku zabírala vždy jen jedna aplikace. Výhodou tohoto přístupu je, že všechny ostatní aplikace, na něž zrovna není vidět, mohou omezit svou činnost a snížit nároky na procesor.

Problém velikosti displeje zhoršuje způsob ovládání. Nástup kapacitních dis-

plejů přinesl sice efektní, ale poněkud nepřesné ovládání prsty, oproti přesnějšímu ovládání stylusem či nehtem u resistivních displejů. To vyžaduje velké ovládací prvky zabírající cenné místo.

Slabou stránkou mobilních zařízení je velice omezená výdrž baterie. Proto mobilní operační systémy používají celou řadu metod, jak omezovat spotřebu energie, kdekoliv je to možné. Operační systém Android ovšem upřednostňuje vývoj aplikací v Javě. Tím na jednu stranu výrazně snižuje nároky na vývojáře a vývoj aplikací. Na druhou stranu ovšem programy v jazyku Java běží na virtuálním stroji, což znamená plýtvání systémovými zdroji. Zařízení s Androidem tedy potřebují výkonnější hardware s vyšší spotřebou energie, aby dosáhly stejného výkonu jako konkurenční systémy s nativními aplikacemi.

Pro Android existuje také možnost vytvořit částečně nativní aplikaci v C++ pomocí Android NDK (*Native Development Kit*). Ale oficiální dokumentace ji doporučuje pouze pro výpočetně intenzivní části kódu protože jinak nepřináší žádné zvýšení výkonu a výrazně komplikují vývoj. Protože pro NDK neexistují jednotná rozhraní pro interakci s hardwarem a systémem jako v Javě, musí se k nim přistupovat buď přes Javu nebo způsobem specifickým pro každé zařízení. Protože zařízení, na kterém byla aplikace vyvíjena není příliš výkonné, nebylo do klienta implementováno žádné předzpracování obrázku.

Aplikace je cílena na zařízení s verzí operačního systému Android 2.3 (Gingerbread - API level 9) a vyšší. Je vyvíjena na zařízení se systémem ve verzi 4.0.4 (Ice Cream Sandwich - API level 15).

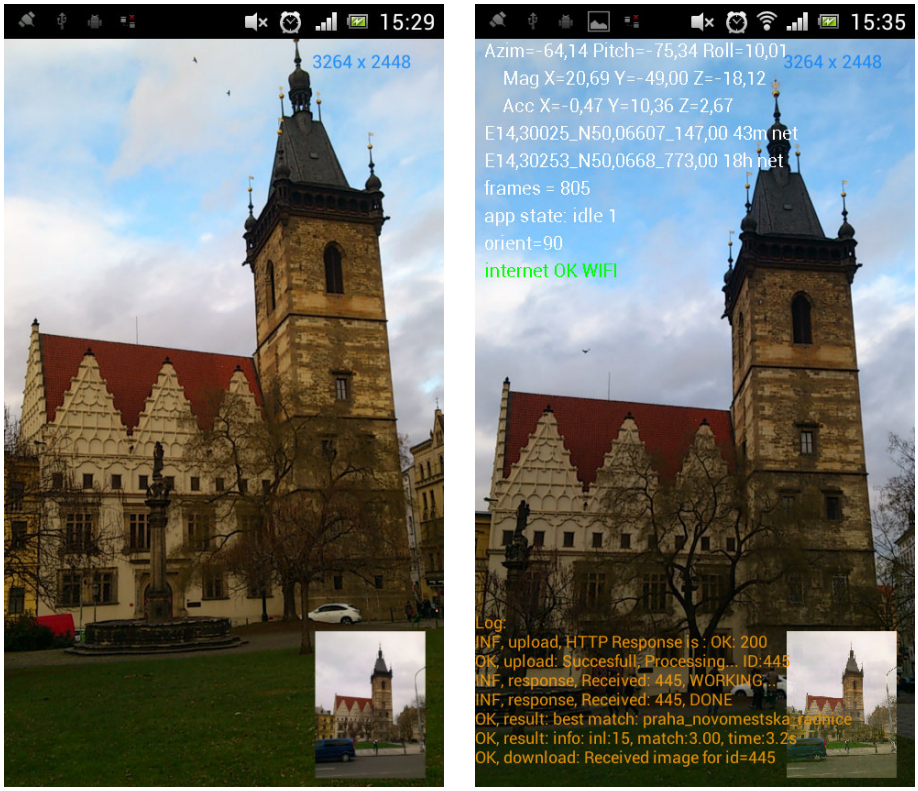
### 3.1.1. Grafické uživatelské rozhraní

Aplikace v této fázi není míněna jako finální produkt, ale jde spíše o prototyp na ověření možností rozpoznávání obrázků v terénu. Proto má aplikace pouze nezbytnou funkcionalitu bez většího důrazu na přívětivost uživatelského rozhraní.

Aplikace nepotřebuje příliš ovládacích prvků. V uživatelském režimu je jediným ovládacím prvkem nastavení velikosti pořizovaných fotografií (obr. 13a). Aplikace má tedy jednoduché uživatelské rozhraní s jedním oknem (Aktivitou v terminologii Android SDK). Jak je u fotografických aplikací zvykem, je uzamčena v portrait režimu (displej na výšku), protože jinak při otočení zařízení systém vyvolá její restart, což nějakou dobu trvá a uživatele obtěžuje. Aktivita slouží k pořizování fotografií i k prohlížení výsledků. Na celou obrazovku zařízení je neustále zobrazován náhled fotoaparátu a přes jsou zobrazeny informace o stavu aplikace a případně výsledky.

Rozhraní je tvořeno dvěma objektovými třídami. Hlavní třídou je aktivita **PreviewAc**, která je zároveň hlavní třídou aplikace. Ta obstarává všechny systémové požadavky a obsahuje všechny vizuální prvky. Na první vrstvě se zobrazuje náhled fotoaparátu. Přes náhled fotoaparátu je vrstva tvořena třídou **OverlaySurface**, ve které se zobrazují informace pro uživatele a navrchu jsou ovládací prvky.

**Testovací režim** V testovacím režimu se navíc přes obraz vypisují informace o stavu aplikace (obr. 13b). V horní části obrazovky se zobrazují aktuální informace o poloze, zrychlení, natočení telefonu, magnetickém poli a o aktuálním



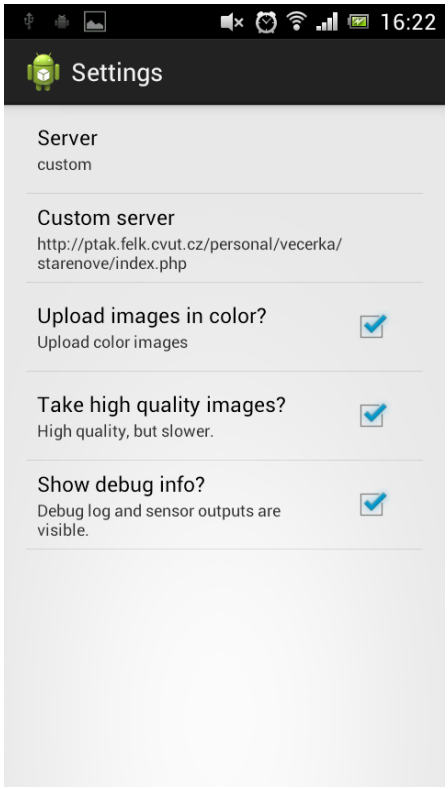

**(a)** Základní uživatelské rozhraní **(b)** Testovací režim

**Obrázek 13.** Vzhled uživatelského rozhraní. Pro oba režimy je společná systémová lišta nahoře, v pravém horním rohu nastavení rozlišení fotografií a zobrazení výsledku dole v rohu. V Testovacím režimu jsou navíc bíle výpisy dat ze senzorů a oranžově záznam událostí a barevně informace o stavu připojení k síti. Význam senzorových dat: Na první řádce je vypočtená orientace vzhledem ke světu (rotace kolem tří os, přibližně svislé, od západu k východu a od severu k jihu), Dále přímo údaje ze senzorů ze kterých je orientace vypočtená, tedy magnetometr a akcelerometr. Další řádky jsou zaměřené pozice, aktuálně nejlepší a nejnovější získaná. Následující řádek *frames* zobrazuje počet vykreslených snímků. Řádek *app state* ukazuje stav aplikace (*idle* znamená, že aplikace pouze čeká na uživatele. Na dalším řádku je uvedena orientace pořízené fotografie. A na posledním je dostupnost internetu.

stavu aplikace. V dolní části se pak zobrazuje záznam diagnostických zpráv, kde se vypisují chyby, výjimky a postup zpracování obrázku.

Navíc je v tomto režimu přístupná aktivita s podrobnějšími nastaveními aplikace (obr. 14. V této části je možné nastavit adresu serveru a způsob pořizování a odesílání fotografií na server.

**Pořízení snímku** Operační systém Android nabízí dvě možnosti pořízení fotografie. Jednodušší možností je použit tzv. *Intent* (záměr), tedy vyvolat základní systémovou fotografickou aplikaci, kde si uživatel pořídí fotografii a ta se pošle zpět aplikaci. Metoda je to jednoduchá na implementaci, ale velice omezená. Složitější na implementaci, ale příhodnější je napsat si vlastní obsluhu kamery pomocí API, protože to umožňuje maximální kontrolu nad pořízením snímku. S vlastní




**Obrázek 14.** Uživatelské rozhraní okna s nastaveními. První položka je výběr známého serveru. Druhá pak nastavení vlastního serveru a třetí umožňuje nastavit zda odstraňovat barevnou složku z fotografií získaných zachycením náhledu. U fotografií pořízených běžným způsobem to není možné. Další položka nastavuje zda pořizovat fotografie zachycením snímku náhledu nebo běžným způsobem. A Poslední nastavení povoluje testovací režim.

kontrolou kamery je navíc možné v aplikaci stále zobrazovat náhled fotoaparátu.

Na zařízeních, která to umožňují, tedy s verzí systému 4.0 a vyšší a zároveň s odpovídající hardwarovou podporou, lze dvěma prsty vybrat oblast, podle které se bude ostřit a nastavovat vyvážení barev obrazu. Bohužel jsou to nové funkce a ještě ne tak rozšířené. Na testovaném zařízení byla k dispozici pouze volitelná oblast ostření. Volitelná oblast pro určování vyvážení barev by mohla velice pomoci u tmavých budov.

Pořídit fotografii je možno dvěma způsoby. Ten rychlý a méně kvalitní využívá náhledu fotoaparátu. Systém si obstarává vykreslování náhledu na definovanou plochu aplikace sám, ale zároveň může posílat aplikaci, při každém překreslení, událost obsahující obrazová data náhledu. Základním formátem obrazových dat pro snímek náhledu je v Androidu NV21. V tom je obrázek reprezentován v YUV barevném modelu. V první části obrazových dat je pro každý pixel 1 B jasová složka. Po ní pak následuje blok prokládaných složek U a V, kde každá hodnota U a V určuje barvu $2 \times 2$ pixelů a zabírá 1 B. To velice snadno umožňuje vyextrahovat černobílý obrázek čehož se využívá v např. aplikacích rozšířené reality.

Náhledy fotoaparátu mají omezená rozlišení a kvalitu, aby příliš nezatěžovaly systém. I když je možné, že moderní zařízení s displeji s vysokým rozlišením budou

mít vyšší kvalitu náhledu. V rychlém režimu se tedy po uživatelově kliknutí zachytí nejbližší snímek náhledu, zkomprimuje se a odešle na server.

V pomalejším režimu se pořídí snímek lepší kvality. Po uživatelově kliknutí se spustí ostření na uživatelem vybrané oblasti a snímek se pořídí až v momentě, kdy fotoaparát dokončí ostření. V tomto režimu nejsou na většině zařízení dostupná nezpracovaná obrazová data, ale pouze již zkomprimovaný JPEG soubor, který se pak pouze odešle na server.

### 3.1.2. Informace o poloze

Naprostá většina, ne-li všechny mobilní telefony s operačním systémem Android, mají integrované senzory pro měření zrychlení, magnetického pole a polohy. Senzory pro měření zrychlení a magnetického pole využívá systém pro automatické otáčení obrazovky podle natočení zařízení.

Možnost zaměřit polohu je u všech mobilních zařízení vyžadována Federální Komunikační Komisí Spojených Států Amerických (FCC - *Federal Communications Commission*) pro případy volání na tísňové linky. Tyto informace, pokud jsou k dispozici, můžeme využít k významnému upřesnění výsledků a zjednodušení vyhledávání.

K dispozici jsou dva zdroje informace o poloze. První, méně přesný, vychází ze vzdálenosti od vysílačů mobilního GSM a z nalezených okolních Wi-Fi sítí. Ve městech, kde bývá mnoho vysílačů, bývá tato informace velice přesná a nestojí žádnou energii navíc. Pokud je potřeba přesnější zaměření, je k dispozici GPS, které dokáže zjistit polohu až s přesností na metry, ale má poměrně vysokou spotřebu, pokud je zapnuté dlouhodobě.

Na přítomnost těchto informací se ovšem nelze spolehnout, protože zařízení nemusí být připojeno, nebo může být mimo dosah mobilní sítě, GPS může být zakázáno, nebo při špatných podmínkách může mít problém se zaměřením dostatečného počtu satelitů.

Aplikace se proto pokouší získat informace o poloze z obou zdrojů a vybírá si ten aktuálnější a přesnější. Vzhledem k tomu, že aplikace nepotřebuje znát absolutně přesnou polohu a předpokládá se, že uživatel se pohybuje pěšky a tedy relativně pomalu, stačí aktualizovat informace o poloze pouze když se uživatel pohne o několik set metrů či uplyne několik minut od poslední kontroly polohy.

### 3.1.3. Komunikace

Veškerou komunikaci obstarává třída **Comm**, která je potomkem třídy **Thread** a komunikace tak probíhá asynchronně. Pokud je potřeba odeslat data na server, je vytvořena instance této třídy obsahující adresu serveru a požadovaná data. Komunikace probíhá přes třídu **HttpURLConnection**. Jejím podstatným nedostatkem je nemožnost sledovat průběh přenosu dat, ale zde přenášená data nejsou tak velká, aby to působilo významné problémy.

Pokud uživatel pořídí fotografii, vytvoří se instance této třídy s daty fotografie a spustí se vlákno. Nejdříve se odečtou všechny údaje senzorů, které se budou

| Popis | Typ | | Rozsah hodnot | Příklad | Proměnná |
|---|---|---|---|---|---|
| Orientace fotografie | int | ° | $\{0, 90, 180, 270\}$ | 90 | "o" |
| Zeměpisná délka | String | ° | $\{W, E\}$ $(-180, 180)$ | E14,41942 | "d" |
| Zeměpisná šířka | String | ° | $\{N, S\}$ $(-90, 90)$ | N50,07764 | "s" |
| Přesnost GPS | float | m | $\langle 0, \infty)$ | 20,00 | "c" |
| Azimut | float | ° | $(-180, 180)$ | 20,00 | "a" |
| Otočení podle osy Západ-Východ | float | ° | $(-180, 180)$ | 20,00 | "p" |
| Otočení podle osy Sever-Jih | float | ° | $(-180, 180)$ | 20,00 | "r" |
| ID zařízení | String | - | - | "802340c5" | "i" |
| Obrázek JPEG | binární data | - | - | - | "image" |

**Tabulka 1.** Tabulka formátu dat odesílaných na server. Sloupec proměnná je název proměnné v HTTP POST žádosti.

ID:362| Submited:2013-12-31 04:32:57| Status:5| Message:finishing job 362 on host cmpgrid-73| Host:cmpgrid-73| Data:/data/362.txt| Result:/results/362/362.txt| Last update:2013-12-31 04:33:02| finished:2013-12-31 04:33:02 Result:

RES:OK|
INF:inl:7, match:1.75, time:2.6s|
OBJ:praha_kamenny_zvon
< img src=thumbs/praha-kamenny-zvon.jpg>

**Tabulka 2.** Formát odpovědi serveru. Důležité položky jsou ID tedy identifikační číslo přiřazené dotazu. Status, stav zpracování dotazu (2-probíhá zpracování, 5-hotovo). Result což je výsledek zpracování dotazu a je tedy přítomen pouze když je zpracování dotazu dokončeno. A především OBJ, což je interní název nejlepšího nalezeného odpovídajícího objektu. Volitelně pak je součástí odpovědi odkaz na nalezený obrázek.

odesílat na server a pokud jsou obrazová data ve formě bitmapy, zkomprimuje se obrázek do formátu JPEG.

Data jsou na server odeslána jako HTTP Form ve formátu *"multipart/form-data"*, který umožňuje posílání proměnných a souborů zároveň metodou POST. Formát odesílaných dat je popsán v tab. 1. Očekávaný formát odpovědi je uveden v tab. 2.

## 3.2. Server

Veškeré zpracování a rozpoznávání obrazu je implementováno na serveru. Vzhledem k tomu že jde o prototyp, je celé implementováno v Matlabu. Implementace je rozdělena na dvě částečně se překrývající části. První částí je vytvoření vyhledávací struktury (databáze) a druhou vyhledávání v ní. Mají společné funkce na zpracování obrázků. Implementace v Matlabu má řadu výhod, ale také nevýhod.

Výhodou je značné množství již implementovaných komplikovaných funkcí a tedy urychlení vývoje. Nesmírnou výhodou při vývoji je interpretovaná povaha programů, takže lze při pozastavení vykonávání funkce snadno prohlížet, zobrazovat a měnit hodnoty. Nevýhodou tohoto řešení je jeho pomalost. Aby program zpracoval dotaz v horizontu sekund je nutné při psaní složitějších funkcí uvažovat pro běžné programovací zvyklosti velice netypickým způsobem a minimalizovat používání cyklů.

### 3.2.1. Rozhraní

Server má dva vstupní body. Prvním je skript pro vytvoření databáze a druhým funkce pro vyhledání obrázku.

**Vytvoření databáze** Skript pro vytvoření databáze `CreateDB.m` načte obrázky ze složky, vygeneruje jejich popisy a vytvoří z nich databázi, kterou uloží do souboru `database.mat` a nastavení, se kterým byla databáze vytvořena uloží do souboru `settings.mat`. Nastavení se specifikují na začátku skriptu (tab. 3 až tab. 6). Nejdůležitějším nastavením je cesta k adresáři, ve kterém jsou uloženy obrázky, ze kterých se má databáze vytvořit. Skript očekává cestu ke složce, která obsahuje jednu podsložku pro každý objekt, obsahující všechny fotografie objektu ve formátu JPEG a správně otočené a volitelně textový soubor `settings.txt` ve kterém jsou uloženy informace o objektu jako jeho poloha viz. tab. 7.

**Zpracování dotazu** Funkce `ProcessImg(file, groundTruth)` pro zpracování dotazu se nachází v souboru `ProcessImg.m`. Prvním parametrem je buď přímo obrázek, který se má vyhledat nebo textový soubor s informacemi ze zařízení, cestou k obrázku a cestou k souboru, do kterého se zapíší výsledky viz. tab. 8. Druhý parametr je volitelný a obsahuje interní jméno objektu, slouží pro testovací účely, aby funkce mohla vyhodnotit, zda našla správnou odpověď. Pokud se funkce spustí bez druhého parametru, pak po dokončení ukončí Matlab.

Funkce vyžaduje ve svém adresáři soubory `database.mat` a `settings.mat` vytvořené skriptem `CreateDB.m`. Průběh a podrobné výsledky vyhledávání funkce vypisuje na standardní výstup. Výsledky zpracování dotazu zapíše do zvláštního textového souboru (tab. 9) pokud byl specifikován.

### 3.2.2. Zpracování obrázku

Zde je uveden přehled základních funkcí pro zpracování obrázku. Načtení obrázku ze souboru, jeho převedení do 256 odstínů šedi, převzorkování na správné velikosti

| Struktura settings - obecná nastavení | | | |
|---|---|---|---|
| Pole | Výchozí hodnota | Popis | Povolené hodnoty |
| resolution | 800 | Rozlišení obrázku [px] | |
| small | 0,5 | Poměr velikosti obrázku pro výpočet SIFT deskriptorů | $(0, 1)$ |
| allowEqualization | 2 | Automatická úprava kontrastu obrázku | 0-zakázána<br>1-povolena<br>2-vynucena |
| detector | 'harris' | Použitý detektor bodů zájmu | 'harris'<br>'hessian'<br>'affine_harris' |
| maxPoints | 1500 | Počet nejlepších bodů zájmu, které se použijí | |
| showPoints | false | Zobrazit nalezené body zájmu | true<br>false |
| descriptor | 'sift' | Použitý deskriptor | 'SIFT'<br>'DCT' |
| rotationInvariant | true | Povolit rotační normalizaci | true<br>false |
| featureSize | 16+1 | velikost okolí pro výpočet deskriptoru [px] | 16+1 |
| halfSize | 9 | velikost poloviny okolí [px] | 9 |
| PCA | true | Použít PCA | true<br>false |
| PCAreduction | 32 | Počet kolik dimenzí redukovat | |
| verbosity | 1 | Výřečnost | 0-nic<br>1-kritické<br>2-vše |

**Tabulka 3.** Obecná nastavení zpracování obrázku.

| Struktura settings.harris - Nastavení Harrisova detektoru | | | |
|---|---|---|---|
| Pole | Výchozí hodnota | Popis | Povolené hodnoty |
| sigma1 | 1,0 | Rozptyl filtračního Gausiánu | |
| sigma2 | 1,0 | Rozptyl integračního Gausiánu | |
| threshold | 0 | Práh pro přijetí bodů zájmu | |
| center | false | Preferovat body zájmu blíže středu obrázku | true<br>false |

**Tabulka 4.** Nastavení Harrisova detektoru rohů.

| Struktura settings.hessian - Nastavení detektoru lokálních maxim Hessiánu | | | |
|---|---|---|---|
| Pole | Výchozí hodnota | Popis | Povolené hodnoty |
| sigma | 3,25 | Rozptyl filtračního Gausiánu | |
| threshold | 0,2 | Práh pro přijetí bodů zájmu | |

**Tabulka 5.** Nastavení detektoru lokálních maxim Hessiánu.

| Struktura settings.clusters - Nastavení shlukování | | | |
|---|---|---|---|
| Pole | Výchozí hodnota | Popis | Doporučené hodnoty |
| spliting | 3 | Počet velkých shluků v rekurzivní fázi | 3-10 |
| fragment | 14 | Průměrný počet prvků na finální shluk | 10-60 |
| Threshold | 5000 | Práh velikosti shluku pro finální shlukování | |

**Tabulka 6.** Nastavení hierarchické metody K-středů.

```
n Novoměstská radnice
d E14,41448
s N50,07293
```

**Tabulka 7.** Příklad formátu souboru s informacemi o objektu.

```
o 90
d E14,41448
s N50,07293
c 23,00 5s
a -125,73
p -81,34
r 159,78
i 802340c5
f D:\Pictures\341.jpeg
v D:\Results\341.txt
```

**Tabulka 8.** Příklad souboru s informacemi pro zpracování dotazu. Význam hodnot je stejný jako v tab. 1. Navíc jsou zde řádky *f* a *v*. První je soubor s obrázkem ke zpracování a druhý je soubor, do kterého se zapíší výsledky

```
RES:OK|
INF:inl:7, match:1.75, time:2.6s|
OBJ:praha_kamenny_zvon|
```

**Tabulka 9.** Příklad souboru s výsledky zpracování dotazu. První řádka říka zda se zpracování povedlo, druhá jak kvalitní je výsledek a jak dlouho trvalo zpracování dotazu a třetí je interní jméno nejpodobnějšího nalezeného objektu.

a vylepšení kontrastu řeší funkce:

`[img,imgSmall] = loadImage(filename, settings, enhance)`.

Parametry jsou cesta k souboru obrázku, struktura s nastaveními (tab. 3) a logická proměnná určující zda vylepšovat kontrast.

Výstupem pak jsou převzorkované a upravené verze obrázku. Výstupní parametr img je obrázek převzorkovaný tak, aby jeho delší strana měla velikost *settings.resolution*. Druhý výstupní obrázek je oproti prvnímu navíc zmenšený v poměru *settings.small*.

Detekci bodů zájmu, odhad jejich afinních transformaci a vytvoření výřezů jejich okolí zajišťuje funkce:

`[x, y, F, s, U] = Detect(img, imgSmall, settings)`.

Vstupními parametry jsou dva obrázky ve formátu výstupu z předchozí funkce a struktura s nastaveními detekce.

Výstupem jsou matice $1 \times n$ polohy a měřítka nalezených bodů (`x,y,s`), kde $n$ je počet nalezených bodů zájmu. Matice $featureSize \times featureSize \times n$ `F` výřezů okolí bodů zájmu z obrázku `imgSmall` a matice $2 \times 2 \times n$ `U` matice odhadů afinních transformací v bodech zájmu.

Popis bodů zájmu metodou SIFT a nalezení dominantního úhlu bodu zájmu provádí funkce:

`[desc, dom0] = describe(F, method, rotationInv)`.

Vstupním parametrem je matice `F` výřezů okolí bodů ve formátu výstupu předchozí funkce. Nastavení použité metody `method` je pouze pro kompatibilitu a není využito. Posledním parametrem je pak `rotationInv` určující zda oblasti rotačně normalizovat.

Výstupem je $128 \times n$ matice SIFT deskriptorů `desc` a $n \times 1$ matice nalezených dominantních úhlů `dom0`.

### 3.2.3. Komunikační rozhraní

Hlavním rozhraním serveru je skript **index.php**, který zajišťuje přijetí dotazu od klienta a s ním spojených dat a jejich uložení na disk a zapsání do databáze. Bez parametru stránka zobrazí formulář, do kterého je možné všechny hodnoty zadat manuálně a připojit obrázek ze souboru na disku. Význam hodnot ve formuláři je stejný jako v tab. 1 a jediné nezbytné položky jsou `Orientation` a nahrání souboru obrázku ve formátu `".jpg"` nebo `".png"`. S parametrem **index.php?id=i** pak skript zobrazí výsledky zpracování dotazu $i$ ve formě HTML stránky obsahující data specifikovaná v tab. 2. Skript **jobs.php** pak zobrazuje přehledem všech dosud přijatých dotazů a jejich výsledků. Přístup k serveru je chráněn jednoduchou HTTP autentizací s pevným heslem a uživatelským jménem.

Další částí systému je Perl skript, který si po přijetí dotazu přebere všechna získaná data a spustí Matlab funkci, která provede porovnávání obrázku. Když funkce skončí, skript uloží výsledky do souboru a označí v databázi úlohu za hotovou.

Testovací verze serveru se momentálně se nachází na školním serveru na adrese: `http://ptak.felk.cvut.cz/personal/vecerka/starenove`. Uživatelské jméno je `starenove` a heslo `praha`.

# 4. Experimenty

## 4.1. Testovací data

Protože hlavní funkcí k otestování je schopnost nalézt podobné současné obrázky, byly při několika příležitostech pořízeny stovky snímků několika pražských dominant. Fotografie, které nebyly vybrány pro vytvoření databáze (kap. 2.2.1), tvoří testovací sadu, na které se testovalo nejlepší nastavení parametrů pro vytvoření databáze a zpracování dotazu.

Hlavní "reálná"testovací sada sloužící k testování výsledků celého systému byla vytvořena z dalších fotografií pořízených nezávisle na fotografiích v databázi a sadě pro nastavení za odlišných podmínek (osvětlení, počasí, ...).

V testovací sadě jsou obsaženy fotografie 13-ti objektů. Patří sem Staroměstská a Novoměstská radnice, Prašná brána, chrám Matky Boží před Týnem a Dům U Kamenného zvonu, budova ČVUT na Karlově náměstí, budova Městského soudu na Karlově náměstí, obchodní dům Kotva, obchodní centrum Paladium, Obecní dům, budova Městské Knihovny, sousoší Františka Palackého, škola ve Vojtěšské ulici a kostel sv. Vojtěcha.

## 4.2. Hodnocení výsledků

Abychom mohli říct jaké je nejlepší nastavení mnoha parametrů databáze a vyhledávání, potřebujeme možnost kvalitativně porovnat výsledky při různých nastaveních. Pro obrázky v databázi $I = \{i_1, i_2, ..., i_n\}$ je výsledkem vyhledávání uspořádaná podmnožina obrázků

$$V = \{a_1, a_2, ...a_q\},$$

kde $q \leq n$ podle podobnosti s dotazem $p(i) \in \mathbb{R}^1$ tak, že

$$p(i_{a_1}) \geq p(i_{a_2}) \geq ... \geq p(i_{a_n}).$$

V databázi jsou obrázky rozděleny podle objektů, které zachycují, do $m$ tříd. Pro každý obrázek $i$ v databázi tedy známe jeho třídu $k(i) \in \{1, 2, ..., m\}$ a ohodnoceni zda jde o správnou odpověď

$$c(i,d) = \begin{cases} 0 & pokud\ k(i) \neq k(d) \\ 1 & pokud\ k(i) = k(d) \end{cases},$$

kde $k(d)$ je známé přiřazení obrázku dotazu do třídy. Relevantní výsledky jsou takové, pro které platí $k(i) = k(d)$. Používanými funkcemi pro zhodnocení výsledků

vyhledávání na anotovaných datech jsou přesnost (*Precision*), což je podíl počtu relevantních výsledků k celkovému počtu výsledků

$$P = \frac{\sum_{j=1}^{q} c(i_{a_j})}{q}.$$

Dále *Recall*, což je počet relevantních výsledků ku počtu relevantních položek v databázi. A různé variace tyto metody jako Průměrná přesnost (*Average precision - AP*) [18].

Při vyhledávání na manuálně anotované testovací sadě máme k dispozici správné výsledky a pro každý dotaz tak dostaneme deset nejlepších výsledků seřazených od nejlepšího k nejhoršímu a jejich ohodnocení zda jde o správnou nebo chybnou odpověď.

Pro zhodnocení kvality výsledků na testovací sadě jsou použity dvě metody. První metodou je průměrná procentuální úspěšnost správných odpovědí. Pro každý dotaz $d$ vytvoříme ohodnocení $\bar{f}_d = \{f_1, f_2, ..., f_q\}$ odpovědí takové, že

$$f_i = \begin{cases} 1 & \text{pokud } c(i,d) = 1 \text{ a zároveň } \forall j \in \{1,2,...,i-1\}(c(j,d) = 0) \\ 0 & jinak \end{cases},$$

pak výsledné ohodnocení bude pro všechny dotazy $d \in \{1, 2, ...n_d\}$, kde $n_d$ je počet dotazů vypadat takto

$$h = \frac{100}{n_d} \sum_{i=1}^{n_d} \bar{f}_i \quad h \in \mathbb{R}^q.$$

Tedy u kolika procent obrázků byla nejlepší správná odpověď na prvním, druhém, ..., desátém místě v pořadí a vypočteme střední hodnotu přes všechny objekty. Tím získáme srozumitelné hodnocení kvality výsledků (poslední sloupec tab. 10).

Druhá metoda je založená na předpokladu, že je lepší mít jeden správný výsledek na vyšším místě než všechny správné výsledky na nižších pozicích. Každá pozice v pořadí $i \in \{1, 2, ..., q\}$ má tedy přiřazenu váhu

$$w_i = \frac{2^{1-i}}{\sum_{j=1}^{q} 2^{1-j}},$$

normalizovanou tak, aby ohodnoceni spadalo do intervalu $(0, 1)$ a perfektní výsledek, tedy správná odpověď na všech pozicích měl ohodnocení 1. Pro každý dotaz se sečtou váhy pozic se správnými odpovědmi a střední hodnota těchto součtů přes všechny dotazy tvoří výsledné hodnocení (Poslední řádek posledního sloupce tab. 10).

K hrubému srovnání stačí první metoda. Respektive jenom její první hodnota, tedy u kolika procent dotazů byla správná odpověď na prvním místě. Při ladění parametrů se více hodí citlivější ukazatel, kterým je druhé hodnocení.

| Pořadí $i$ | Váha $w_i$ | dot. 1 | dot. 2 | dot. 3 | dot. 4 | Správnost | Hodnocení |
|---|---|---|---|---|---|---|---|
| 1. | 0.5 | **ANO** | **ANO** | NE | NE | 50% | **50%** |
| 2. | 0.25 | ANO | ANO | NE | NE | 50% | 0% |
| 3. | 0.125 | ANO | ANO | **ANO** | NE | 75% | **25%** |
| 4. | 0.062 | ANO | ANO | ANO | NE | 75% | 0% |
| 5. | 0.031 | ANO | NE | ANO | **ANO** | 75% | **25%** |
| 6. | 0.016 | ANO | NE | ANO | NE | 50% | 0% |
| 7. | 0.008 | ANO | NE | ANO | ANO | 75% | 0% |
| 8. | 0.004 | NE | NE | NE | ANO | 25% | 0% |
| 9. | 0.002 | NE | NE | NE | ANO | 25% | 0% |
| 10. | 0.001 | NE | NE | NE | NE | 0% | 0% |
| $\sum$ | 1 | 0.993 | 0.938 | 0.242 | 0.044 | 50% | **0.555** |

**Tabulka 10.** Příklad hodnocení výsledků čtyř dotazů pro počet výsledků $q = 10$. Ve sloupcích dot.1 je uvedeno zda byl na dané pozici nalezen obrázek odpovídajícího objektu či ne. Správnost udává v kolika procentech dotazů byla na dané pozici nalezena správná odpověď. V posledním sloupci (Hodnocení) je pak v kolika procentech dotazů byla na dané pozici nalezena první správná odpověď, tedy první hodnotící metoda. Poslední řádek ukazuje hodnocení druhou metodou pro každý dotaz a výslednou hodnotu v pravém dolním rohu. Hodnota odpovídá ohodnocení 50% první metody hodnocení a vypovídá navíc o tom jak vypadají výsledky dále v pořadí.

| Nastavení | Hodnota | Nastavení | Hodnota |
|---|---|---|---|
| resolution | 900 | descriptor | 'sift' |
| small | 0.5 | featureSize | 16+1 |
| allowEqualization | 2 | rotationInvarint | true |
| detector | 'harris' | PCA | true |
| maxPoints | 1500 | PCAreduction | 32 |
| harris.sigma1 | 1 | clusters.spliting | 5 |
| harris.sigma2 | 1 | clusters.fragment | 200 |
| harris.threshold | 0 | fragment.Threshold | 5000 |

**Tabulka 11.** Nastavení, při kterých bylo dosaženo uvedených výsledků. Význam hodnot viz. tab. 3–tab. 6

## 4.3. Výsledky

Testování rozpoznávání probíhalo především na stolním počítači se čtyřjádrovým procesorem Intel Core i5-2500k @ 3.3 GHz s 8 GB DDR3 RAM. Na této sestavě je průměrná doba potřebná na zpracování jednoho dotazu 4,5 s. Výsledky byly naměřeny s nastavením dle tab. 11.

V první fázi byly otestováno jak velký dopad má náhodnost přiřazení bodů zájmu do shluku. Při opakovaném vytvoření databáze ze stejné sady nalezených deskriptorů se ohodnocení výsledků klasifikace testovací sady pohybovaly v rozmezí 2% od střední hodnoty.

Žádný významný vliv na výsledky nemělo ani použití PCA. Opět byly rozdíly

v rozmezí 2% s použitím PCA i bez. Nicméně zpracovaní dotazů i databáze je při plné délce deskriptorů přibližně 3× pomalejší než s použitím PCA.

Jedna z metod vylepšení databáze, konkrétně hledání vizuálních slov, která se nevyskytují v obou dotazech [15], byla velice jednoduchá na implementaci a proto byla otestována, ale žádný viditelný dopad na úspěšnost vyhledávání nebyl pozorován.

Průměrná úspěšnost identifikace správného objektu na testovací sadě je 82% jak ukazuje tab. 12. Pro sadu, na které se testují nastavení je úspěšnost 92%, ale obrázky jsou zde bližší těm v databázi a svou roli zde hraje i přeučení.

Velký vliv má úspěšnost vyhledávání má nastavení počtu shluků. Pokud jsou shluky malé (menší než 100 prvků) dosahuje vyhledávání vyšší úspěšnosti, ale neposkytuje dostatek korespondencí pro kvalitní výpočet geometrických transformaci. Není tedy možné dobře výsledky ověřit. Naopak větší shluky (více než 100 prvků) umožňují vznik dostatečného množství korespondencí pro dobrý odhad transformace, ale zhoršují výsledky vyhledávání až o 10 procentních bodů. Na druhou stranu výsledky z větších shluků jsou lépe geometricky ověřené a tedy jistější. Malé shluky také podporují přílišnou přeučenost rozpoznávání na trénovací množinu.

Vzhledem k použití jednoduchého Harrisova detektoru bodů zájmu je úspěšnost poměrně citlivá na vzdálenost a úhel pohledu. Úspěšnost rozpoznávání rapidně klesá asi od 15° změny úhlu pohledu. To je do značné míry kompenzováno omezeným prostorem k fotografování v husté zástavbě a použitím většího počtu fotografií z mnoha různých míst k vytvoření databáze.

Vzhledem k náročnosti získání skutečných homografií mezi obrázky v testovací sadě byla kvalita získaných transformací posuzována pouze manuálně. Kvalitní homografii se podaří najít asi v 70% případů, ve kterých se podaří najít odpovídající obrázek. Většinou protože se nalezená homografie vypočítá z bodů, které neleží v jedné rovině.

Úspěšnost rozpoznávání je velice citlivá na osvětlení. Fotografie v databázi jsou většinou pořízené při zatažené obloze a tedy rozptýleném světle. Některé fotografie testovací sady jsou naproti tomu pořízeny v ostrém slunečním svitu a u těch je rozpoznávání znatelně horší. Ostré světlo zhorší úspěšnost identifikace objektu asi o 10 procentních bodů.

Mobilní aplikace byla zatím důsledně otestována pouze na mobilním telefonu Sony Ericsson Xperia Arc S, kde funguje podle předpokladů. Byla také krátce testována na mobilním telefonu Samsung Galaxy S2, kde podle všeho rovněž funguje. Podle prvních výsledků testování uživateli se zdá, že manuální nastavování místa ostření a vyvažování barev zbytečně komplikuje užívání aplikace.

Testování probíhalo bez použití informací o poloze mobilního zařízení protože bylo důležitější posoudit schopnost rozpoznat obrázky stejného objektu. Při použití informace o poloze by se rozsah vyhledávání zúžil na tolik, že by špatná odpověď v podstatě nebyla možná. Úspěšnost vyhledávání bez použití GPS by měla více vypovídat o úspěšnosti v případě rozšíření databáze.

| Objekt | počet v databázi | počet v testu | Správně | Úspěch | 2.místo | 3.místo |
|---|---|---|---|---|---|---|
| Týnský chrám | 17 | 17 | 16 | **94%** | 6% | 0% |
| Městská knihovna | 4 | 17 | 13 | **76%** | 6% | 6% |
| Prašná brána | 18 | 23 | 18 | **78%** | 17% | 4% |
| Městský soud | 6 | 25 | 22 | **88%** | 0% | 0% |
| kostel sv. Vojtěch | 18 | 11 | 10 | **91%** | 0% | 0% |
| ČVUT, Karlovo náměstí | 6 | 3 | 2 | **67%** | 0% | 0% |
| OD Kotva | 8 | 13 | 11 | **85%** | 8% | 0% |
| Obecní dům | 6 | 9 | 9 | **100%** | 0% | 0% |
| OC Paladium | 4 | 11 | 10 | **91%** | 9% | 0% |
| František Palacký | 16 | 3 | 1 | **33%** | 33% | 33% |
| škola Vojtěšská | 10 | 11 | 11 | **100%** | 0% | 0% |
| Staroměstská radnice | 12 | 10 | 9 | **90%** | 10% | 0% |
| Novoměstská radnice | 15 | 49 | 35 | **71%** | 2% | 2% |
| celkem | 150 | 202 | 167 | **82%** | | |

**Tabulka 12.** Průměrné výsledky rozpoznávání podle objektu. Průměrná úspěšnost pro všechny obrázky v testovací sadě je **82%** a podle druhé metody hodnocení **0.72**. Socha Františka Palackého dosahuje velice podprůměrného skóre protože obrázky v testovací sadě jsou pořízeny na ostrém slunci a v databázi jsou obrázky bez přímého slunce. U Novoměstské radnice je pak problémem všudypřítomná vegetace. která neumožňuje čistý pohled.

# 5. Závěr

Implementace značné části metod zpracování a rozpoznávání obrazu je dostupná jako knihovny, či zdrojové kódy. Jsou často poskytované samotnými autory pro testování jejich metod ostatními výzkumníky. Tyto implementace umožňují významně urychlit vývoj projektu, ale rozhodl jsem žádnou takovou nepoužít a použít místo nich implementace vlastní. Vzhledem k tomu, že jsou napsány v Matlabu, jsou pravděpodobně řádově pomalejší, a je možné, že obsahují těžko odhalitelné chyby, či špatné pochopení některých myšlenek metod a také stáli více práce a času. Věřím, ale že není lepší způsob pochopení a porozumění metodám než jejich vlastní implementace. Při vývoji, opravování chyb, testování a experimentování s parametry a implementací samotnou lze nejlépe pochopit jak metoda funguje. Další výhodou je volný přístup ke všem mezivýsledkům funkcí z nichž lze pochopit chování algoritmů v praxi. Nebylo tedy dosaženo špičkových výsledků, ale doufám, že v průběhu práce nabyté zkušenosti, zejména v tom jak věci nedělat, budou o mnoho cennější.

Byla vytvořena jednoduchá mobilní aplikace pro operační systém Android, která umožňuje uživateli vyfotografovat si stavbu a prohlédnout si její historickou podobu. Podle prvních pokusů se zdá, že odladění mobilní aplikace pro různá zařízení je větší problém než by obecná architektura systému Android naznačovala.

Testování ukázalo, že systém dokáže ve většině případů správně rozpoznat hledanou budovu z fotografie. Nicméně 80% úspěšnost má ještě mnoho prostoru pro zlepšení. Navíc je nutné počítat s tím, že datová sada použitá na testování má pouze velice omezený rozsah a může otestovat jenom zlomek skutečných možností, které mohou nastat při skutečném použití. Je tedy jisté, že úspěšnost ve skutečných podmínkách ještě klesne.

Současná implementace počítá s tím, že hledaná stavba bude zabírat velkou část fotografie a zmenšení fotografie na začátku procesu tak pouze potlačí šum v obraze a sníží výpočetní nároky. Kvůli tomu, ale nemůže být zařazena největší pražská dominanta, Pražský hrad. Nejčastěji se fotografuje z nábřeží a tedy z poměrně velké vzdálenosti, takže na fotografiích z mobilních zařízeních bez možnosti přiblížení zabírá pouze zlomek snímku. Řešením by mohlo být nastavení výřezu z obrazu, který uživatele zajímá.

Největší přínos pro vylepšení rozpoznávání by v současnosti asi přineslo vylepšení obsahu databáze. Zařadit více fotografií zachycujících více pohledů na objekty za různých odlišných podmínek. Nebyla například příležitost pro pořízení fotografií se zasněženými budovami protože v průběhu tvorby této práce, žádný sníh nebyl. Zkušenosti z tvorby projektu také umožňují pořizovat snímky, které lépe vystihují vybraný objekt pro použité metody zpracování obrazu.

Dalším poměrně rychlým krokem, na který už nezbyl čas, ale který by pravděpodobně výrazně vylepšil úspěšnost, by bylo použití rozšířeného dotazu [18].

Podstatným dalším krokem by měla být analýza stávajícího systému a jeho vylepšení a přepracování na základě zkušeností získaných při vývoji a testování, které už se nestačili zapracovat.

Možností dalšího zlepšování je pak nepřeberné množství. Značný přínos by měla mít detekce bodů zájmu ve více měřítcích či použití afinního detektoru bodů zájmu. Se zde implementovaným afinním detektorem dosahuje podle testování vyhledávání horších výsledků než s obyčejným detektorem. Je tedy možné, že implementace má některé nedostatky, které se nepodařilo včas odhalit. Správně fungující afinní detektor by měl tedy také přispět ke zlepšení funkčnosti systému. Další zlepšení by měla přinést vylepšení navrhovaná v kap. 2.2.5.

Než se aplikace dostane k běžným uživatelům je nutné přepracovat uživatelské rozhraní, aby bylo efektnější, přívětivější a poutavější.

Existuje spousta možností, kterým směrem se dále vydat při zlepšování aplikace. Jen tak chodit po městě, fotografovat budovy a doufat, že jedna z nich bude v databázi by asi bylo poněkud nudné. Proto by bylo zajímavým směrem dalšího vývoje pokusit se uživatele trochu více vtáhnout. Lidé si rádi hrají a proto by mohlo být zajímavé zobrazit přibližný "radar" zobrazující, že v okolí jsou stavby, které jsou v databázi. Ale pouze by dovedl uživatele do blízkého okolí a nechal ho hledat tu správnou budovu. S tím souvisí možnost zapojení v dnešní době tak populárních sociálních sítí, kde by uživatel mohl sdílet své objevené historické podoby budov.

Na to navazuje v současnosti také velice populární myšlenka rozšířené reality. Mobilní zařízení s dostatečným výkonem pro zpracování a rozpoznání obrazu se stávají stále dostupnější a proto by mohlo být zajímavé přesunout celé zpracování obrazu na mobilní zařízení a ze serveru pouze streamovat vizuální popisy objektů potřebné k rozpoznávání a historické obrázky budov v okolí. Uživatel by tak mohl jít po městě a v reálném čase by se mu přes rozpoznané domy vykreslovala jejich historická podoba.

Pokud by se v projektu mělo pokračovat je nevyhnutelné přepracování celého serverového systému do C++, kde by bylo možné ty nejnáročnější části výpočtu paralelizovat nebo přímo přesunout na grafickou kartu.

# Reference


[1] J. M. Lau. *Prague then and now*. Thunder Bay Press, 2006. ISBN: 978-1-59223-656-5.

[2] P. Scheufler. *Praha 1848-1914 : Hledání ztraceného města*. Baset, 2004. ISBN: 80-7340-056-1.

[3] P. Scheufler. *Fotohistorie*. 2013. URL: `http://www.scheufler.cz/cs-CZ/fotohistorie/fotoarchiv.html`.

[4] J. Sivic a A. Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: *Proceedings of International Conference on Computer Vision*. Sv. 2. 2003, s. 1470–1477.

[5] J. Matas et al. "Robust wide baseline stereo from maximally stable extremal regions". In: *Proceedings of British Machine Vision Conference*. 2002, s. 384–396.

[6] C. Harris a M. Stephens. "A combined corner and edge detector". In: *Proceedings of the 4th Alvey Vision Conference*. 1988, s. 147–151.

[7] K. Mikolajczyk et al. "A comparison of affine region detectors". In: *International Journal of Computer Vision* 65.1/2 (2005), s. 43–72.

[8] K. Mikolajczyk a C. Schmid. "An affine invariant interest point detector". In: *Proceedings of European Conference on Computer Vision*. 2002, s. 128–142.

[9] R. Lakemond, C. Fookes a S. Sridharan. "Affine adaptation of local image features using the hessian matrix". In: *Proceedings of International Conference on Advanced Video and Signal Based Surveillance*. 2009, s. 496–501.

[10] D.G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the International Conference on Computer Vision*. 1999, s. 1150–1157.

[11] Š. Obdržálek a J. Matas. "Local affine frames for image retrieval". In: *Proceedings of International Conference The Challenge of Image and Video Retrieval* (2002), s. 318–327.

[12] K. Pearson. "On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine* 2.6 (1901), s. 559–572.

[13] J. Philbin et al. "Object retrieval with large vocabularies and fast spatial matching". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[14] D. Nister a H. Stewenius. "Scalable recognition with a vocabulary tree". In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Sv. 2. 2006, s. 2161–2168.

[15] H. Jégou a O. Chum. "Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening". In: *Proceedings of European Conference on Computer Vision*. 2012, s. 774–787.

[16] P. Turcot a D.G. Lowe. "Better matching with fewer features: The selection of useful features in large database recognition problems". In: *Proceedings of International Conference on Computer Vision Workshops*. 2009, s. 2109–2116.

[17] M. A. Fischler a R. C. Bolles. "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the Association for Computing Machinery* 24.6 (1981), s. 381–395.

[18] O. Chum et al. "Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval". In: *Proceedings of the International Conference on Computer Vision*. 2007, s. 1–8.

[19] *Android API Guides*. 2013. URL: `https://developer.android.com/guide/index.html`.

# Zkratky

Preliminary text...

| | |
|---|---|
| GPS | Global Positioning System, navigační systém. |
| MSER | Maximally Stable Extremal Regions. |
| SIFT | Scale-Invariant Feature Transform. |
| PCA | Principal Component Analysis. Analýza hlavních komponent. |
| API | Application Programming Interface |
| GSM | Groupe Spécial Mobile |
| PHP | PHP: Hypertext Preprocessor |
| HTML | HyperText Markup Language |
| RANSAC | Random Sample Consensus |

# Příloha A.

# Obsah přiloženého CD

Přiložené CD obsahuje čtyři adresáře.

V adresáří `text` je text této práce ve formátu `PDF`.

V adresáří `server` se nachází zdrojové kódy serveru v Matlabu.

Ve složce `klient_src` se nachází zdrojové kódy aplikace pro mobilní zařízení jako projekt pro vývojové prostředí Eclipse.

Posledním adresářem je `klient`, kde se nachází instalační balíček mobilní aplikace v souboru `"SN.apk"`.