FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Rozpoznávání a editace urbanistické scény II.
Student:	Tatiana Popova
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Zpracováni obrazových dat ovlivněných osvětlením, počasím atp. je náročnou úlohou rozpoznávání vyžadující aplikaci znalosti machine learningu a data miningu.
Jednu z možností jak zlehčit tuto obtížnou úlohu je využít geolokačních dat (akcelerometr, gyroskop atp.). Cílem práce je zefektivnění procesu detekce scén (pomocí NN) a budov z pohledu přesnosti detekce i výkonu (např. paralelizace).
1) Proveďte rešerši tématu, včetně práce J. Šefčíka „Rozpoznávání a editace urbanistické scény I."
2) Analyzujte ji z hlediska návrhu výpočetní optimality a náročnosti. Navrhněte zlepšení.
3) Tyto analyzujte s ohledem na omezení projektu Věnných měst českých královen (již neexistující budovy).
4) Rozšiřte stávající databázi snímků. Navrhněte změny (přidání geoinf.) ve struktuře metadat.
5) Navrhněte a implementujte váš prototyp. Soustřeďte se na porovnání stávajícího řešení a vaším, podrobně porovnejte přesnost i rychlost.
6) Debatujte vhodnost a omezení prototypu pro projekt VMČK.

Elektronicky schválil/a doc. Ing. Jan Janoušek, Ph.D. dne 17. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rozpoznávání a editace urbanistické scény

Tatiana Popova

Katedra teoretické informatiky
Vedoucí práce: Ing. Radek Richtr, Ph.D.

13. května 2021

Poděkování

Ráda bych poděkovala Ing. Radku Richtrovi, Ph.D. za odborné vedení a konzultace po celou dobu mé práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 13. května 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Popova, Tatiana. *Rozpoznávání a editace urbanistické scény.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Registrace obrazu je velmi komplexní problém v rámci oboru strojového vidění. Takový problém pomáhají vyřešit moderní metody hlubokého učení. Tato práce se zabývá analýzou metod pro registraci obrazů a implementací prototypu na rozpoznávání budov v urbanistických scénách za komplikovaných podmínek osvětlení.

Pozice uživatele je získána z metadat vstupního snímku a pro následnou klasifikaci obrazu jsou použity lokální příznaky pomocí metody SuperPoint. Hledání shod mezi budovami v databázi a na snímku proběhlo pomocí metody SuperGlue. Součástí práce je rozšíření stávající databáze se snímky budov Prahy a jejich geolokačními daty. Dále je přesnost a rychlost vlastně vytvořeného prototypu otestována a porovnávána s již existujícím prototypem, ve kterém byl využit postup klasického strojového vidění. V závěru obsahuje práce debatování vhodnosti použití pro účely projektu Věnná města českých královen.

Klíčová slova Hluboké učení, SuperPoint, SuperGlue, geolokační data, lokální příznaky, rozpoznávání budov, urbanistické scény

Abstract

Image registration is a very complex problem within the field of machine vision. Modern methods of deep learning help to solve it. This work contains amalysis of the methods for image registration and prototype implementation for buildings' recognition in urban scenes using geolocation and image data.

The user's position is obtained from the metadata of the captured image, and local flags are used for the subsequent image classification using the SuperPoint method. The search for matches between the buildings in the database and the image was performed using the SuperGlue method. Part of the work is an extension of the existing database with images of buildings in Prague and their geolocation data. The accuracy of the method was tested and compared to the already existing prototype, which used classical machine vision. This work contains debates. This work discusses the suitability of use for project called Věnná města českých královen purposes.

Keywords Deep learning, SuperPoint, SuperGlue, geolocation data, local features, buildings recognition, urban scenes

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Hluboké učení je součástí oblasti strojového učení. Zařízení používá pro řešení problémů algoritmy hlubokého učení a architekturu neuronové sítě, která je podobná lidskému mozku.

Výhoda daného přístupu spočívá v lepších výsledcích, než udávají algoritmy klasického strojového učení, pokud je model hlubokého učení natrénován na velkém počtu dat.

Hluboké učení skvěle řeší problémy strojového vidění. Modernější modely mohou nejen rozpoznávat objekty na obrázcích, ale také hledat shody mezi objekty na dvojicích obrázků.

Tato práce je zaměřena na problém párování shodných snímků vnějších scén pomocí moderních algoritmů hlubokého učení, a to i za komplikovaných světelných podmínek.

Práce je součástí projektu Věnná města českých královen. Výsledkem projektu bude mobilní aplikace, která využívá prvky rozšířené reality za účelem zobrazení historických verzí skutečných budov v závislosti na počasí či denní době. Projekt je rozsáhlý a pracuje na něm více studentů, moje práce navazuje především na bakalářskou práci Jana Šefčíka, jejíž cílem bylo vytvoření prototypu nástroje pro rozpoznávání a editaci urbanistické scény a implementace modulu rozpoznávání obrazů.

Moje práce spočívá ve výzkumu způsobů pro rozpoznávání a editaci urbanistické scény na základě geolokačních a obrazových dat. Dalším cílem je do rozpoznaných scén umístit snímky budov z databáze patřící do této scény. Další součástí této práce je rozšíření a vylepšení stávající databáze.

Kapitola 1

Cíl práce

Cílem této práce je analyzovat vhodnost použití algoritmů hlubokého učení v rámci projektu Věnná města českých královen. Podle výsledků analýzy vytvořit prototyp nástroje v rámci projektu, který umožní rozpoznávat obsah urbanistické scény na základě geolokačních a obrazových dat. Cílem je do rozpoznaných scén přesně umístit snímky rozpoznaných budov, které jsou vybrány z databáze.

Prvním cílem této práce je analýza moderních algoritmů hlubokého učení pro hledání lokálních příznaků snímků vnějších scén. Dalším cílem je seznámení se s možnostmi párování lokálních příznaků mezi dvojice snímků a hledání shod mezi různými snímky stejné vnější scény.

Třetím cílem je návrh a implementace prototypu nástroje, který na základě algoritmu hlubokého učení a geolokačních dat rozpozná urbanistickou scénu a umístí do ní fotografie budov na místo rozpoznaných budov. Čtvrtým cílem je testování výsledného řešení a jeho porovnání s již existujícím řešením podle rychlosti a přesnosti. Pátým cílem je rozšíření a vylepšení stávající databáze. Výsledky této práce budou přínosné pro projekt Věnná města českých královen, na kterém spolupracují studenti i pedagogové z Fakulty informačních technologií na ČVUT v Praze s historiky z Hradce Králové a Historického ústavu Akademie věd České republiky.

Kapitola 2

Rešerše

Tato kapitola obsahuje popis postupů a algoritmů pro rozpoznávání urbanistické scény, úvod do lokálních příznaků obrazů a popis různých druhů geolokáčních dat.

2.1 Registrace obrazu

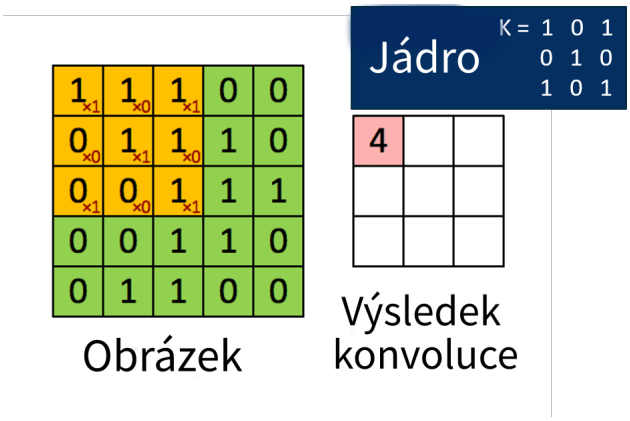
Registrace obrazu[1] je proces transformace různých obrazů jedné scény do stejného systému souřadnic. Tyto snímky mohou být vytvořeny v různých časech (multi-temporální registrace), různými senzory (multi-modální registrace), anebo z různých úhlů pohledu. Prostorové vztahy mezi těmito obrazy mohou být rigidní (například rotace), afinní (například škálování, posun nebo rotace), nebo množinou různých deformací.

Od počátku nultých let proces registrace obrazů většinou využíval tradiční přístupy založené na lokálních příznacích. Lokální příznaky nebo zajímavé body odkazují na vzor nebo odlišnou strukturu nalezenou v obrázku, například roh, hranu nebo malé políčko ze snímku. Obvykle jsou spojeny s částí obrazu, která se od svého bezprostředního okolí liší strukturou, barvou nebo intenzitou. Není podstatné, co tento příznak ve skutečnosti představuje. Důležité je především to, že je odlišný od svého okolí.

Tyto tradiční přístupy jsou založeny na čtyřech krocích:

1. *Detekce příznaků.* Nejvýznamnější charakteristické objekty (hrany, obrysy, průsečíky, rohy atd.) jsou detekovány ručně, nebo (pokud možno) automaticky. Pro další zpracování mohou být tyto prvky reprezentovány svými zástupci bodů (těžiště, konce čar, charakteristické body), které se v nazývají klíčové body.

2. *Porovnávání příznaků.* V tomto kroku se hledají shody mezi prvky detekovanými ve snímaném obrazu a těmi, které byly detekovány v referenčním snímku. K tomuto účelu se používají různé deskriptory příznaků.



Obrázek 2.1: Ukázka operace konvoluce

3. *Odhad transformačního modelu.* Typ a parametry takzvaných mapovacích funkcí, které srovnávají snímaný obraz s referenčním obrazem, jsou odhadnuty. Parametry mapovacích funkcí se počítají pomocí korespondence zavedených příznaků.

4. *Převzorkování a transformace obrazu.* Snímaný obraz je transformován pomocí funkcí mapování. Hodnoty obrazu v necelých souřadnicích se počítají pomocí vhodné interpolační techniky.

2.2 Konvoluční neuronové sítě

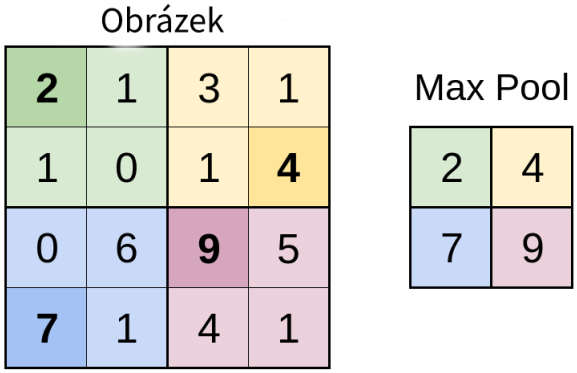
V současné době se dá velké množství problémů klasického strojového vidění řešit pomocí konvolučních neuronových sítí [2] s různou mírou úspěchu.

Vstup

Konvoluční neuronová síť jako vstup přijímá obrázek. Ten je reprezentován jako $h \times w \times d$ pole hodnot pixelů, kde h - výška (v pixelech), w - šířka (v pixelech), d - počet kanálů. Provést nějaké výpočty pomocí základních modelů neuronových sítí, kde každý neuron jedné vrstvy je spojen se všemi ostatními neurony další vrstvy, není možné – je to výpočetně náročné. Cílem konvolučních neuronových sítí je redukování dimenze snímku bez ztráty užitečných příznaků.

Konvoluční vrstva

Konvoluční vrstva je hlavní vrstvou v konvolučních neuronových sítích, vykonává operaci konvoluce. Pro tuto operaci je definované jádro, které je maticí velikosti $f_h \times f_w \times d$. Každá konvoluční vrstva postupně aplikuje vlastní jádro pro všechny výstupy předchozí úrovně.



Obrázek 2.2: Max pool vrstva

Na obrázku 2.1 je ukázáno, jak vypadá operace konvoluce s jádrem velikosti $3 \times 3 \times 1$.

Jádrem postupně prochází celý vstup a násobí hodnoty pixelů s hodnotami jádra. Tyto výsledky sečte a výsledkem bude jedno číslo. Výstupem konvoluční vrstvy je pole velikosti $(h - f_h + 1) \times (w - f_w + 1) \times 1$. Díky takovému postupu dojde ke zmenšení obrazu.

Pro konvoluční vrstvu je také definován krok, který určuje, o kolik pixelů se jádro přesune na vstupním obrázku. Například pokud krok je 1, pak se jádro přesune o 1 pixel po každém kroku.

Ne vždy má jádro vhodnou velikost pro vstup. V takovém případě je možné obklopit vstupní matici nulami (zero-padding) nebo oříznout jádro (valid-padding).

Pool vrstva

Hlavním cílem této vrstvy je zmenšení dimenze vstupní matice a zmenšení počtu parametrů modelu. Pool vrstva kontroluje přeučení. Má definované jádro velikosti $d \times w$ a krok s, o který se podoblast posouvá vůči vstupní matici. Hodnota kroku je nejčastěji šířkou jádra.

Max pool vrstva

Nejčastěji používaný druh pool vrstvy je max pool. Vrstva vybere maximální hodnotou dané podoblasti a zapíše ji na výstup, poté se posune dále.

Na obrázku 2.2 je ukázáno, jak funguje max pool vrstva pro vstup 4×4, jádro velikosti 2×2 a krok s $= 2$. Každé číslo první matice je hodnota pixelu na příslušné pozici. Čtverec velikosti 2×2 jedné barvy je část obrázku, pro kterou byl použit jeden krok operace max pool. Tučným písmem jsou označené výsledky operace.

Fully connected vrstva

Je její chování a architektura podobné skrytým (vnitřním) vrstvám neuronových sítí. Tato vrstva obsahuje neurony, z nichž každý je propojen se všemi neurony předchozí vrstvy. Na vstupu do fully connected vrstvy je matice transponována do vertikálního pole.

Aktivační funkce

Je potřeba, aby hodnoty pixelů nikdy nebyly zápornými. Proto se v konvolučních neuronových sítích používají aktivační funkce.

ReLu funkce

Nejpoužívanějším druhem aktivačních funkcí je ReLU (Rectified linear unit) [3].

Rectified linear unit (ReLU) funkce:

$$f(x) := max(0, x) \ , \tag{2.1}$$

kde x je aktuální hodnota pixelu.

Softmax funkce

Funkce softmax [4], známá také jako softargmax nebo normalizovaná exponenciální funkce, je zobecněním logistické funkce na více dimenzí. Často se používá jako poslední aktivační funkce neuronové sítě k normalizaci výstupu sítě.

Softmax funkce:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \ , \tag{2.2}$$

kde x je vstupní vektor, $\exp(x_i)$ je exponenciální funkce pro vstupní vektor, $\exp(x_j)$ je exponenciální funkce pro výstupní vektor.

Ztrátová funkce

Stroje se učí pomocí ztrátové funkce (angl. loss function) [5]. Jedná se o metodu hodnocení, jak dobře konkrétní algoritmus produkuje uvedená data. Pokud by se předpovědi příliš lišily od skutečných výsledků, výsledkem ztrátové funkce by měla být vysoká hodnota. Postupně se pomocí některé optimalizace ztrátová funkce učí snižovat chybu v predikci.

Algoritmy ve strojovém učení nemají žádnou univerzální ztrátovou funkci. Při výběru ztrátové funkce pro konkrétní problém existují různé faktory, jako je například zvolený typ algoritmu strojového učení.

Obecně lze ztrátové funkce rozdělit do dvou hlavních kategorií v závislosti na typu úkolu, kterým se zabýváme – regresní ztráty a ztráty klasifikace. Při klasifikaci se snažíme předvídat výstup ze sady konečných kategoriálních hodnot, tj. vzhledem k velké datové sadě obrazů ručně psaných číslic, jejich kategorizaci do jedné z 0–9 číslic. Regrese se naproti tomu zabývá predikcí spojité hodnoty například dané podlahové plochy, počtu pokojů, velikosti pokojů, predikce ceny pokoje.

Ztrátová funkce regrese

- *Střední kvadratická chyba (Mean Square Error/Quadratic Loss/L2 Loss):*

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \ , \tag{2.3}$$

  se měří jako průměr kvadratického rozdílu mezi předpovědí a skutečným výsledkem. Jde pouze o průměrnou velikost chyby bez ohledu na její směr. Avšak kvůli kvadratuře jsou předpovědi, které jsou daleko od skutečných hodnot, silně penalizovány ve srovnání s méně odchýlenými předpověďmi.

- *Střední absolutní chyba (Mean Absolute Error/L1 Loss):*

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \ , \tag{2.4}$$

  Průměrná absolutní chyba se naproti tomu měří jako průměr součtu absolutních rozdílů mezi predikcemi a skutečnými výsledky. Stejně jako MSE i toto měří velikost chyby bez ohledu na její směr. Na rozdíl od MSE potřebuje MAE k výpočtu přechodů složitější nástroje, jako je lineární programování. MAE je navíc odolnější vůči odlehlým hodnotám, protože nepoužívá čtverec.

Ztrátová funkce klasifikace

- *Křížová entropie:*

$$CrossEntropyLoss = -\left(y_i \cdot log\left(\hat{y}_i\right) + (1 - y_i) \cdot log\left(1 - \hat{y}_i\right)\right) \ . \tag{2.5}$$

  Toto je nejběžnější nastavení problémů s klasifikací. Ztráta napříč entropií se zvyšuje, pokud se predikovaná pravděpodobnost odchyluje od skutečného štítku.

Batch normalizace

Učení hlubokých neuronových sítí je obtížný úkol, jehož řešení zahrnuje několik problémů. Ačkoliv mají obrovský potenciál, mohou být náchylné k přeučení.

Batch normalizace (zkráceně Batch Norm) [6] je široce používaná technika v oblasti hlubokého učení. Zlepšuje rychlost učení neuronových sítí a pomáhá vyhnout se přeučení.

Pro pochopení Batch normalizace je nutné uvést, co je to normalizace.

Normalizace je technika předzpracování používána ke standardizaci dat. Jinými slovy převádí různé zdroje dat do stejného rozsahu. Existují 2 metody, jak provádět normalizaci:

$$x_{norm} = \frac{x - m}{x_{max} - x_{min}} \ , \tag{2.6}$$

kde x je hodnota, kterou je nutné normalizovat, m je výběrový průměr hodnot ze souboru dat, který je nutné normalizovat, x_{min} je minimální hodnota ze souboru těchto dat, x_{max} je maximální hodnota ze souboru dat. Použití tohoto vzorku normalizuje hodnoty dat do rozsahu od 0 do 1.

$$x_{norm} = \frac{x - m}{s} \ , \tag{2.7}$$

kde x je hodnota, kterou je nutně normalizovat, m je výběrový průměr hodnot ze souboru dat, který je nutně normalizovat, s je směrodatná odchylka této množiny dat. Výsledná proměnná má tedy nulový průměr a rozptyl roven jedné.

Batch Norm je normalizační technika prováděna mezi vrstvami neuronové sítě. Dělá se to v malých podmnožinách dat místo plné datové sady. Slouží pro urychlení učení. Vzorec pro Batch normalizaci je uveden níže.

$$z^N = \left(\frac{z - m_z}{s_z} \right) \ , \tag{2.8}$$

kde m_z je výběrový průměr hodnot z výstupů neuronů, s_z je směrodatná odchylka výstupů neuronů.

2.3 Extrahování příznaků

Lokální příznaky

Lokální příznaky[7] odkazují na opakující se vzorec nebo odlišnou strukturu v obrázku, mohou být bodem, hranou nebo políčkem obrázku. Obvykle jsou lokální příznaky spojeny s tou částí obrázku, která se od svého bezprostředního okolí liší strukturou, barvou, nebo intenzitou. Nezáleží na tom, co tato vlastnost ve skutečnosti představuje. Důležité je pouze to, že je odlišná od svého okolí. Mezi lokální příznaky patří například rohy a hranové pixely.

Deskriptor lokálních příznaků

Deskriptory lokálních příznaků [7] kódují zajímavé informace do pole čísel a fungují jako jakýsi číselný „otisk prstu“, který lze použít k odlišení jednoho lokálního příznaku od druhého. V ideálním případě by tato informace byla při transformaci obrazu neměnná, takže můžeme funkci znovu najít, i když je obraz nějakým způsobem transformován.

Vlastnosti ideálního deskriptoru lokálních příznaků:

- *Opakovatelný:* Je invariantní vůči rotaci, měřítku, fotometrickým variacím.
- *Výrazný:* Deskriptor je potřeba přiřadit k mnoha obrázkům / objektům.
- *Kompaktní:* Mělo by zachytit bohaté informace, ale nemělo by to být vysoce dimenzionální
- *Efektivní:* Možnost počítání v téměř reálném čase.

2.3.1 SIFT (Scale-invariant feature transform)

V roce 1999 David Lowe vyvinul nový algoritmus Scale Invariant Feature Transform (SIFT)[8], který extrahuje klíčové body a vypočítává jeho deskriptory.

Scale-space peak Selection

Měřítkovým prostorem (scale-space) obrazu je funkce $L(x, y, \sigma)$. Je vytvořena konvolucí gaussovského jádra (blurring) v různých měřítkách obrazu. Měřítko je rozděleno na oktávy, počet oktáv a měřítko závisí na velikosti původního obrázku. Generujeme tedy několik oktáv původního obrazu. Velikost obrázku každé oktávy je poloviční oproti předchozímu.

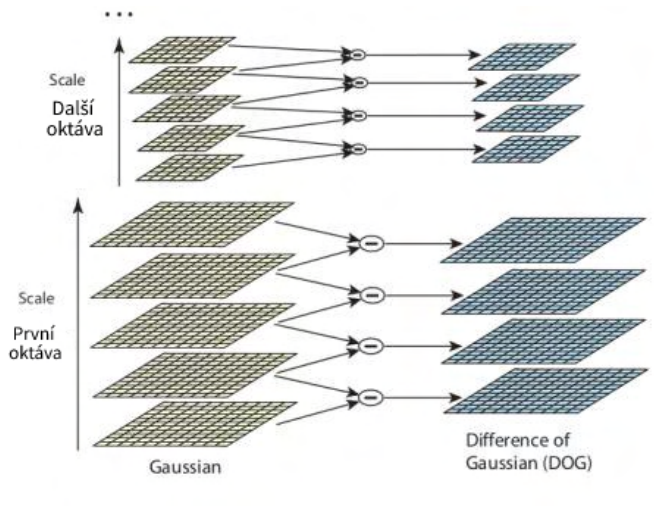
V rámci oktávy jsou obrázky postupně rozmazány pomocí operátoru Gaussian Blur. Matematicky se „blur“ označuje jako konvoluce Gaussova operátoru a obrazu. Gaussian Blur má konkrétní výraz nebo „operátor“, který se aplikuje na každý pixel. Výsledkem je rozmazaný obraz:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \ , \tag{2.9}$$

kde G je operátor Gaussian Blur a I je obraz. Zatímco x, y jsou souřadnice pixelu a σ je parametr „scale“:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \ . \tag{2.10}$$

Tyto rozmazané obrázky jsou používané ke generování další sady obrázků - Difference of Gaussian (DoG). Tyto obrázky DoG jsou skvělé pro vyhledávání



Obrázek 2.3: Difference of Gaussian

zajímavých klíčových bodů v obrázku. Difference of Gaussian se získá jako rozdíl Gaussian Blur obrazu se dvěma různými σ, ať už σ, nebo $k\sigma$. Tento proces se provádí pro různé oktávy obrazu v Gaussově pyramidě. Je to znázorněno na obrázku 2.3.

Dosazené výsledky se pak používají k výpočtu Laplacian of Gaussian aproximací, které jsou v měřítku neměnné.

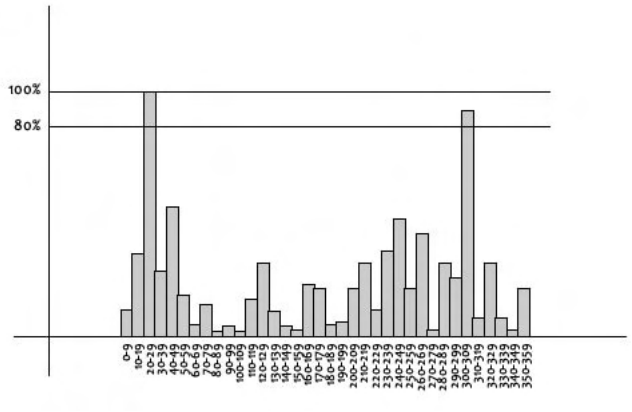
Jeden pixel v obraze je srovnáván s 8 sousedy, stejně jako 9 pixelů v dalším měřítku a 9 pixelů v předchozích měřítkách. Tímto způsobem se provede celkem 26 kontrol. Pokud se jedná o lokální extrém, je to potenciální klíčový bod. V podstatě to znamená, že klíčový bod je v tomto měřítku nejlépe zastoupen.

Lokalizace klíčových bodů

Potenciální klíčové body generované v předchozím kroku vytvářejí mnoho klíčových bodů. Některé z nich leží podél okraje, nebo nemají dostatečný kontrast. V obou případech nejsou dost užitečné, aby mohly být lokálními příznaky, takže se jich zbavíme. U bodů s nízkým kontrastem jednoduše zkontrolujeme jejich intenzitu. DoG má vyšší odezvu na hrany, takže některé hrany je také třeba odstranit. Pro odstranění hran a dalších chybných bodů se používá 2×2 Hessova matice H:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad . \tag{2.11}$$

Přidělení orientace



Obrázek 2.4: Orientační histogram

Po předchozím kroku jsou pouze legitimní klíčové body. Měřítko, ve kterém byl klíčový bod detekován, již známe (je stejné jako měřítko rozmazaného obrazu). To znamená, že jsou klíčové body invariantní vůči měřítku (scale invariance). Dále je potřeba přiřadit orientaci ke každému klíčovému bodu, aby byla invariance rotace.

Okolí umístění klíčového bodu se provede v závislosti na měřítku a v této oblasti se vypočítá velikost a směr přechodu. Vytvoří se orientační histogram z 36 kategorií pokrývajícími 360 stupňů, jak je ukázáno na obrázku 2.4. Je vybrán nejvyšší vrchol histogramu, který vytvoří klíčový bod na stejném místě a se stejným měřítkem, ale s upraveným směrem/orientací. Přispívá to ke stabilitě shody.

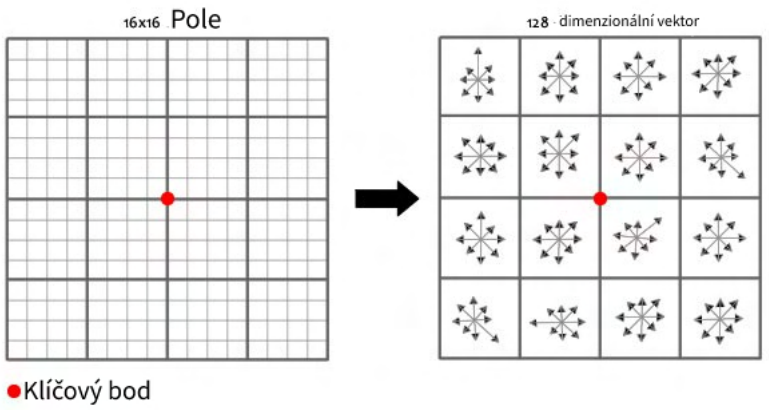
Deskriptor klíčových bodů

Je třeba vypočítat deskriptor pro lokální oblast kolem každého klíčového bodu, který je vysoce charakteristický a co nejvíce invariantní pro variace, jako jsou změny v pohledu a osvětlení. K tomu se provede okno 16×16 kolem klíčového bodu. Je rozděleno do 16 dílčích bloků velikosti 4×4 pixelů.

Pro každý blok je vytvořen histogram s orientací na 8 kategorií.

Deskriptor klíčových bodů je reprezentován jako vektor příznaků, což lze vidět na obrázku 2.5. Tento vektor příznaků přináší několik komplikací, které je rovněž potřeba vyřešit:

1. *Závislost rotace.* Vektor příznaků používá orientaci gradientu. Je zřejmé, že pokud obrázek otočíte, vše se změní. Změní se také všechny orientace gradientu. K dosažení nezávislosti rotace se rotace klíčového bodu odečte od každé orientace. Každá orientace gradientu je tedy relativní k orientaci klíčového bodu.



Obrázek 2.5: Vektor příznaků

2. *Závislost na osvětlení.* Pokud nastavíme prahová čísla, která jsou vysoká, můžeme dosáhnout nezávislosti na osvětlení. Takže jakékoli číslo (ze 128) větší než 0,2 se změní na 0,2. Tento výsledný vektor funkcí se znovu normalizuje. Po provedení této operace je vektor příznaků nezávislý na osvětlení.

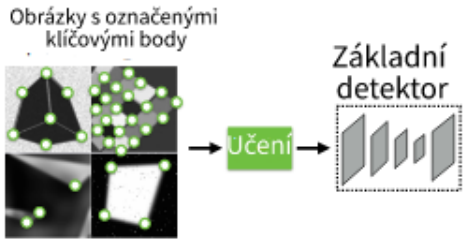
Detekce a extrahování klíčových bodů pomocí metod hlubokého učení

Tradiční detektory příznaků lze nahradit konvoluční neuronovou sítí (CNN), protože CNN mají silnou schopnost extrahovat složité příznaky, které vyjadřují obraz mnohem podrobněji, naučí se specifické vlastnosti úkolu a jsou mnohem efektivnější.

2.3.2 SuperPoint

Autoři článku [9] v roce 2018 vyvinuli self-supervised model pro extrahování klíčových bodů a výpočet jejich deskriptorů. Model je self-supervised, což znamená, že pro neoznačenou trénovací množinu jsou automaticky generované výstupy (ang. pseudo-labels) a dále je řešení považováno za učení s učitelem. Pro začátek je potřeba vygenerovat velkou množinu trénovacích dat a pseudo-ground truth výstupy, které budou obsahovat umístění pseudo-ground truth klíčových bodů v reálných obrazech.

„Ground truth“ je pojem používaný k označení informací poskytovaných přímým pozorováním, na rozdíl od informací poskytovaných odvozením. V daném případě ground truth je binární obrázek stejné velikosti jako vstupní obrázek, s 1 na pozicích umístění klíčového bodu a s 0 všude jinde.



Obrázek 2.6: Učení MagicPoint

Syntetic Shapes

Pro vytvoření pseudo-ground truth klíčových bodů nejprve byla naučena plně konvoluční neuronová síť na milionech příkladů ze syntetické datové sady nazvané Synthetic Shapes, kterou vytvořili autoři daného modelu. Synthetic Shapes se skládá z jednoduchých 2D tvarů, například z obrázků s čtyřúhelníky, trojúhelníky, čarami a elipsami. Jakmile jsou syntetické obrázky vykresleny, je na každý obrázek aplikována transformace pomocí homografie pro zvýšení počtu příkladů trénovacích dat. Zatímco typy zájmových bodů představované v Synthetic Shapes představují pouze podmnožinu všech zájmových bodů, mohou se vyskytovat na reálných snímcích, potom lze zajímavé body ze Syntetic Shapes použit k učení základního detektoru klíčových bodů, což je uvedeno na obrázku 2.6. Výsledný základní detektor se nazývá MagicPoint.

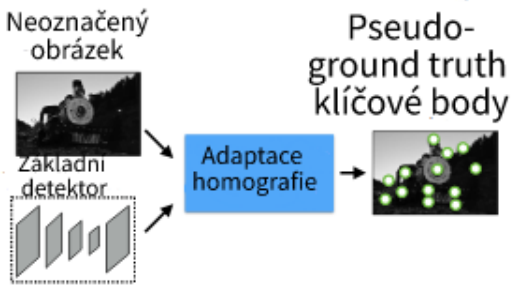
Adaptace homografie

Základní detektor MagicPoint, který byl generován v předchozím kroku, funguje překvapivě dobře na jednoduchých obrazech bez složitých textur, ale na reálných datech však detektoru MagicPoint chybí mnoho potenciálních klíčových bodů. Pro vylepšování detektoru MagicPoint autoři článku vyvinuli multi-scale multi-transform techniku - adaptace homografie.

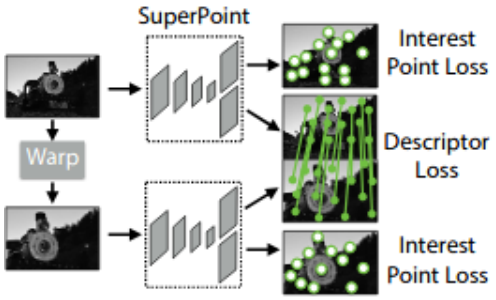
Homografie poskytují přesné, nebo téměř přesné transformace obrazu na obraz pro pohyb kamery pouze s rotací kolem středu kamery, scény s velkou vzdáleností od objektů a rovinné scény. Navíc, protože většina světa je přiměřeně rovinná, homografie dobře ukazuje, co se stane, když je stejný 3D bod viděn z různých úhlů pohledu. Protože homografie nevyžadují 3D informace, mohou být náhodně vzorkovány a snadno aplikovány na jakýkoli 2D obraz.

Výsledná funkce pro SuperPoint detektor je:

$$\hat{F}(I; f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} f_\theta(\mathcal{H}_i(I)) \ , \tag{2.12}$$



Obrázek 2.7: Adaptace homografie



Obrázek 2.8: SuperPoint descriptor

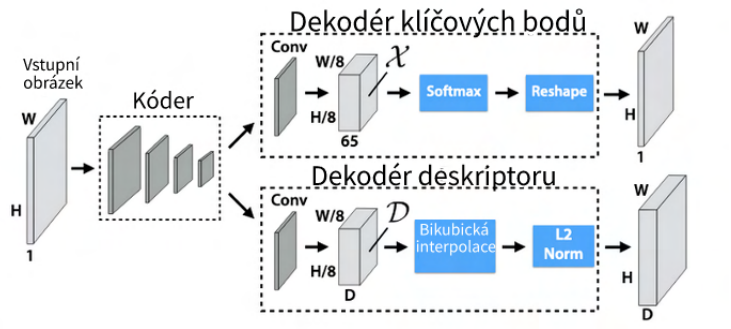
kde I je vstupní obrázek, f_θ je počáteční funkce klíčových bodů, kterou je potřeba zlepšit, $\mathcal{H}$ je náhodná homografie a $\hat{F}(\cdot)$ je výsledný detektor SuperPoint. Počet deformací homografií N_h je hyperparametrem daného postupu. V průběhu experimentů byl parametr $N_h = 100$ považován za optimální.

Technika adaptace homografie je použita pro generalizaci základní architektury MagicPoint na reálných obrázcích, což lze vidět na obrázku 2.7. Tento proces lze opakovat iterativně, aby na sebe neustále dohlížel a vylepšoval detektor klíčových bodů.

Výsledný detektor je autory pojmenován SuperPoint. Nejčastějším krokem po detekci robustních a opakovatelných zájmových bodů je tedy připojení pevného vektoru dimenzionálního deskriptoru ke každému bodu pro sémantické úkoly na vyšší úrovni, např. porovnávání obrázků. Konečně je tedy SuperPoint kombinován s podsítí deskriptoru, jak je ukázáno na obrázku 2.8.

Architektura modelu

Jako vstup model přijímá obrázek velikosti $H \times W$ v odstínech šedi. Tento obrázek je vstupem pro enkodér, jehož cílem je snížení dimenze vstupního obrázku. Enkodér je konvoluční neuronová síť, obsahuje osm konvolučních vrstev velikostí 64-64-64-64-128-128-128-128 neuronů respektive. Každá ta-



Obrázek 2.9: Architektura modelu SuperPoint

ková vrstva má jádro $3 \times 3 \times 1$. Po každých dvou vrstvách následuje max pool vrstva s jádrem 2×2. Po zpracování obrázku enkodérem je jeho velikost $H/8 \times W/8 \times F$, kde $F > 1$. Po enkodéru se model rozdělí na dvě „hlavy“ dekodéru, každý se učí váhy specifické pro vlastní úkol - jeden pro hledání zajímavých bodů a druhý pro výpočet deskriptorů zajímavých bodů. Každá ”hlava”dekodéru má jednu 3×3 konvoluční vrstvu s 256 neurony, dále je 1×1 konvoluční vrstva s 65 neurony a 256 neuronů pro detektor zajímavých bodů a deskriptor zajímavých bodů. Každá konvoluční vrstva je následována aktivační funkcí ReLU a normalizací BatchNorm.

Dekodér pro detekci zajímavých bodů má ve výstupu pole velikosti $H \times W$ čísel, každé číslo odpovídá pravděpodobnosti, že je na stejné pozici ve vstupním obrázku zajímavý bod. Na začátku tohoto dekodéru je explicitní dekodér bez parametrů, potom pole racionálních čísel velikosti $H/8 \times W/8 \times 65$ (mřížka hodnot 8×8 plus jedna hodnota navíc pro „nezajímavý pixel“ - dustbin) postupně projde funkcí softmax, a tím získává výsledek. Ten dustbin je nutný, pokud se v mřížce nebude vyskytovat žádný klíčový bod.

Dekodér deskriptoru má ve výstupu pole velikosti $H \times W \times D$ deskriptorů. Jeden deskriptor na každých 8 pixelů. Dekodér používá L2 normalizaci a bikubickou interpolaci. Interpolace je nalezení přibližné hodnoty funkce v nějakém intervalu, je-li její hodnota známa jen v některých jiných bodech tohoto intervalu. Bikubická interpolace je interpolací datových bodů na dvourozměrnou pravidelnou mřížku.

Podrobná architektura modelu SuperPoint je na obrázku 2.9.

2.3.3 LF-Net

Autoři článku [10] navrhli metodu s novou hlubokou architekturou pro detekci příznaků a generování deskriptoru příznaků. Je end-to-end a nevyžaduje použití ručně vytvořeného detektoru ke generování trénovací množiny. Místo toho jsou použity páry obrazů, pro které známe pozici objektů na snímku a

hloubkovou mapu odpovídající tomuto snímku.. Hloubková mapa je obraz, který obsahuje informace týkající se vzdálenosti povrchů objektů scény od hlediště.

Metoda

LF-Net má dvě hlavní komponenty. První z nich je plně konvoluční neuronová síť, která vrací pozice, měřítka a orientace klíčových bodů. Je navržena tak, aby dosáhla rychlého času běhu a byla nezávislá na velikosti obrázku. Druhou komponentou je neuronová síť, která vrací deskriptory klíčových bodů vytvořených první sítí.

Nejprve je aplikována plně konvoluční neuronová síť ke generování mapy příznaků **o** ze vstupního obrázku **I**, kterou lze použít k extrahování klíčových bodů a jejich atributů, tj. měřítka a orientace. K tomu existují dva důvody. Nejprve se ukázalo, že použití takové reprezentace k odhadu veličin pomáhá zvýšit prediktivní úspěšnost hlubokých neuronových sítí. Za druhé umožňuje použití více obrázků současně, proto je klíčové robustní detektor. Pro implementaci je používána síť ResNet se třemi bloky. Každý blok obsahuje 5×5 konvoluční jádro následované batch normalizací, aktivací ReLU a další posloupností z 5×5 konvolucí. Všechny konvoluce jsou zero-padding, aby měly stejnou velikost výstupu jako vstup, a mají 16 výstupních kanálů.

K detekci klíčových bodů neměnných vůči měřítku je navržen nový přístup k detekci scale-space, který se opírá o mapu příznaků **o**. Na generování scale-space výsledků je změněna velikost mapy N-krát, v jednotných intervalech mezi $1/R$ a R, kde $N = 5$ a $R = \sqrt{2}$ podle experimentů autorů. Výsledky jsou zpracované N nezávislými jádry velikostí 5×5, což má ve výsledku N map h^n pro $1 \leq n < N$, jedna pro každou stupnici. Dále je na každou mapu aplikován operátor softmax s jádrem 15×15 konvolučním způsobem, což má ve výsledku N skóre map h^n, kde $1 \leq n < N$. Dále velikost každé skóre mapy h_n změněna zpět na původní velikost obrázku. Nakonec jsou všechny h_n spojené do konečné scale-space mapy **S** pomocí následujícího vzorce:

$$\mathbf{S} = \sum_n h^n \bigodot softmax_n(h^n) \ , \tag{2.13}$$

kde $\bigodot$ je Hadamardův součin. Nejlepší K pixelů jsou klíčovými body.

Pro učení orientace je na mapu příznaků **o** aplikována jedna konvoluce 5×5. Výstupem pro každý pixel jsou dvě hodnoty. Berou se jako sinus a kosinus orientace a používají se k výpočtu husté orientační mapy θ pomocí arktanové funkce.

Jak bylo popsáno výše, jsou ze scale-space mapy **S** extrahované klíčové body K a jejich umístění v obraze (x, y). S měřítkovou mapou **s** a orientační mapou θ to dává K čtveřic ve tvaru $p^k = \{x, y, s, \theta\}^k$, pro které je třeba vypočítat deskriptory. Pro každý klíčový bod je vytvářeno pole sousedů to-

Obrázek 2.10: Párování příznaků metodou D2-Net

hoto bodu. Tato pole jsou oříznuta z normalizovaných obrázků a jejich velikost je změněna 32×32. Neuronová síť pro deskriptor zahrnuje tři 3×3 konvoluční filtry s krokem 2 a 64, 128 a 256 kanálů. Po každém z nich následuje batch normalizace a aktivace ReLU. Po konvolučních vrstvách je fully connected vrstva s 512 kanály, po ní je provedena batch normalizace, pak aktivace ReLU a na konci je fully connected vrstva, pro snížení dimenze na $M = 256$. Deskriptory jsou pak normalizovány.

2.3.4 D2-Net

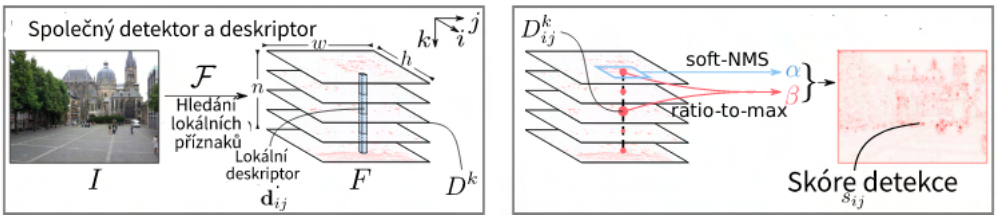
V této práci [11] se autoři věnovali problému hledání shodných pixelů za obtížných zobrazovacích podmínek. Byl navržen postup, kde jedna konvoluční neuronová síť je současně deskriptorem příznaků a detektorem klíčových bodů.

Model D2-Net je zaměřen na získání sady lokálních příznaků, které jsou robustní za náročných obrazových podmínek (například různá denní doba) a jsou efektivní při porovnávání a ukládání. Nejprve je přepočítána množina aktivačních map pomocí konvoluční neuronové sítě. Tyto mapy se poté používají k výpočtu deskriptorů (jako řezy skrz všechny mapy na konkrétní pozici pixelu) a k detekci klíčových bodů (jako lokální maxima prvků map). Výsledkem je, že detektor lokálních příznaků je pevně spojen s deskriptorem příznaků. Výsledná struktura D2-Net je ukázaná na obrázku 2.11.

Experimenty, které provedli autoři metody D2-Net, ukazují, že takový přístup funguje srovnatelně dobře, nebo dokonce lépe za náročných podmínek, jako jsou změny denního a nočního osvětlení. Příklad párování příznaků pomocí metody D2-Net je ukázán na obrázku 2.10. Experimentům se podrobně věnuje kapitola Analýza.

Metoda

Konvoluční neuronová síť $\mathcal{F}$ se používá k extrahování aktivačních map, které plní dvě různé role:



Obrázek 2.11: Model D2-Net

1. Lokální deskriptory d_{ij} se jednoduše získají procházením všech n aktivačních map D_k v prostorové poloze (i, j);

2. Detekce se získají provedením „non-local-maximum“ potlačení na aktivační mapu, po kterém následuje „non-maximum“ potlačení napříč každým deskriptorem - během učení se skóre detekce klíčových bodů s_{ij} vypočítá z „local-maximum“ skóre α a „ratio-to-maximum“ skóre na deskriptor β.

Prvním krokem metody je použití CNN $\mathcal{F}$ na vstupní obraz I k získání pole $F = \mathcal{F}(I)$, $F \in \mathbb{R}^{h \times w \times n}$, kde $h \times w$ je prostorové rozlišení aktivačních map a n je počet kanálů obrazu.

Pole F obsahuje hustou množinu vektorů deskriptorů d:

$$d_{ij} = F_{ij}, d \in \mathbb{R}^n \ , \tag{2.14}$$

kde $i = 1, ..., h$ a $j = 1, ..., w$. Tyto vektory deskriptorů lze snadno porovnat mezi obrazy a najít tak shody pomocí euklidovské vzdálenosti. Během fáze učení budou tyto deskriptory upraveny tak, aby stejné body ve scéně vytvářely podobné deskriptory, i když obrázky obsahují silné změny vzhledu. Je použita normalizace L2 na deskriptory před jejich porovnáním.

Odlišná interpretace pole F je jako množina D 2D shod:

$$D^k = F_{::k}, D^k \in \mathbb{R}^{h \times w} \ , \tag{2.15}$$

kde $k = 1, ..., n$. V této interpretaci lze funkci extrahování příznaků F považovat za n různých funkcí D^k detektorů příznaků, z nichž každá vytváří 2D mapu D^k shod. Tyto mapy detekčních shod jsou analogické s mapami shod Difference-Gaussians (DoG) získanými v Scale Invariant Feature Transform (SIFT). V této práci se tyto hrubé výsledky následně zpracovávají, aby se jako výstupní klíčové body vybrala pouze podmnožina míst.

V této metodě existuje několik detekčních map $D^k (k = 1, ..., n)$ a detekování může probíhat na kterékoli z nich. Proto pro detekci bodu (i, j) vyžadujeme:

(i, j) je detekován $\iff$ D^k_{ij} je lokální maxima v D^k, $k = \underset{t}{argmax} D^t_{ij}$

Intuitivně to pro každý pixel (i, j) odpovídá výběru nejvýznamnějšího detektoru D^k (výběr kanálu) a následnému ověření, zda je lokální maximum v poloze (i, j) na mapě odezvy konkrétního detektoru D^k.

V průběhu učení se procedura detekování, která je popsána výše, trochu mění, aby byla vhodná pro back-propagaci. Back-propagace je postup získání vah neuronů prodanou síť a hodnotu ztrátové funkce.

Local-max skóre:

$$\alpha_{ij}^k = \frac{exp(D_{ij}^k)}{\sum_{i',j' \in \mathcal{N}(i,j)} exp(D_{i'j'}^k)} \ , \tag{2.16}$$

kde $\mathcal{N}(i,j)$ je množina 9 sousedů pixelu (i, j) včetně sebe.

Je definován výběr kanálu, který se počítá jako poměr ratio-to-max per deskriptor:

$$\beta_{ij}^k = D_{ij}^k / \underset{t}{max} D_{ij}^t \ . \tag{2.17}$$

Dále, aby byly v úvaze obě kritéria, je maximalizován součin obou skóre napříč všemi aktivačními mapami ‖, aby byla získána jedna mapa skóre: $\gamma_{ij} = \underset{t}{max}(\alpha_{ij}^k, \beta_{ij}^k)$.

Skóre detekce s_{ij} v pixelu (i,j) se získá provedením normalizace na úrovni obrazu:

$$s_{ij} = \gamma_{ij} / \sum_{(i',j')} \gamma_{i',j'} \ . \tag{2.18}$$

2.4 Metody párování příznaků

V předchozí sekci byly prozkoumány metody na hledání lokálních příznaků na obrázku. Nezabývají se ale tím, jak spárovat příznaky dvojic obrázků a jak najít shody mezi obrázky. Na to se používají algoritmy, které jsou popsány v této sekci.

2.4.1 Brute Force párování

Metoda Brute Force [12] postupně prochází každým z klíčových bodů jednoho obrázku, porovnává s každým klíčovým bodem druhého obrázku a hledá nejlepší párování. Nejlepší párování má nejmenší vzdálenost mezi příznaky. Udává přesné výsledky, ale pro velký počet klíčových bodů je tato metoda pomalá.

2.4.2 FLANN based matcher

FLANN (ang. Fast Library for Average Nearest Neighbours) [12] obsahuje sadu algoritmů optimalizovaných pro rychlé hledání nejbližších sousedů ve velkých datových sadách a pro vysoce dimenzionální příznaky. Funguje rychleji než Brute Force matcher pro velké datové sady.

Jako parametry FLANN based matcher dostává dva slovníky, které specifikují použitý algoritmus, jeho související parametry atd. První je IndexParams. Druhým slovníkem je SearchParams. Určuje počet opakování procházení stromů v indexu. Vyšší hodnoty poskytují lepší přesnost, ale také vyžadují více času.

2.4.3 SuperGlue

V tomto článku [13] autoři představili SuperGlue, což je neuronová síť, která přiřadí elementy dvou sad lokálních příznaků mezi sebou. Síť zároveň hledá shody mezi příznaky a eliminuje neshodné body.

Je nutné uvést další omezení:

1. Klíčový bod může mít nanejvýš jednu shodu v jiném obrázku.

2. Některé klíčové body nebudou překonány kvůli okluzi a poruše detektoru.

SuperGlue si klade za cíl najít veškeré párování mezi stejnými body v několika obrazech a identifikovat klíčové body, které mezi sebou nejsou shodné.

Metoda

Architektura SuperGlue se skládá ze dvou základních komponent: Attentional Graph Neural Network a Optimal Matching layer.

Attentional Graph Neural Network je enkodérem klíčových bodů, kterým je každý klíčový bod kódován. Tento kodér neuronové síti umožňuje uvažovat o společném vzhledu i poloze.

Lze představit klíčové body dvou obrázků jako jediný graf, kde každý klíčový bod je uzel. Pak existují dva způsoby, jak umístit neorientované hrany:

1. Hrany uvnitř obrazu (vlastní hrany), které spojují klíčové body s jinými klíčovými body ve stejném obrazu.

2. Inter-image hrany (cross hrany), které spojují klíčové body s dalšími klíčovými body v jiném obrázku. Formulace předávání zpráv se používá k šíření informací mezi oběma typy hran.

Vlastní hrany jsou založeny na self-attention a příčné hrany jsou založeny na cross-attention. Existují L vrstev s různými parametry, které jsou spojeny dohromady a agregovány podél vlastních a cross hran. Liché vrstvy vytvářejí vlastní hrany a sudé vrstvy provádějí cross hrany. Nakonec jsou konečné deskriptory shody vytvořeny pro každý klíčový bod lineárními projekcemi.

Optimal Matching layer vytváří matici částečného přiřazení. Nejprve se vypočítá matice skóre S pro všechna možná párování a maximalizuje se celkové skóre $S \times P$ (P je soft matice přiřazení). Každá sada klíčových bodů je také rozšířena o dustbin, který potlačuje nenapárované klíčové body. Skóre matice bude $(M+1) \times (N+1)$ matice, kde M je počet místních rysů obrazu A, N je počet místních rysů obrazu B. Poté se k normalizaci součtu použije Sinkhornův algoritmus řádků a sloupců matice skóre[14]. Jedná se o diferencovatelnou verzi maďarského algoritmu [15], klasicky používanou pro bipartitní párování, která spočívá v iterativní normalizaci $\exp S$ podél řádků a sloupců, podobně jako u řádků a sloupců Softmax. Po T iteracích jsou dustbins odhozeny a P je obnoveno (P matice má velikost $M \times N$).

2.5 Knihovny Pythonu pro strojové učení

Tato sekce je věnována dvěma knihovnám Pythonu, které se běžně používají pro strojové učení.

2.5.1 TensorFlow

TensorFlow [16] je knihovnou s otevřeným zdrojovým kódem pro Python, která dokáže rychle provádět číselné výpočty. Umožňuje běh algoritmů na procesoru (CPU), na grafických kartách (GPU) a na mobilních zařízeních. Výpočetní výkon GPU je mnohem vyšší než u procesoru.

TensorFlow výpočty provádí pomocí grafů. V těchto grafech jsou vrcholy vektory (tenzory), zatímco hrany reprezentují vztahy mezi těmito tenzory.

2.5.2 PyTorch

PyTorch [17] je knihovnou pro rychlé číselné výpočty založené na Pythonu, který využívá grafické karty pro výpočty. Je to také jedna z upřednostňovaných výzkumných platforem pro hluboké učení postavených tak, aby poskytovaly maximální flexibilitu a rychlost. Je známo, že poskytuje dvě funkce na nejvyšší úrovni - tenzorové výpočty se silnou podporou akcelerace GPU a budování hlubokých neuronových sítí.

2.6 Čidla mobilních telefonů

Čidla dělají mobilní telefon chytrým zařízením [18]. Pomáhají člověku orientovat se v prostoru, reagují na svět, dotyk nebo pohyb. Dále jsou uvedeny

základní senzory chytrých mobilů, pomocí kterých je možné získat informaci o přesné lokalitě a poloze zařízení v prostoru.

GPS

Globální polohový systém (GPS) je satelitní systém vyvinutý ve Spojených státech amerických na začátku 70. let 20. století. Posílá informaci o poloze zařízení bez ohledu na počasí. Jde o jednosměrový systém, což znamená, že uživatel může od GPS informaci pouze dostávat, nikoliv posílat.

GPS sestává ze souboru z 24 umělých satelitů. Aby bylo možné získávat informaci o poloze zařízení, které se nachází v libovolném místě na Zemi, jsou tyto satelity umístěny vždy po čtyřech na každé z šesti oběžných rovin. Díky takovému umístění je zařízení vždy schopné se propojit se 4 až 10 umělými družicemi. To stačí pro získání informace o lokalitě v libovolném místě na Zemi.

Akcelerometr

Je senzorem, který reaguje na zrychlení zařízení. Vrátí proudově množinu vektorů $Acc =< x_i, y_i, z_i >, i \subseteq \mathbb{N}$. V současné době se používá pro rozpoznávání různých druhů aktivit uživatelů (běh, chůze, rozpoznává také, když uživatel s mobilním telefonem stojí na místě).

Kompas

V moderních telefonech se vyskytuje analog už všemi známého nástroje pro získání informace o poloze zařízení, jeho směru a orientace ve fyzickém prostoru.

Mobilní kompas se skládá ze tří magnetických senzorů, které měří orientaci mobilního telefonu ve třírozměrném prostoru. Vrátí racionální číslo v rozsahu od 0 do 360. Toto číslo je úhlem mezi směrem mobilního zařízení a azimutem.

Gyroskop

Gyroskop je nástroj, který měří rotaci zařízení v třírozměrném prostoru. Vrátí úhel rotace podle každé ze tří os X, Z a Z.

Kapitola 3

Analýza

V této kapitole je popsána analýza existujícího řešení, informací z rešeršní části, vlastních pozorování, porovnání jednotlivých algoritmů a metod.

Změny a případné zlepšení stávajícího řešení jsou popsány v kapitole Návrh.

3.1 Analýza existujícího řešení

Jan Šefčík ve své práci [19] navrhl a implementoval modul pro rozpoznání budovy na snímku s využitím metod klasického strojového vidění. Podrobnější analýza je popsána v této sekci.

3.1.1 Analýza z hlediska návrhu

Modul prototypu aplikace používá konfigurační soubor pro základní nastavení parametrů aplikace. Konfigurační soubor obsahuje nastavení poloměru okolí kolem uživatele v metrech. Tato konstanta určuje maximální vzdálenost mezi uživatelem a budovou. Dalšími parametry pro nastavení jsou cesta do vstupního snímku, metadata budov z databáze (název, id, cesta do složky s obrázky, GPS souřadnice), parametry pro výpočet a párování lokálních příznaků. Z těchto parametrů by se měla měnit pouze cesta do vstupního obrázku pro každé spuštění aplikace uživatelem. Z toho důvodu je vhodné změnit místo pro nastavení vstupního obrázku, například přidat cestu do obrázku jako argument při spuštění aplikace.

Jan Šefčík navrhl, aby byl v databázi budov každý snímek svoji upravenou verzi s oříznutou budovou a černým pozadím. Tímto způsobem se hledají shody mezi příznaky ležící na budově, a tím přesněji se umístí budova do scény. Má ale z hlediska cíle projektu Věnná města českých královen jeden nedostatek– nemůže do scény umístit budovu, která byla postavena ve středověku, ale v realitě již neexistuje. V takovém případě nemůže mít uživatel představu, jak tato scéna ve středověku vypadala.

3.1.2 Analýza z hlediska časové náročnosti

Pro hledání klíčových bodů a výpočet jejich deskriptorů byla použitá implementace metody SIFT z knihovny OpenCV. Metoda SIFT je založena na histogramech gradientů. To znamená, že je třeba vypočítat gradienty každého pixelu v políčku. Tyto výpočty zaberou spoustu času. Což znamená, že metoda SIFT nemůže být používána v aplikacích reálného času, pokud budou používány velké obrázky.

Pro párování příznaků byl v této práci zvolen FLANN based matcher, což je optimální řešení pro větší databázi. Díky předběžnému filtrování příznaků podle Lowe poměrového testu má FLANN based matcher ještě menší časovou náročnost. Nemá ale ideální přesnost, protože nehledá FLANN based matcher přesné párování, ale párování přibližné.

3.1.3 Analýza z hlediska optimality

Z každé budovy v databázi je ručně oříznuta budova a umístěna do černého pozadí. To je pomalý způsob, vyžaduje několik minut práce správcem aplikace pro každý obrázek. Databáze budov a jejich snímků se bude zvětšovat, což dělá takový způsob oříznutí budov ze snímků neoptimálním.

3.2 Použití lokalizačních dat pro rozpoznávání scény

V databázi projektů Věnná města českých královen mají modely budov unikátní parametr, jsou to GPS souřadnice těchto budov v realitě. Jan Šefčík[19], na jehož práci ve své práci navazuji, použil vyhledávání budovy v databázi podle jejích GPS souřadnic. Poloměr okolí kolem uživatele byl zvolen na 250 metrů. Tato konstanta bere v úvahu nepřesnost dat o poloze uživatele i budovy.

V současné době není databáze projektu příliš rozsáhlá. Ale i nyní se v okolí poloměru 250 metrů kolem uživatele může vyskytovat několik budov z databáze. Pokud program najde několik budov v databázi, pak se budou příznaky z každého snímku každé z těchto budov porovnávat s příznaky vstupního snímku. Pro urychlení doby běhu celého programu je nutně zúžit okolí uživatele.

Z těchto důvodů je potřeba určit směr kamery. Pro určení směru kamer dle roviny Země je možné použít data z mobilního kompasu a hledat potom v databázi pouze budovy ve směru kamery. Mnoho moderních telefonů má fotoaparát s úhlem záběru 120° nebo větší. Z důvodu nepřesnosti geolokačních dat jsem zvolila hodnotu 140° jako optimální.

Okolí uživatele má tvar kruhu, program bude tedy hledat budovy, které jsou v kruhové výseči poloměrem 250 metrů (dle výsledků analýzy [19] je optimální hodnota), středovým úhlem 140°, se středem v poloze uživatele a směr je určen daty z mobilního kompasu.

3.3 Analýza metod pro extrahování klíčových bodů a výpočet deskriptorů

V této podkapitole je uvedena analýza a porovnávaní metody SIFT s moderními metodami hlubokého učení.

Je nutně uvést informaci o používaných implementacích metod, analyzovaných v této podkapitole:

- *SIFT.* Pro analýzu byla zvolena standardní implementace metody SIFT z knihovny OpenCV [20].

- *SuperPoint.* Tato metoda nemá oficiální implementaci, má ale Open-Source PyTorch implementaci. Zdrojové kódy a už natrénované modely lze stáhnout z GitHubu[1]. Je implementována přesně podle článku s použitím v dekodéru deskriptoru bilineární interpolace (interpolace funkce dvou proměnných na pravidelnou prostorovou mřížku) místo interpolace bikubické. Taková změna dělá výpočty rychlejšími, ale neovlivňuje to přesnost výsledků.

- *D2-Net.* Metoda má oficiální PyTorch implementaci s už natrénovanými modely. Je ke stažení na GitHubu[2] [11]. Je nutně uvést, že lokální příznaky, které byly extrahovány pomocí těchto modelů, nejsou invariantní vůči rotaci.

- *LF-Net.* Metoda má oficiální Tensorflow implementaci s už natrénovanými modely. Je ke stažení na GitHubu[3] [10].

V průběhu analýzy byla metoda SuperPoint považována za nejvhodnější pro účely dané práce. Je schopna fungovat pro aplikaci reálného času, velice dobře extrahuje klíčové body a přepočítá deskriptory, díky čemuž dobře rozpoznává budovy na snímcích efektivit dané práce, a to i za komplikovaných podmínek osvětlení. Je moderní a má implementaci s otevřeným přístupem.

Dále jsou uvedena porovnání metod SIFT a SuperPoint z častí experimentů z článku [9] podle míry opakovatelnosti a odhadu homografie. Experimenty byly provedeny na datové sade obsahující 116 scén a 696 unikátních obrázků. Prvních 57 scén obsahuje snímky se změnami světelných podmínek, ostatních 59 scén obsahuje obrázky snímané z různých uhlů pohledu. U všech obrazů je poskytována odhadovaná ground truth homografie H. V analýze pro hledání shod mezi příznaky byla použita metoda Brute Force matcher, pro eliminaci špatných párování byla použita metoda RANSAC.[21]

[1] https://github.com/magicleap/SuperPointPretrainedNetwork
[2] https://github.com/mihaidusmanu/d2-net
[3] https://github.com/vcg-uvic/lf-net-release

	Opakovatelnost
SuperPoint	0.581
SIFT	0.495

Tabulka 3.1: Porovnání podle míry opakovatelnosti

Porovnání podle míry opakovatelnosti

Míra opakovatelnosti je pravděpodobnost, že klíčový bod extrahovaný z jednoho obrázku se vyskytuje i na druhém obrázku. Měří se pomocí vzdálenosti mezi extrahovanými středy 2D bodů. Pomocí ϵ je reprezentována správná prahová hodnota vzdálenosti mezi dvěma body. Konkrétněji předpokládejme, že máme N1 bodů na prvním obrázku a N2 bodů na druhém obrázku. Správnost (ang. correctness) pro experimenty s opakovatelností je definována takto:

$$corr(x_i) = (\min_{j \in \{1,..,N_2\}} ||x_i - \hat{x}_j||) \leq \epsilon \ , \tag{3.1}$$

kde x_i je klíčový bod prvního obrázku a $\hat{x}_j$ je jeho 2D projekce v druhém obrázku. Klíčový bod je považován za „opakovatelný“, pokud vzdálenost mezi ním a jeho 2D projekcí v druhém obraze je menší nebo rovna ϵ.

Potom je vzorec pro výpočet opakovatelnosti následující:

$$Rep = \frac{1}{N_1 + N_2}(\sum_i corr(x_i) + \sum_j corr(x_j)) \ . \tag{3.2}$$

Toto porovnání bylo provedeno na obrázcích velikosti 240×320, počet hledaných pixelů byl nastaven na 300. Navíc byla použita metoda Non Maximum Suppression na klíčové body. Non Maximum Suppression je třída algoritmů pro výběr jedné entity z mnoha překrývajících se entit. K dosažení konkrétních výsledků lze zvolit kritéria výběru. Kritériem je nejčastěji nějaká forma čísla pravděpodobnosti spolu s nějakou formou míry překrytí. V daném případě kritériem výběru byla zvolena hodnota 3 pixely.

Míra opakovatelnosti může nabývat hodnoty v intervalu $[0; 1]$. Čím je táto hodnota blíže k 1, tím jsou klíčové body extrahované metodou opakovatelnější. Podle výsledků 3.1 mají klíčové body SuperPoint větší opakovatelnost než klíčové body SIFT.

Odhad homografie

Měří schopnost algoritmu odhadnout homografii vztahující se k dvojici obrazů porovnáním odhadované homografie $\hat{\mathcal{H}}$ s ground-truth homografií $\mathcal{H}$. Srovnání matic 3×3 $\mathcal{H}$ není jednoduché protože různé položky v matici mají

	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
SuperPoint	0.310	0.684	0.829
SIFT	0.424	0.676	759

Tabulka 3.2: Porovnání podle odhadu homografie

	Matching Score
SuperPoint	0.372
SIFT	0.184
LF-Net	0.406

Tabulka 3.3: Porovnání podle matching score

různé váhy. Místo toho je potřeba porovnávat, jak dobře homografie transformuje čtyři rohy jednoho obrazu na druhý. Definujeme čtyři rohy prvního obrázku jako c_1, c_2, c_3, c_4. Poté použijeme ground-truth $\mathcal{H}$, abychom získali rohy ground-truth na druhém obrázku c'_1, c'_2, c'_3, c'_4 a odhadovanou homografii $\hat{\mathcal{H}}$ pro výpočet $\hat{c}'_1, \hat{c}'_2, \hat{c}'_3, \hat{C}'_4$. Pro označení správné homografie používáme práh ϵ. Vzorec pro výpočet je:

$$CorrH = \frac{1}{N}\sum_{i=1}^{N}((\frac{1}{4}\sum_{j=1}^{4}||c'_{ij} - \hat{c}'_{ij}||) \leq \epsilon) \ . \tag{3.3}$$

Při porovnání byly použity obrázky s rozlišením 480×640 a počet hledaných příznaků byl nastaven na 1000.

Dle výsledků 3.2 SIFT funguje dobře u homografií s přesností na subpixely $\epsilon = 1$. To je pravděpodobně způsobeno tím, že SIFT provádí další lokalizaci subpixelů, zatímco metoda SuperPoint tento krok neprovádí.

Porovnání podle matching score

Tato metrika měří celkový výkon kombinovaného detektoru a deskriptoru zajímavých bodů [22]. Měří poměr ground-truth shod, které mohou být nalezeny v celém pipelinu přes počet příznaků navržených pipelinem ve sdíleném pohledu. Tato metrika je vypočítána symetricky napříč dvojicí obrázků a zprůměrována.

Podle experimentů popsaných v článku [10], kde byly porovnány metody LF-Net, SuperPoint a SIFT, LF-Net má v průměru větší matching score na datové množině snímků vnějších scén, obsahující 11 scén. Výsledky experimentů jsou uvedeny v tabulce 3.3.

	NF	NI	mAA
SuperPoint	2048	120.1	0.2577
SIFT	2048	84.9	0.2489
D2-Net	2038	149.3	0.1524
LF-Net	2020	100.2	0.1927

Tabulka 3.4: Porovnání podle mean Average Accuracy

Porovnání podle mean Average Accuracy
V článku[23] jsou analyzovány různé metody pro hledání lokálních příznaků, včetně D2-Net, SIFT, LF-Net a SuperPoint.

Pro porovnání byla využita obrovská datová množina s 9 vnějšími scénami, které obsahují skoro 10 000 snímků různých architektonických památek.

Autoři vypočítali rozdíl ve stupních mezi odhadovaným a groung-truth vektory posunutí a rotace mezi dvěma kamerami. Poté výsledky omezili dle daného pro všechny dvojice obrazů prahu. Učinili to přes různé úhlové prahové hodnoty a vytvořila se křivka. Vypočítali průměrnou přesnost (mAA) jako integrál této křivky až k maximální prahové hodnotě, kterou nastavili na 10°.

NF - počet příznaků (ang. number of features)

NI- počet správně spárovaných příznaků (ang. number of inliers)

Dle výsledků v tabulce 3.4 metody D2-Net a LF-Net vykazují horší výsledky než metody SuperPoint a SIFT. Pro dosažení lepších výsledků je u metod D2-Net, LF-Net třeba modifikovat (ang. tune) hyperparametry. Metoda SuperPoint podle mean Average Accuracy dosáhla lepších výsledků.

3.3.1 Analýza časové náročnosti

Obecně se pro porovnání a analýzu časové náročnosti používá časová složitost algoritmů. Pro komplexní algoritmy takový postup nemá smysl, protože komplexní algoritmy velmi závisí na konstantách. Pro časovou analýzu takových algoritmů se používají experimenty.

Doba běhu algoritmů závisí i na technických charakteristikách stroje, na kterém se tyto experimenty provedly.

Technické charakteristiky:

- CPU: Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz
- GPU: NVIDIA GM108M [GeForce 840M]

Všechny metody jsem otestovala na datové sadě z 10 snímků vnějších scén. Počet extrahovaných klíčových bodů je pro každou metodu nastaven na 1000. Postupně byly obrázky oříznuty na různé velikosti. Pro každou velikost obrázku byla spuštěna testovaná metoda 10krát na každý obrázek, pak

Velikost obrázku	SIFT	SuperPoint	LF-Net	D2-Net
320×240	0.09	1.21	1.1	2.46
640×480	0.22	1.22	1.1	3.11
960×540	0.91	1.24	1.09	3.52
1080×720	1.1	1.24	1.09	4.1
2000×1500	3.01	1.24	1.07	6.7

Tabulka 3.5: Analýza časové náročnosti

průměr z těch 100 naměřených hodnot je výsledná časová náročnost dané metody pro danou velikost obrázku.

Výsledky testování jsou ukázány v tabulce 3.5. Hlavní nevýhodou metod SIFT a D2-Net je závislost na velikosti obrázku. Kvůli tomu není možné používat tyto metody pro aplikace reálného času. Pro menší obrázky je nejrychlejší metodou SIFT.

3.3.2 Analýza metod na vlastní datové množině

Vytvořila jsem vlastní datovou množinu. Tato množina obsahuje 49 sekvencí snímků vnějších scén. Každá sekvence obsahuje 2 až 5 snímků budovy z různých úhlů pohledu a s různými podmínkami osvětlení. Pro každý snímek existuje jemu odpovídající upravená verze snímku s oříznutou budovou na černém pozadí.

Každý vstupní obrázek a jeho upravená verze byly oříznuty na velikost 240×320, převedeny do odstínů šedi a normalizovány.

Schopnost rozpoznání budovy

n je počet příznaků nalezených testovanou metodou na testovaném snímku, m je počet příznaků, které jsou umístěny na budově. Pak $k = \frac{m}{n}$ je koeficientem úspěšnosti dané metody. Hodnota koeficientu je v rozsahu $[0, 1]$. Čím větší je ten koeficient, tím lépe testovaná metoda rozpoznává budovu ze snímku.

Pro provedení analýzy podle dané metriky byla na každý obrázek spuštěna testovaná metoda, a navíc byly získány pozice klíčových bodů. Byla vytvořena maska upravené verze každého obrázku (budova byla vyplněna bílou barvou, pozadí zůstalo černé). Byl přepočítán podíl počtu klíčových bodů, umístěných na bílé oblasti upraveného snímku ku celkovému počtu klíčových bodů na snímku. Výsledkem (úspěšnost testované metody) je zprůměrovaná hodnota pro všechny obrázky z datové množiny. Úspěšnosti každé testované metody jsou ukázány v tabulce 3.6. Největší úspěšnost dle dané metriky má metoda SIFT.

	Úspěšnost
SuperPoint	0.424
SIFT	0.485
D2-Net	0.429
LF-Net	0.412

Tabulka 3.6: Rozpoznání budovy na snímku

	Úspěšnost
SuperPoint	0.361
SIFT	0.344
D2-Net	0.221
LF-Net	0.181

Tabulka 3.7: Úspěšnost párování deskriptorů

Porovnání podle úspěšnosti párování deskriptorů

Pro každou ze sekvencí byly vytvořeny páry obrázků: první obrázek z každé sekvence byl spárován s každým dalším obrázkem ze stejné sekvence. Pro každý obrázek každého páru byla spuštěna testovaná metoda pro výpočet lokálních příznaků a jejich deskriptorů. Pro párování deskriptorů byl použit Brute Force Matcher, který je popsán v kapitole Rešerše.

Z důvodu zmenšení paměťové náročnosti a zrychlení procesu párování klíčových bodů byl pro každou metodu omezen počet hledaných klíčových bodů pro každý obrázek na 500. Po párování deskriptorů byla pomocí metody RANSAC eliminována chybná spojení. Koeficientem úspěšnosti metody je poměr počtu dobře spárovaných klíčových bodů ku celkovému počtu klíčových bodů. Hodnota koeficientu nabývá hodnot od 0 do 1, kde 0 znamená, že žádný z deskriptorů nebyl spárován správně, 1 znamená, že žádné párování nebylo eliminováno. Výsledky porovnání dle dané metriky jsou uvedeny v tabulce 3.7. Metoda SuperPoint dle této metriky udává nejlepší výsledky.

3.4 Analýza metody SuperGlue

V části Rešerše jsou popsány 3 algoritmy pro párování lokálních příznaků.

Brute Force matcher je neefektivní metoda pro velké datové sady. Vzhledem k tomu, že se bude stávající databáze zvětšovat, je metoda Brute Force Matcher v rámci daného projektu považována za nepoužitelnou.

V té sekci byly porovnány metody SuperGlue a FLANN based matcher. Na základě další analýzy byla zvolena metoda SuperGlue pro použití v dané

práci.

Použitá implentace

Metoda SuperGlue má oficiální PyTorch implementaci s již naučeným modelem. Zdrojové kódy této implementace jsou na GitHubu[4] [24]. Obsahuje model, naučený na datové sadě vnitřních scén a model naučený na datové sadě vnějších scén.

Analýza časové náročnosti

Před výpočtem doby běhu programu pro párování klíčových bodů bylo extrahováno 1024 klíčových bodů u 50 obrázků. Tyto obrázky byly převzaty z vlastního datasetu, který už byl popsán v kapitole Analýza. Poté byl zvolen jeden z těchto obrázků a byla spuštěna metoda SuperGlue s prahem nastaveným na 0.2, počtem iterací Sinkhorného algoritmu rovným 20 a přepočítanými pro dataset vnějších scén váhy pro každý z ostatních obrázků.

Pro měření doby běhu byla použitá standardní knihovna „time“ pro Python. Výsledky pak byly zprůměrovány. Výsledná průměrná doba běhu metody SuperGlue na GPU NVIDIA GM108M [GeForce 840M] je 0.2 vteřiny.

Analýza podle matching score a presicion

Kvůli tomu, že se táto práce zabývá rozpoznáváním vnějších scén, je nutně analyzovat model naučený na datasetu vnějších scén.

Pro porovnání byl použit dataset s 11 sekvencemi vnějších scén, celkem obsahuje 4000 snímků různých architektonických památek. Velikost každého obrázku ze vstupní dvojice se změní tak, aby se jeho nejdelší strana rovnala 1600 pixelům. Počet extrahovaných klíčových bodů byl nastaven na 2048. Pro SuperPoint byla hodnota Non Maximum Suppression zvolena tak, aby se rovnala 3.

NN je metoda párování pomocí hledání nejbližšího souseda (ang. nearest neighbor), což je algoritmem ze třídy FLANN based matcher. Pro SIFT je použit Lowe poměrový test (ang. ratio test), který odstraňuje špatné párování. Poměr vzdálenosti mezi dvěma nejbližšími shodami uvažovaného klíčového bodu je vypočítán a je dobrá shoda, pokud je tato hodnota pod prahovou hodnotou. Podle článku [25] je pro SIFT nejvhodnější prahová hodnota 0.6.

Přesnost (P) je průměrný poměr počtu správných shod k celkovému počtu odhadovaných shod. Matching score (MS) je průměrný poměr počtu správných shod k celkovému počtu detekovaných klíčových bodů.

V tabulce 3.8 jsou uvedeny výsledky porovnání. Podle těchto výsledků metoda SuperGlue dosazuje výrazně lepších výsledků než FLANN based matcher.

[4] https://github.com/magicleap/SuperGluePretrainedNetwork

Lokální příznaky	Párování	P	MS
SIFT	NN + ratio test	43.4	1.7
SIFT	SuperGlue	74.1	7.2
SuperPoint	NN	22.5	4.9
SuperPoint	SuperGlue	84.8	11.1

Tabulka 3.8: SuperGlue analýza

Kapitola 4

Návrh

Kapitola obsahuje návrh výsledné aplikace, její funkční a nefunkční požadavky a případy užití. Kapitola taky obsahuje návrh struktury databáze budov a diagramy aktivit vlastního prototypu.

4.1 Funkční a nefunkční požadavky

Funkční požadavky

1. Aplikace musí běžet v reálném čase. Interval mezi nahráváním videa uživatelem a umístěním rozpoznaných budov do scény nesmí trvat déle než 3 sekundy.

2. Jádro aplikace musí kontrolovat, aby se uživatel nacházel na území České republiky.

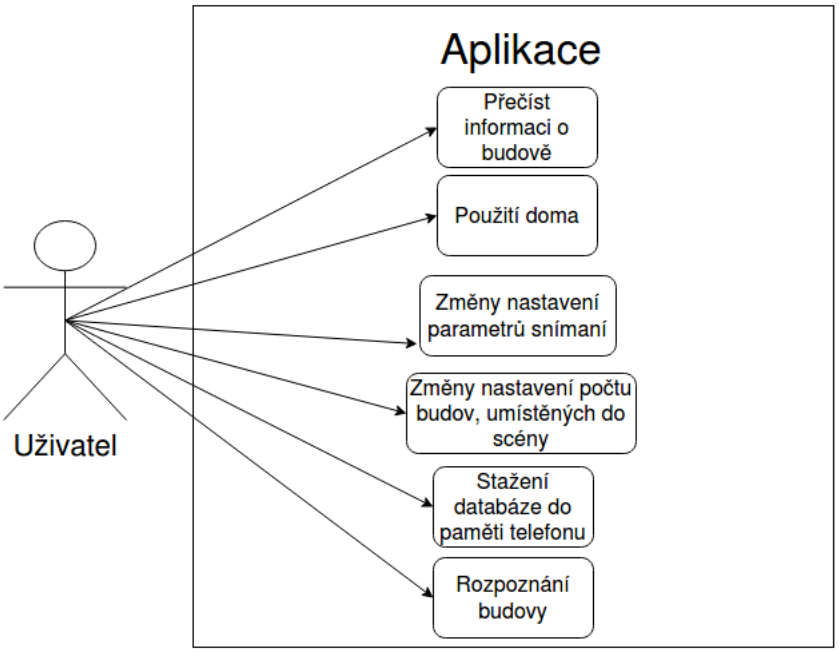
3. Jádro aplikace musí v databázi hledat budovy, které jsou viditelné přes kameru mobilního telefonu.

4. Výsledná aplikace musí uchovávat budovy v databázi podle jejích GPS souřadnic.

5. Jádro aplikace musí kontrolovat, aby každá budova v databázi měla alespoň jeden snímek každé své viditelné zdi.

6. Výsledná aplikace musí při přidávání snímků do databáze automaticky extrahovat budovy ze snímků. Tato vlastnost zrychluje čas na přidávání velkého množství snímků najednou.

7. Výsledná aplikace musí provádět výpočty pro extrahování a párování lokálních příznaků vzdáleně na serveru. Tato vlastnost optimalizuje komplikované pro mobilní telefon výpočty.



Obrázek 4.1: Diagram případů užití

Nefunkční požadavky

1. Modul aplikace pro rozpoznávání budov a umístění jejích snímku do scény musí být psán v programovacím jazyce Python.

2. Aplikace musí být konfigurovatelná. Implementace musí obsahovat konfigurační soubor, kde bude možné nastavit parametry jako například poloměr okolí uživatele nebo úhel záběru.

3. Aplikaci bude možné zlepšovat a rozšířit podle potřeb uživatele.

4. Aplikace musí budovy rozpoznávat i v noci.

4.2 Případy užití

Diagram případů užití je na obrázku 4.1. Podrobně je každý případ užití popsán níže.

1. **Prohlédnutí informací o vybraných budovách.** Uživatel si bude moci přečíst informace o budovách, které přímo uvidí před sebou. Informace budou obsahovat rok stavby a krátkou historii dané budovy.

2. **Použití doma.** Uživatel si bude moci sám vybrat budovu z databáze (podle jména, roku stavby nebo geolokace), stáhnout ji a ručně umístit její snímek do scény využitím dotykové obrazovky mobilního telefonu.
3. **Změny nastavení parametrů snímaní.** Uživatel může nastavit barevný, nebo černobílý režim snímání.
4. **Změny nastavení počtu budov umístěných do scény.** Uživatel může nastavit, chce-li vidět umístěnými do scény všechny budovy, které existují v databázi a patří do scény nebo pouze jednu budovu, která je nejlépe rozpoznána.
5. **Stažení databáze do paměti telefonu.** Uživatel bude moci stáhnout databázi nebo její část pro následující použití offline.
6. **Rozpoznání budovy.** Uživatel bude pomocí aplikace schopen vidět středověkou podobu budov ze scény. Uživatel musí mířit fotoaparátem telefonu na scénu, která bude zpracována. V případě nalezení budovy ve scéně je do ní umístěn daný snímek. Pokud aplikace najde v databázi budovu, která patřila do scény před lety, ale teď tato budova v realitě není, pak uživatel uvidí budovu na přibližném místě, na kterém stála ve středověku.

4.3 Databáze budov

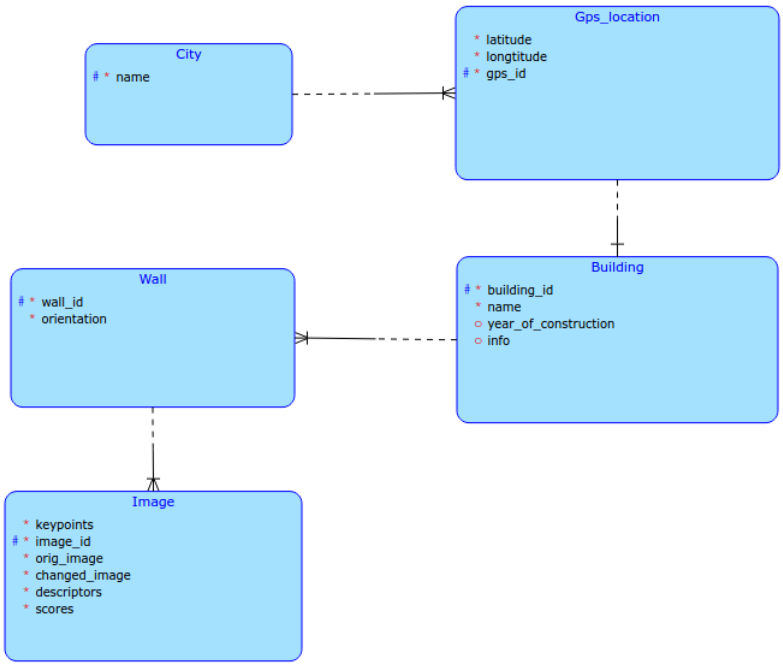
Výsledná aplikace bude běžet převážně online, ale bude možnost stáhnout databázi nebo její část lokálně. Konceptuální model databáze je představen na obrázku 4.2.

Databáze bude vypadat následovaně:

- Každá budova bude mít tyto parametry: jméno, rok založení, historickou zprávu, GPS lokaci, viditelné zdi.
- Entita GPS lokace v sobě obsahuje dvě hodnoty - longtitude a latitude.
- Každá zeď má přiřazenou orientaci a seznam obrázků. Orientace zdi je hodnota azimutu vektoru, kolmému roviny zdi směrem ven z budovy.
- Každý obrázek má přiřazeno následující: samotný snímek s metadaty, množinu deskriptorů, klíčové body a upravený snímek. Upravený snímek je obrázkem s oříznutou budovou na černém pozadí.

4.4 Vstupní data

Uživatel spustí aplikaci, nastaví základní parametry: jestli chce stáhnout snímky konkretní budovy, případně část databáze. Pak povolí přístup aplikaci k fotoaparátu mobilního telefonu a k datům o jeho geolokaci. Vstupem pro aplikaci



Obrázek 4.2: Konceptuální model databáze

bude konfigurační soubor, video snímané uživatelem v reálném čase a data o jeho geolokaci získaná z mobilních senzorů telefonu.

4.5 Rozpoznávání budov

Výsledná aplikace bude určena pro Věnná města českých královen, tato množina obsahuje 9 měst. Pro rychlejší běh aplikace bude jádro aplikace filtrovat databázi budov podle města. U každé budovy města, která je v databázi, aplikace zkontroluje, jestli její lokace může být viditelná přes fotoaparát mobilního telefonu v reálném čase. Pro tuto kontrolu bude použit postup z kapitoly Analýza.

Databáze budov obsahuje pro každou budovu seznam jejích viditelných zdí. Každá zeď má přiraženou orientaci. Díky tomu, že aplikace zaznamenává geolokační data z mobilních senzorů, a to včetně dat z mobilního kompasu, jádro aplikace vypočítá orientaci uživatele. S použitím těchto dat jádro aplikaci bude vypočítávat, která zeď budovy je nejlépe viditelná přes fotoaparát mobilního telefonu v reálném čase.

4.6 Hledání nejvhodnějších snímků

Aplikace bude každé 2 sekundy dělat záznam obrazovky. Takový časový úsek je způsoben časovou náročností pro extrahování klíčových bodů a vizualizací umístění snímku budovy do scény.

Pote bude sledovat výpočet lokálních příznaků a deskriptorů obrázku z současného záznamu pomocí metody SuperPoint.

Pak se provede postupné párování přepočítaných lokálních příznaků každého snímku vybraných budov s lokálními příznaky z obrázku současného záznamu obrazovky. Tím se najde snímek z každé budovy z databáze, který je nejvhodnější pro umístění do scény. K párování příznaků bude použita metoda SuperGlue.

4.7 Umístění snímku budovy do scény

Díky tomu, že při párování lokálních příznaků byla použita neupravená verze snímku budovy, program rozpoznává nejen budovu zvlášť, ale i okolí této budovy. Program může umístit do scény i ty budovy, které již v realitě neexistují, pokud bude dostatečný počet shod mezi vstupní scénou a okolím budovy z databáze.

Dále postupně pro každý snímek, který je výsledkem předchozího kroku, budou provedeny následující kroky:

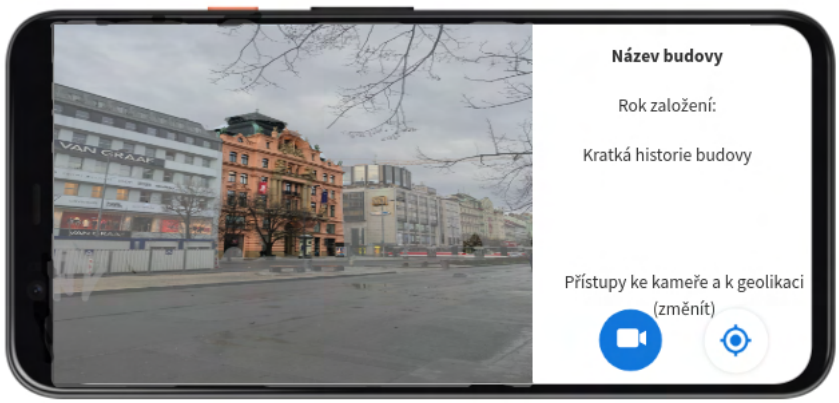
1. Za prvé bude vypočítána transformační matice. Pomocí této matice bude možné deformovat snímek z databáze a přesně umístit do scény.
2. Dále se pomocí funkce warpPerspective z knihovny OpenCV vytvoří nový transformovaný obrázek.
3. Pomocí segmentace upravené verze snímku jsou vytvořeny masky budovy a reálné scény, které poté umožní složit snímky dohromady.
4. Pro výsledný složený snímek a další snímek vhodný pro umístění bude proveden krok 1. Totéž bude pokračovat, dokud neskončí snímky vhodné pro umístění.

4.8 Vzhled modulu

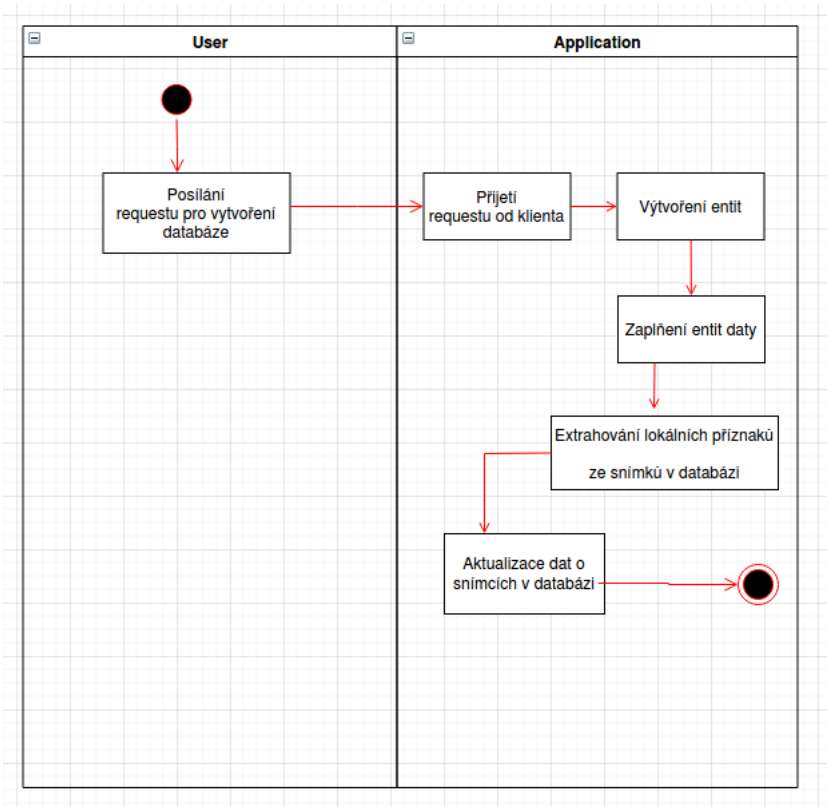
Na obrázku 4.3 je ukázán předpokládaný vzhled aplikace.

4.9 Diagramy aktivit

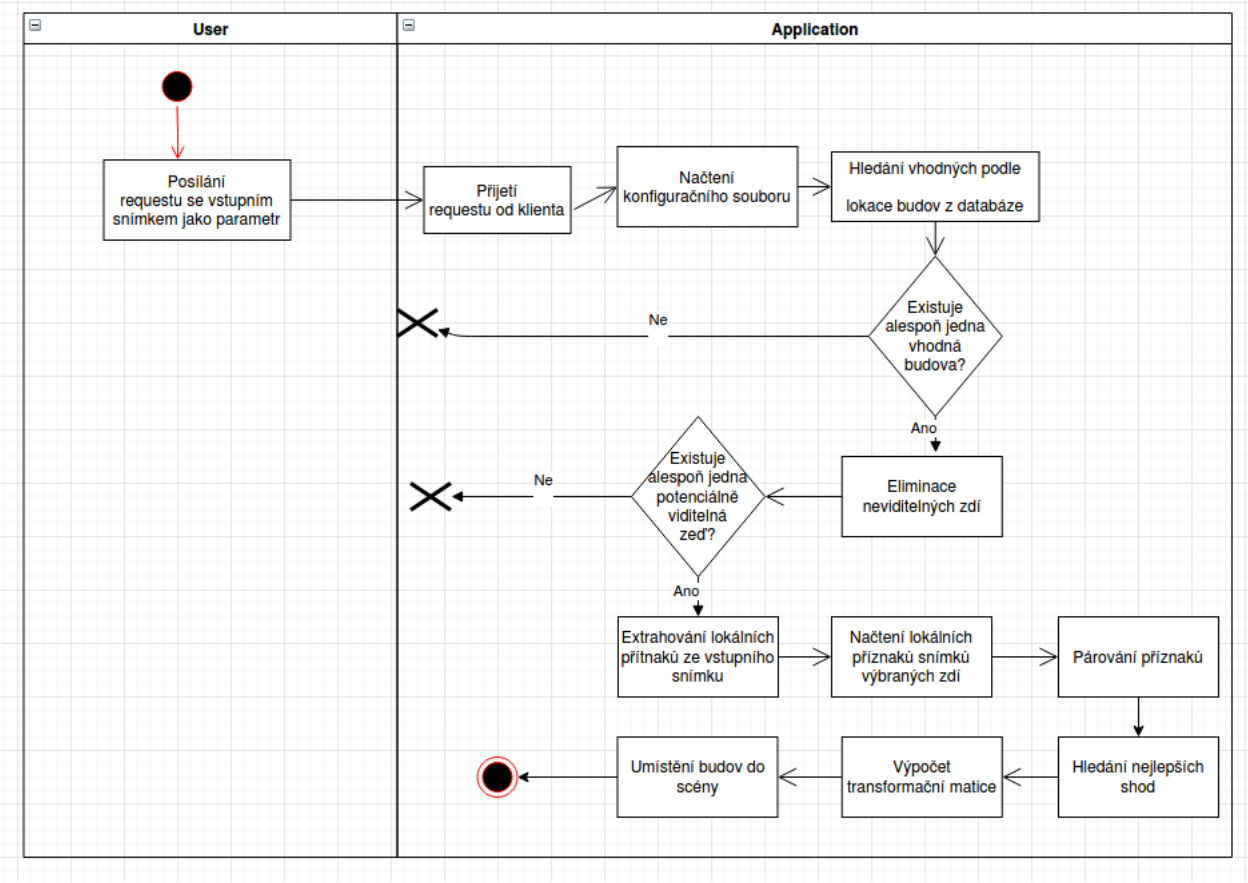
V této sekci jsou uvedeny dva diagramy aktivit pro vlastní prototyp. První diagram 4.4 ukazuje běh aplikaci pro přípravu databáze. Druhý diagram 4.5 ukazuje průběh samotné aplikace.



Obrázek 4.3: Vzhled navržené aplikace



Obrázek 4.4: Diagram aktivit: příprava dat



Obrázek 4.5: Diagram aktivit: běh aplikace

Kapitola 5

Implementace

V této kapitole je popsána implementační část bakalářské práce.

5.1 Použité technologie

- **PyTorch** - knihovna pro učení a implementaci modelů hlubokého učení [26]. V této práci se v převzatých implementacích SuperGlue a SuperPoint používá.
- **OpenCV** - knihovna pro účely strojového vidění [20].
- **Flask** - framework pro webové aplikace. V této práci se používá pro realizaci komunikace typu „klient-server“.
- **psycopg2** - knihovna pro snadnou práci s databází PostgreSQL v jazyce Python.
- **math** - knihovna pro různé matematické výpočty. V této práci se používá pro zjištění, jestli budova z databáze, dle její geolokace, může být viditelná na vstupním snímku.
- **NumPy** - knihovna pro náročné matematické výpočty [27].
- **Pillow** - knihovna pro práci s obrázky. Podporuje jednoduchou práci s metadaty snímků [28].

5.2 Konfigurační soubor

Prototyp používá dva konfigurační soubory. První soubor obsahuje implicitní nastavení parametrů, které může uživatel dle potřeby změnit.

- **image.** Obsahuje nastavení pro předzpracování snímků.

- **geolocation.** Obsahuje konstanty pro výpočty geolokací.
- **matching.** Obsahuje konstantu pro eliminaci špatných párování.

Druhý obsahuje nastavení hyperparametrů modelů, tyto hodnoty jsou podle experimentů autorů metod považovány za nejlepší.

- **superpoint.** Obsahuje parametry pro nastavení metody SuperPoint. Například počet extrahovaných klíčových bodů.
- **superglue.** Obsahuje konstanty k nastavení metody SuperGlue. Například práh pro párování klíčových bodů.

5.3 Testovací datová množina

Pro testování výsledného řešení byl vytvořen vlastní dataset snímků. Dataset obsahuje 15 snímků, v Příloze je uveden krátký popis každého z nich. Na některých snímcích je znázorněno několik budov z databáze pro testování schopnosti prototypu rozpoznat a umístit více budov najednou. Snímky byly provedeny za různých podmínek osvětlení, některé byly vytvořeny v noci. Dataset obsahuje i špatné příklady pro testování schopnosti eliminace všech budov z databáze a demonstraci výjimek a chyb.

5.4 Databáze budov

Pro účely projektu Věnná města českých královen a pro testování implementovaného prototypu byl vytvořen dataset nasnímaných budov Prahy. Datová množina obsahuje 50 budov.

Struktura databáze budov odpovídá popisu databáze z návrhu. Data jsou uložena pomocí PostgreSQL [29]. PostgreSQL je široce používaná v mobilních a webových aplikacích databáze. Má otevřený přístup, t.j. je OpenSource. Komunikace mezi jazykem Python a PostgreSQL probíhá pomocí knihovny psycopg2. Knihovna psycopg2 usnadňuje posílání SQL (Structured Query Language) dotazů do databáze.

V dané struktuře má každá budova zadané GPS souřadnice. Získat standardizované GPS souřadnice je velký problém. Nenašla jsem žádnou databázi, která by obsahovala GPS souřadnice budov Prahy. Z toho důvodu jsem ručně získala přibližné GPS souřadnice geometrického středu každé budovy z databáze. Na to jsem použila aplikaci Google Maps[5].

Některé budovy mají přidělený rok postavení a krátký popis, převzatý z internetové encyklopedie Wikipedia[6]. To ale mají pouze historické budovy.

[5] https://www.google.com/maps
[6] https://www.wikipedia.org/

Orientace každé zdi byla naměřena ručně, přiložením kompasu ke zdi a uložením naměřené hodnoty do databáze.

Každá budova obsahuje minimálně 1 snímek z každé své viditelné zdi. V průměru je každá budova ofocena 3krát. Vytvořená databáze obsahuje celkem 175 snímků.

5.5 Vstupní data

Na rozdíl od návrhu pro realizaci tohoto prototypu nebyla implementována podpora pro běh v reálném čase. Kvůli tomu není vstupem do programu video, snímané uživatelem v reálném čase, ale snímek vnější scény. Data o poloze uživatele (směr a GPS souřadnice) jsou získána z metadat snímku.

5.6 Rozpoznání budov

Jak bylo uvedeno výše, prototyp aplikace získává informace o geolokaci a směru kamery z metadat vstupního snímku. Ostatní implementace části rozpoznání budov odpovídá návrhu. Implicitní hodnota pro úhel záběru fotoaparátu je nastavena na 140°. Lze ji změnit v konfiguračním souboru. Validní hodnota musí být větší nebo rovna 140°. Důvodem pro omezení okolí úhlem záběru je zmenšení počtu spuštění metody na párování klíčových bodů, a díky tomu zrychlení doby běhu aplikace. Proto nemusí tato hodnota přesně odpovídat úhlu záběru fotoaparátu používaného telefonu, ale neměla by být méně než reálná hodnota.

Většina ze současně používaných mobilních telefonů má kameru s úhlem záběru 120° a více. Kvůli tomu, že někdy nejsou geolokační data z metadat přesná, je doporučená implicitní hodnota pro úhel záběru 140°. Je nutné uvést, že čím je hodnota nastavená v konfiguračním souboru vyšší, tím se bude porovnávat více budov z databáze se vstupním snímkem a tím déle bude trvat výpočet. Žádná kamera z mobilního telefonu nemá úhel záběru větší než 180°. Proto je validní hodnota pro tento parametr omezena zprava na 180° včetně.

Implicitní hodnotu pro poloměr okolí jsem zvolila na 250 metrů podle analýzy z práce [19]. Tuto hodnotu je možné měnit v konfiguračním souboru.

U vybraných budov databáze, které leží uvnitř kruhové výseče s nastavenými parametry, program najde zdi, které mohou být viditelné na vstupním obrázku. Zeď může být viditelná, pokud není stejným směrem, kterým byl vytvořen vstupní obrázek.

Například nechť uživatel ofotil budovu, hodnota azimutu ze vstupního snímku je 100°. Při použití implicitní hodnoty úhlu záběru není zeď budovy viditelná, pokud je její orientace v rozsahu od 30° do 170°.

Postup pro rozpoznávání scény na základě geolokačních dat

V této části je popsán vlastně navrženy postup pro kontrolu, jestli je budova může vyskytovat na snímku, nebo ne podle geolokačních dat.

Na obrázku 5.1 jsou schematicky ukázány označení úhlů, které jsou nutnými pro implementaci daného postupu. Úhel β je úhel záběru „radiusy“ jsou hranice kruhové výseči, γ je úhlem mezi severem, uživatelem a levou hranicí, α je úhlem mezi severem, uživatelem a budovou, θ je úhlem mezi severem, uživatelem a pravou hranicí. Úhel β je daný v konfiguračním souboru. Z metadat je známy azimut uživatele.

1. Kontrola, jestli je vzdálenost mezi budovou a uživatelem je menší, nebo rovná danému poloměru.
2. Výpočet úhlů γ a θ jako azimut uživatele $\pm\frac{\beta}{2}$.
3. Výpočet úhlu α jako úhel mezi severem, GPS souřadnicemi uživatele a GPS souřadnicemi budovy.
4. Kontrola, jestli α leží mezi γ a θ. A to i v případě, když $\gamma > \theta$.

Pokud všechny 4 kroky vrací True, pak budova je považována za potenciálně viditelnou.

5.7 Hledání nejvhodnějších snímků

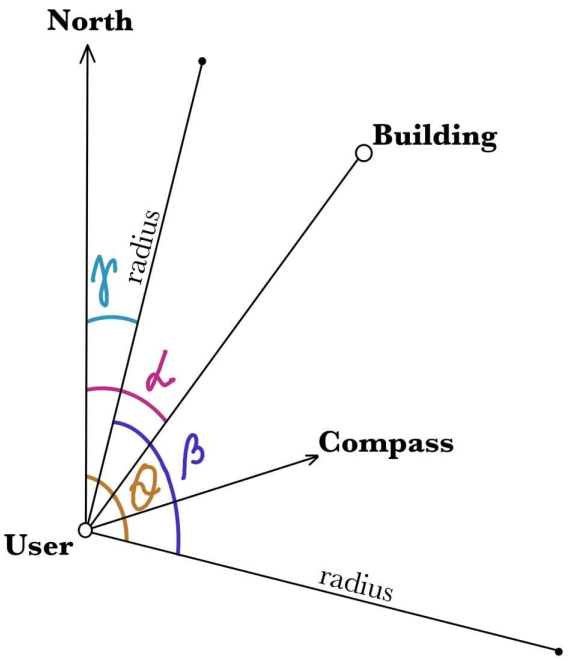
Na začátek je vstupní obrázek oříznut do velikosti 640×480 pro zrychlení výpočtu. Referenční velikost lze změnit v konfiguračním souboru. Ale dle autorů metody SuperGlue obrázky velikosti menší než 160×120 nebo větší než 2000×1500 nejsou doporučeny jako vstup do programu.

Ze vstupního snímku budou metodou SuperPoint extrahovány klíčové body a vypočteny deskriptory. Hodnota NMS (Non Maximum Suppression) je implicitně nastavena na 4, práh pro detektor klíčových bodů je nastaven na 0.005, maximální počet extrahovaných klíčových bodů je roven 1024. Všechny tyto hodnoty byly autory metody SuperPoint považovány jako nejlepší.

Pomocí metody SuperGlue byly lokální příznaky spárovány s příznaky vybraných snímků. Podle počtu nalezených shod byly nalezeny nejvhodnější snímky pro každou z budov, která je zároveň na vstupním snímku i v databázi. Počet iterací Sinkhorn algoritmu je implicitně nastaven na 20, hodnota prahu pro párování je rovna 0.2. Tyto hodnoty byly považovány autory metody SuperGlue jako nejlepší.

5.8 Umístění snímku budovy do scény

Výsledkem předchozího kroku je seznam snímků vhodných pro umístění. Postup pro umístění snímku do scény úplně odpovídá postupu z kapitoly Návrh.



Obrázek 5.1: Schéma umístění budovy

Výsledkem tohoto kroku je vnější scény s budovami, které jsou do ní umístěny.

5.9 Průběh aplikace

Prototyp aplikace simuluje komunikaci typu „klient-server“. Komunikace je realizována pomocí frameworku Flask. Uživatel může posílat dva druhy requestu do serveru.

První druh requestu se používá při spuštění aplikace nebo po libovolných změnách v konfiguračním souboru. Server po přijetí prvního druhu requestu vytvoří databázi a naplní ji daty pro každou entitu. Dále spouští metodu SuperPoint pro každý snímek z databáze. Metoda extrahuje klíčové body, jejich deskriptory a pole hodnot „Score“ velikosti jako obrázek, každá hodnota z pole „Score“ je pravděpodobnost, že je na příslušném místě na obrázku klíčový bod. Získané výsledky server zapíše do příslušných položek databáze.

Druhý druh requestu má jako parametr soubor s obrázkem s metadaty. Server po přijetí druhého druhu requestu postupně prochází kroky 5.7-5.9 z kapitoly Implementace. Jako výstup vrátí výsledný obrázek.

Kapitola 6

Testování

V této kapitole je otestován výsledný prototyp na vlastní testovací množině, která je popsána v kapitole Implementace. Hlavním cílem této kapitoly je porovnání stávajícího řešení [19] s řešením vlastním. Pro testování stávajícího řešení bylo použito implicitní nastavení konfiguračního souboru. Změnila jsem pouze databázi na vlastně vytvořenou.

Vzhledem k tomu, že nastavení hyperparametrů metod SuperGlue a SuperPoint bylo považováno autory těchto metod za nejlepší podle výsledků experimentů, pak jsem testovala prototyp s implicitními hodnotami v konfiguračním souboru. Konfigurační soubory vlastního prototypu a jejich nastavení jsou popsány v kapitole Implementace.

6.1 Technické limitace prototypu

Výsledky rozpoznávání budov nejsou závislé na počasí či denní době. Prototyp má limitace ohledně technických charakteristik serveru, na kterém běží.

Server s technickými charakteristikami popsanými v kapitole Analýza má pouze jedno staré GPU, na němž se ten prototyp nespustil kvůli překročení limitu paměti CUDA. CUDA je programem, který umožňuje provádět paralelní výpočty na GPU, což mnohem zrychluje dobu běhu těchto výpočtů.

Prototyp může běžet i na CPU, ale je v tomto případě pomalejší. Otestovala jsem vlastní prototyp na CPU, střední doba běhu pro každý snímek z testovací množiny ořízntý na 960 pixelů je 18 vteřin. Nemůže taková rychlost být použita pro aplikaci v reálném čase.

Výsledná aplikace podle návrhu poběží převážně online, to znamená a všechny výpočtu se budou provádět na vzdáleném serveru. Pro účely testování vlastního prototypu jsem použila vzdálený server (k tomu jsem použila servis Vast.ai[7])s jedním jádrem GPU NVIDIA RTX 2070S, ten je mnohem výkonnější než GPU NVIDIA GM108M [GeForce 840M]. Následující testování

[7] https://vast.ai/

	Vlastní (CPU)	Vlastní (GPU)	Stávající
960 pixelů	282.42	52.56	169.23
2000 pixelů	-	56.06	525.12

Tabulka 6.1: Testování rychlosti přípravy databáze

vlastního prototypu na GPU proběhlo na daném serveru. Takže je potřeba upřesnit, že pouze modely SuperPoint a SuperGlue běží na GPU, ostatní výpočty probíhají na CPU.

6.2 Testování časové náročnosti pro přípravu databáze

Před základním spuštěním vlastního prototypu se vstupním snímkem jako parametr je potřeba extrahovat lokální příznaky z každého snímku a uložit je do příslušné položky databáze. A to platí i pro stávající řešení. V této sekci je analyzována celková doba běhu pro extrakci lokálních příznaků ze 175 snímků vnějších scén.

Časová náročnost byla naměřena pro obrázky oříznuté do velikosti 960 pixelů a pak pro obrázky oříznuté do velikosti 2000 pixelů. Pro měření času byla použita knihovna „time“ pro jazyk Python. Čas je naměřen v sekundách. Vlastní prototyp jsem otestovala podle rychlosti i na CPU Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz. Výsledky jsou v tabulce 6.1. Podle výsledků testování použití metody SuperPoint je vhodným řešením pro aplikaci reálného času. Metoda SuperGlue je pomalejší, než FLANN based matcher, který byl použit v již stávajícím prototypu. Výhodou metody SuperGlue je přesnější rozpoznávání budov v nezávislosti na denní době a počasí. Takže je nutné uvést omezení pro používání vlastního postupu. Pokud server, na kterém poběží aplikace, nemá GPU, pak aplikace nebude běžet pro snímky velikosti větší než 960 pixelů kvůli překročení paměťové náročnosti.

Čas běhu vlastního prototypu není závislý na velikost obrázků.

6.3 Testování rychlosti běhu aplikace

Testování proběhlo pro obrázky o velikosti 2000 pixelů, aby byla vstupní data byly podobná vstupním datům navržené aplikace reálného času podle velikosti.

Pro každý snímek z testovací množiny byly naměřeny následující parametry:

- **Doba běhu prototypu.** Pro porovnání časové náročnosti prototypu je naměřena doba běhu pro každý vstupní snímek se zadaným kon-

figuračním souborem. Výsledný čas udává počet vteřin potřebných k načtení vstupního snímku, jeho zpracování, rozpoznání budov a jejich následné umístění do vstupních snímků.

- **Počet budov z databáze znázorněných na snímku.** Některé snímky z testovacího datasetu byly vytvořeny tak, aby v sobě obsahovali několik budov z databáze pro testování schopnosti umístění několika budov do scény. V testovacím datasetu existují i snímky, na nichž žádná budova z databáze není.
- **Počet budov rozpoznaných prototypem.** Stejně jako předchozí parametr, i tento parametr se používá pro testování schopnosti umístit více budov do scény.

V tabulce 6.2 jsou uvedeny výsledky testování stávajícího řešení. V tabulce 6.3 jsou uvedeny výsledky testování vlastního řešení. V místech s vysokou hustotou budov v okolí (v rámci databáze) je řešení Jana Šefčíka rychlejší, v opačném případě je vlastní řešení rychlejší. Stávající řešení je schopné umístit maximálně jednu budovy do scény, vlastní řešení do scény umístí všechny budovy, které rozpozná ve scéně a najde jejich podobu v databázi.

V některých testovacích příkladech je možné vidět, že vlastní prototyp umístí více budov než kolik scéna obsahuje. Je to z toho důvodu, že párování probíhá mezi vstupním snímkem a originálními snímky z databáze, ne jejich upravenými verzemi. Upravené verze se používají pouze pro získaní masky a umístění do scény. Pokud nějaká budova z databáze není na vstupním snímku, ale její okolí v něm je, pak prototyp umístí tuto budov tam, kam by měla patřit. Taková vlastnost může být využita pro účely projektu. Prototyp je schopen umístit do scény i neexistující budovu, pokud rozpozná její okolí.

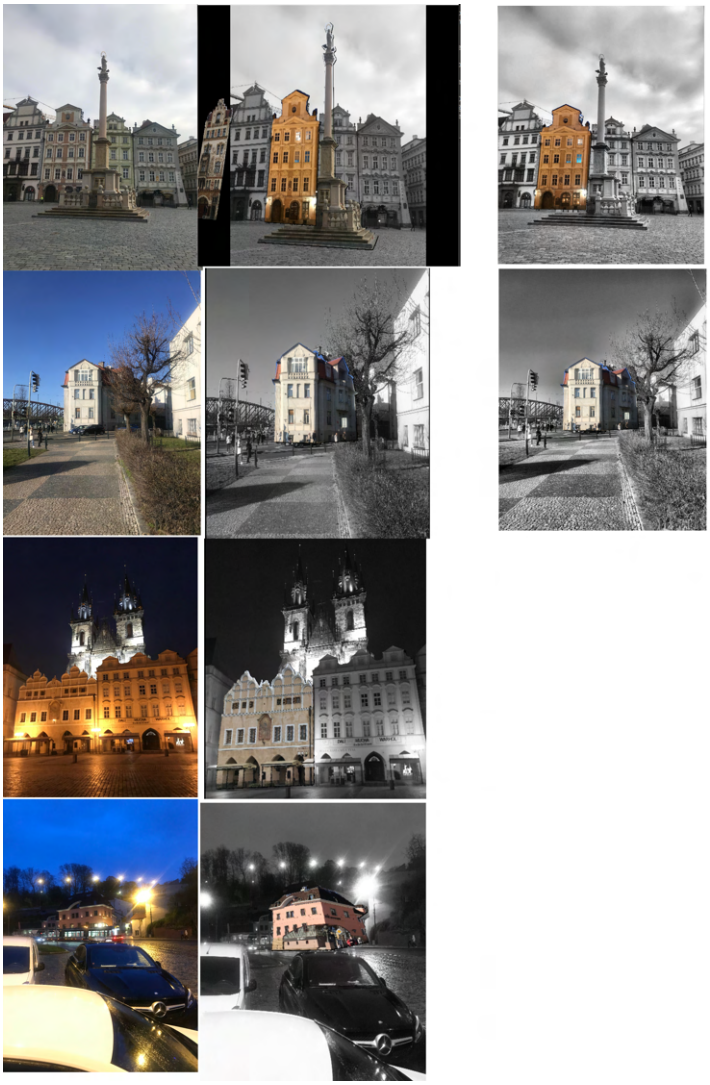
Vlastní prototyp je navíc schopen budovy rozpoznávat i v noci, což stávající prototyp neumí.

Několik příkladů z testování lze vidět na obrázku 6.1. Snímky 3 a 6 z testovací množiny nebyly rozpoznány již existujícím prototypem. Všechny testy jsou znázorněny v Příloze D.

6.4 Testování přesnosti rozpoznávání a umístění

Pro ohodnocení přesnosti umístění budovy do scény bylo použito následující označení:

- **[1]:** kontury umístěné budovy přesně překrývají kontury budovy ze scény.
- **[2]:** budova je dobře umístěna, ale někdy jsou viditelné kontury budovy ze scény přes umístěnou budovu.
- **[3]:** budova je umístěna, ale velká část budovy ze scény je viditelná přes umístěnou budovu.



Obrázek 6.1: Výsledky testování: zleva doprava originál testovaného snímku, výsledek vlastního prototypu, výsledek prototypu Jana Šefčíka. shora dolů: snímek 1, snímek 2, snímek 3, snímek 6 z testovací množiny.

Snímek	# budov na snímku	# rozpozn. budov	Doba běhu CPU 960p.	Doba běhu CPU 2000p.
1	2	1	5.1	6.3
2	1	1	3.3	4.2
3	1	0	5	6.3
4	1	1	3.8	4.5
5	1	1	3.5	4.4
6	1	0	3.3	4.4
7	1	0	0.03	0.03
8	1	1	3.4	4.5
9	1	0	3.4	4.5
10	0	1	3.5	4.5
11	2	1	5.1	6.4
12	1	1	3.4	4.4
13	1	1	3.4	4.5
14	0	0	0.03	0.03
15	0	0	3.7	4.6

Tabulka 6.2: Testování existujícího řešení: rychlost

Snímek	# budov na snímku	# rozpozn. budov	Doba běhu CPU 960p.	Doba běhu GPU 960p.	Doba běhu GPU 2000p.
1	2	3	43.723	11.4	11.5
2	1	1	15.8	4.4	4.4
3	1	2	32.9	10.8	10.8
4	1	1	11.79	3.5	3.6
5	1	1	14.27	3.8	3.9
6	1	1	9.5	3.4	3.4
7	1	0	0.07	0.07	0.08
8	1	1	16.05	4.4	4.4
9	1	1	8.96	3.4	3.5
10	0	1	17.4	4.5	4.5
11	2	2	27.04	10.1	10.1
12	1	1	14.2.	4.6	4.7
13	1	1	15.2	4.6	4.6
14	0	0	0.1	0.1	0.1
15	0	1	14.2	4.6	4.7

Tabulka 6.3: Testování vlastního řešení: rychlost

Snímek	Umístěná budova	Umístění
1	14	2
2	3	1
3	13	X
4	7	2
5	21	2
6	24	X
7	34	X
8	12	2
9	8	X
10	11	4
11	14	1
12	29	4
13	45	2

Tabulka 6.4: Testování existujícího řešení: přesnost

- **[4]:** budova je rozpoznána na vstupní scéně, ale neumístěna do výsledné scény, pokud se nedá z výsledné scény rozpoznat, jak umístěná budova vypadá.
- **[N/S]:** budova nepatří do vstupního snímku, ale na základě porovnání okolí byla správně umístěna tam, kam by měla patřit.
- **[N/N]:** budova nepatří ani do vstupního snímku ani do scény, ale na základě porovnání okolí byla špatně umístěna.
- **[X]:** budova nebyla rozpoznána.

V tabulce 6.4 jsou uvedeny výsledky testování stávajícího řešení. V tabulce 6.5 jsou uvedeny výsledky testování vlastního řešení. Prototyp Jana Šefčíka přesně umístí budovy pro snímky vytvořené v denní době a dobře získává snímky, které neobsahují žádnou budovu z databáze. Vlastní prototyp dobře umístí budovy i za komplikovaných podmínek osvětlení.

Snímek	Umístěná budova	Umístění
1	5	1
1	14	1
1	36	N/S
2	3	1
3	13	1
3	35	N/S
4	7	1
5	21	2
6	24	3
7	34	X
8	12	1
9	8	1
10	11	N/S
11	5	N/N
11	14	1
12	29	1
13	45	2
15	45	N/S

Tabulka 6.5: Testování vlastního řešení: přesnost

Závěr

Tato práce se zabývá řešením problému pro rozpoznávání a editaci urbanistické scény pomocí moderních metod hlubokého učení. Cílem této bakalářské práce bylo navrhnout a implementovat prototyp na základě analýzy s ohledem na omezení projektu Věnná města českých královen. Analýza obsahuje možnosti výpočtu lokálních příznaků pomocí metod hlubokého učení, jejich následného párování a popis různých geolokačních nástrojů, které využívají moderní mobilní telefony. Jedním z cílí této práce bylo rozšíření stávající databáze budov. Jedním z výsledků práce je rozšířená databáze budov Prahy. Obsahuje 50 budov a 175 snímků, vytvořených z různých úhlů pohledů a za různých podmínek osvětlení.

Pro implementaci prototypu byl použit jazyk Python společně s knihovnami pro práci s obrazem, např. OpenCV, a knihovnou pro práci s modely hlubokého učení PyTorch. Prototyp byl následně otestován na vytvořeném testovacím datasetu dle přesnosti a rychlosti. Zároveň byl porovnáván s již existujícím řešením, které využívá metody klasického strojového vidění. Testování potvrdilo, že modely naučené na velkém množství dat hlubokého učení lépe rozpoznávají budovu ze scény a párují dvojice různých obrázků stejné vnější scény, a to i pro snímky, které byly vytvořeny v noci. Nejlepších výsledků bylo dosaženo s použitím metody SuperPoint pro hledání lokálních příznaků a metody SuperGlue při jejich následném párování. Metoda SuperPoint podle výsledků testování byla považována za vhodnou pro aplikaci reálného času.

Cíle této práce byly splněny. Práce tvoří část většího projektu a její výsledky už se používají v jedné bakalářské práci. Do budoucna lze uvažovat také o používání segmentace pomocí neuronových sítí pro automatický ořez budovy ze snímků v databázi, což zrychluje přidávání nového snímku do databáze.

Bibliografie

[1] Barbara Zitová a Jan Flusser. „Image registration methods: a survey“. In: *Image and Vision Computing* (2003). [cit. 2021-03-04], s. 977–1000. ISSN: 0262-8856. DOI: https://doi.org/10.1016/S0262-8856(03)00137-9. URL: https://www.sciencedirect.com/science/article/pii/S0262885603001379.

[2] Keiron O'Shea a Ryan Nash. „An Introduction to Convolutional Neural Networks“. In: *In: CoRR* (2015). [cit. 2020-12-04]. URL: http://arxiv.org/abs/1511.08458.

[3] Vinod Nair a Geoffrey E. Hinton. „Rectified Linear Units Improve Restricted Boltzmann Machines“. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, s. 807–814. ISBN: 9781605589077.

[4] Ian Goodfellow, Yoshua Bengio a Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[5] Lorenzo Rosasco et al. „Are Loss Functions All the Same?“ In: *Neural Computation* 16.5 (květ. 2004). [cit. 2021-04-11], s. 1063–1076. ISSN: 0899-7667. DOI: 10.1162/089976604773135104. eprint: https://direct.mit.edu/neco/article-pdf/16/5/1063/815882/089976604773135104.pdf. URL: https://doi.org/10.1162/089976604773135104.

[6] Sergey Ioffe a Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. Francis Bach a David Blei. Sv. 37. Proceedings of Machine Learning Research. [cit. 2021-05-01]. Lille, France: PMLR, čvc 2015, s. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[7] Jianguo Zhang et al. „Local features and kernels for classification of texture and object categories: A comprehensive study“. In: *International journal of computer vision* 73.2 (2007). [cit. 2021-05-11], s. 213–238.

[8] D. G. Lowe. „Object recognition from local scale-invariant features“. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Sv. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[9] Daniel DeTone, Tomasz Malisiewicz a Andrew Rabinovich. *SuperPoint: Self-Supervised Interest Point Detection and Description*. [cit. 2021-03-20]. 2018. arXiv: 1712.07629 [cs.CV].

[10] Yuki Ono et al. *LF-Net: Learning Local Features from Images*. [cit. 2021-05-11]. 2018. arXiv: 1805.09662 [cs.CV].

[11] Mihai Dusmanu et al. *D2-Net: A Trainable CNN for Joint Detection and Description of Local Features*. [cit. 2021-03-20]. 2019. arXiv: 1905.03561 [cs.CV].

[12] *Feature Matching. OpenCV*. [cit. 2021-05-11]. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html.

[13] Paul-Edouard Sarlin et al. *SuperGlue: Learning Feature Matching with Graph Neural Networks*. 2020. arXiv: 1911.11763 [cs.CV].

[14] Gabriel Peyré a Marco Cuturi. *Computational Optimal Transport*. [cit. 2021-05-11]. 2020. arXiv: 1803.00567 [stat.ML].

[15] James Munkres. „Algorithms for the Assignment and Transportation Problems“. In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957). [cit. 2021-03-20], s. 32–38. DOI: 10.1137/0105003. eprint: https://doi.org/10.1137/0105003. URL: https://doi.org/10.1137/0105003.

[16] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[17] Adam Paszke et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Ed. H. Wallach et al. [cit. 2021-05-11]. Curran Associates, Inc., 2019, s. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] Nicholas D. Lane et al. „A survey of mobile phone sensing“. In: *IEEE Communications Magazine* 48.9 (2010). [cit. 2021-03-20], s. 140–150. DOI: 10.1109/MCOM.2010.5560598.

[19] Jan Šefčík. „Rozpoznávání a editace urbanistické scény. České vysoké učení technické v Praze, Fakulta informačních technologií“. [cit. 2021-05-11]. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií. Praha, 2020.

[20] G. Bradski. „The OpenCV Library“. In: *Dr. Dobb's Journal of Software Tools* (2000). [cit. 2021-05-11].

[21] Martin A. Fischler a Robert C. Bolles. „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography“. In: *Commun. ACM* 24.6 (červ. 1981). [cit. 2021-05-11], s. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[22] K. Mikolajczyk a C. Schmid. „A performance evaluation of local descriptors“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005). [cit. 2021-05-11], s. 1615–1630. DOI: 10.1109/TPAMI.2005.188.

[23] Yuhe Jin et al. „Image Matching Across Wide Baselines: From Paper to Practice“. In: *International Journal of Computer Vision* 129.2 (říj. 2020). [cit. 2021-05-11], s. 517–547. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01385-0. URL: http://dx.doi.org/10.1007/s11263-020-01385-0.

[24] Paul-Edouard Sarlin et al. „SuperGlue: Learning Feature Matching with Graph Neural Networks“. In: *CVPR*. [cit. 2021-05-11]. 2020. URL: https://arxiv.org/abs/1911.11763.

[25] David G. Lowe. „Distinctive Image Features from Scale-Invariant Keypoints“. In: *Int. J. Comput. Vision* 60.2 (lis. 2004). [cit. 2021-05-11], s. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[26] Adam Paszke et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Ed. H. Wallach et al. [cit. 2021-05-10]. Curran Associates, Inc., 2019, s. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[27] Charles R. Harris et al. „Array programming with NumPy“. In: *Nature* 585.7825 (zář. 2020). [cit. 2021-05-11], s. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[28] *Pillow*. [cit. 2021-05-11]. URL: https://pillow.readthedocs.io/en/stable/.

[29] *Postgres*. [cit. 2021-05-11]. URL: https://www.postgresql.org/.

Příloha **A**

Seznam použitých zkratek

GPS Gloval positioning system

SIFT Scale invariant feature transform

CNN Convolutional neural networks

DoG Laplacian of Gaussian

NMS Non Maximum Supression

BF Brute Force

FLANN Fast Library for Approximate Nearest Neighbors

RANSAC Random Sample Consensus

GPU Graphics processing unit

ReLU Rectified Linear Unit

Příloha B

Contents of enclosed CD

- `readme.txt` ..... popis obsahu média a návod pro instalaci a použití
- `UrbanSceneRecognition/` ..... zdrojové kódy
  - `analysis/` ..... zdrojové kódy analýzy
  - `database/` ..... vlastní databáze
- `thesis/` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `src/` ..... zdrojová forma práce ve formátu LaTeX

Příloha C

Testovací množina

Snímek č.1

Snímek byl vytvořen vytvořen v době, kdy bylo zataženo. Uprostřed snímku se nachází budova číslo 5 z databáze (Mariánský sloup). Na snímku lze také vidět budovu číslo 14 z databáze.

Snímek č.2

Snímek byl vytvořen za bezoblačného počasí. Budova číslo 3 je ve vzdálenosti 100 metrů od kamery.

Snímek č.3

Snímek byl vytvořen v noci. Na snímku je umístěna budova 13.

Snímek č.4

Snímek byl vytvořen v noci. Na snímku je umístěna budova 7.

Snímek č.5

Snímek byl vytvořen za bezoblačného počasí. Na snímku je za bílou budovou umístěna budova 21.

Snímek č.6

Snímek byl vytvořen v noci. Budova číslo 24 je ve vzdálenosti 130 metrů od kamery. V popředí snímku je několik aut.

Snímek č.7

Snímek byl vytvořen v denní době. Budova číslo 34 je ve vzdálenosti 250 metrů od kamery.

Snímek č.8

Snímek byl vytvořen v noci. Na snímku je umístěna budova 12.

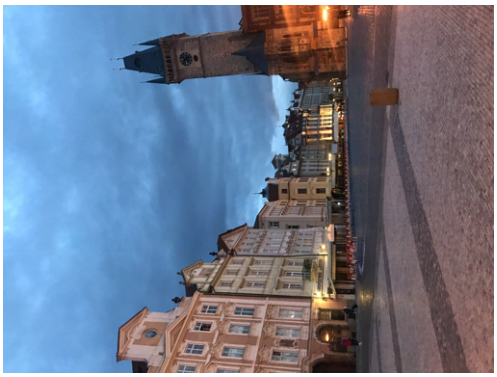
Snímek č.9

Budova číslo 8 byla ofocena v noci, ale je dobře osvětlena pouličními světly.

Snímek č.10

Snímek byl vytvořen v denní době. Neobsahuje žádnou budovu z databáze, ale zleva je na této ulici umístěna budova 11.

Snímek č.11



Snímek byl vytvořen pozdě večer, obsahuje budovu číslo 14 zleva a budovu číslo 6 ve vzdálenosti 200 metrů.

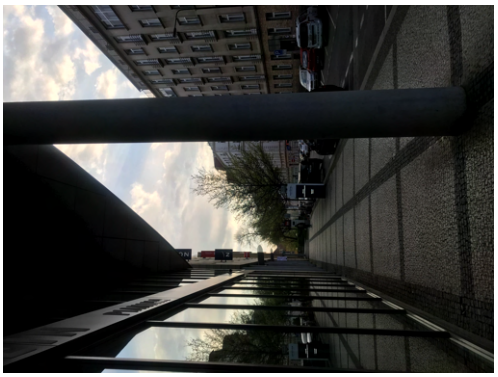
Snímek č.12

Snímek byl vytvořen v době, kdy bylo zataženo. Na snímku je za bílou budovou umístěna budova 29.

Snímek č.13

Snímek byl vytvořen v době, kdy bylo zataženo. Na snímku je za bílou budovou umístěna budova 45 ve vzdálenosti 50 metrů.

Snímek č.14



Snímek byl vytvořen v denní době. Neobsahuje žádnou budovu z databáze.

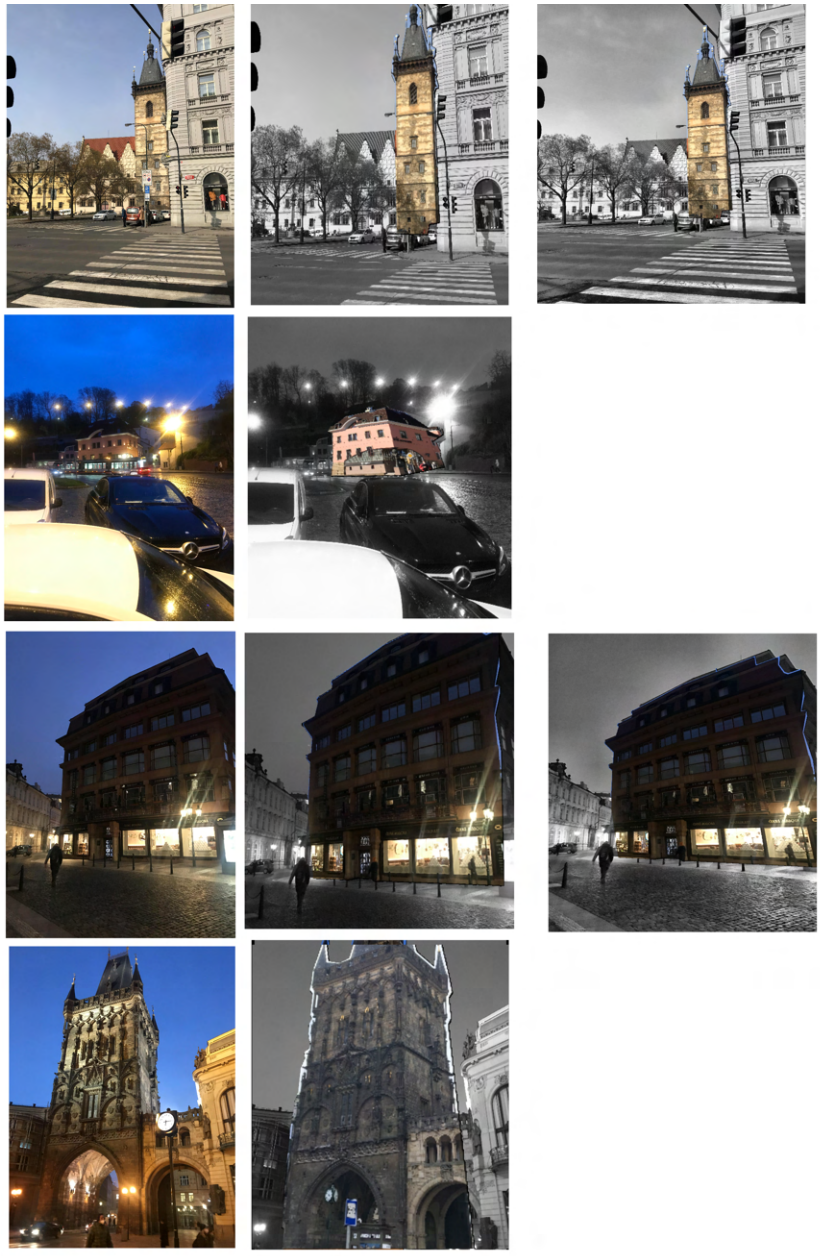
Snímek č.15

Snímek byl vytvořen v době, kdy bylo zataženo. Neobsahuje žádnou budovu z databáze, ale zprava je na této ulici umístěna budova 50.

Příloha D

Výsledky testování

Obrázek D.1: Výsledky testování: zleva doprava originál testovaného snímku, výsledek vlastního prototypu, výsledek prototypu Jana Šefčíka. shora dolů: snímek 1, snímek 2, snímek 3, snímek 4 z testovací množiny.



Obrázek D.2: Výsledky testování: zleva doprava originál testovaného snímku, výsledek vlastního prototypu, výsledek prototypu Jana Šefčíka. shora dolů: snímek 5, snímek 6, snímek 8, snímek 9 z testovací množiny.

Obrázek D.3: Výsledky testování: zleva doprava originál testovaného snímku, výsledek vlastního prototypu, výsledek prototypu Jana Šefčíka. shora dolů: snímek 10, snímek 11, snímek 12, snímek 13, snímek 15 z testovací množiny.