Geolokacija fotografija i videosnimki bazirana na prostornim i geometrijskim obilježjima

Carin Balić, Max

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 612

GEOLOKACIJA FOTOGRAFIJA I VIDEOSNIMKI BAZIRANA NA PROSTORNIM I GEOMETRIJSKIM OBILJEŽJIMA

Max Carin Balić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 612

GEOLOKACIJA FOTOGRAFIJA I VIDEOSNIMKI BAZIRANA NA PROSTORNIM I GEOMETRIJSKIM OBILJEŽJIMA

Max Carin Balić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 612

Pristupnik:	**Max Carin Balić (0036521748)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Ante Đerek

Zadatak: **Geolokacija fotografija i videosnimki bazirana na prostornim i geometrijskim obilježjima**

Opis zadatka:

Geolokacija fotografija i videosnimki je proces određivanja njihove geografske lokacije na temelju informacija koje se nalaze u samom sadržaju. U okviru diplomskog rada potrebno je istražiti postojeće metode geolokacije zadanog sadržaja usporedbom poznatih prostornih i geometrijskih obilježja tog sadržaja sa satelitskim slikama i geodetskim podacima. Na temelju provedenog istraživanja, potrebno je osmisliti i implementirati novu metodu geolokacije Ili implementirati neku od postojećih metoda. Ekstrakcija prostornih i geometrijskih značajki iz slike ili videozapisa nije glavni fokus rada. Umjesto toga, potrebno je razviti sustav koji omogućuje specificiranje značajki za određenu sliku ili videozapis. Radu je potrebno priložiti izvorni kod razvijenih i korištenih programa, citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Geolokacija medijskih sadržaja je relevantna tema s velikim brojem primjena. U nedavnom ratu u Ukrajini vidjeli smo vojnu važnost geolokacije slika i videa objavljenih na društvenim mrežama [2]. Uz vojnu važnost, geolokacija raznih medijskih sadržaja objavljenih u ratu je važan alat novinarima i istraživačima prilikom istrage potencijalnih ratnih zločina i kršenja ljudskih prava [1]. Ali i izvan rata, saznati točnu lokaciju fotografije ili videa vrlo je korisno za novinare, policiju, i druge istraživače.

Geolokacija slika i videa se često obavlja ručno – stručnjaci prvo pokušaju pronaći lokaciju baziranu na nekom očitom tragu. Ako to ne uspije, nacrtaju skicu lokacije i pokušaju ručno pronaći lokaciju na kartama ili satelitskim slikama [1][3]. Cilj ovog rada je istražiti mogućnost automatskog pronalaženja ovakvih skica u *OpenStreetMap* (OSM) podacima.

Razvili smo uređivač za izradu skica i algoritam koji pokušava pronaći skice u OSM podacima. Cilj nam je bio da algoritam bude otporan na nepreciznosti u skici i razlike između OSM baze podataka i stvarnog svijeta. Iako nismo postigli željenu otpornost na nepreciznost skice, pokazali smo da je algoritam efektivan za urbane slike iz ptičje perspektive, gdje je lako nacrtati preciznu skicu. Dodatno, testirali smo algoritam na stvarnim videozapisima nepoznate lokacije unutar Pojasa Gaze i pronašli lokacije dva videozapisa.

U prvom poglavlju razmatramo dosadašnje radove na temu geolokacije slika i videozapisa te iznosimo vlastiti pristup. U drugom poglavlju se upoznajemo s *OpenStreetMap* bazom podataka i definiramo kako ćemo iskoristi njene podatke. U trećem poglavlju opisujemo algoritam pretrage koji smo razvili. Postavljamo hipotezu na efektivnost algoritma, opisujemo razvoj algoritma i objašnjavamo kako radi. Uz to, opisujemo moguće poteškoće. U četvrtom poglavlju evaluiramo algoritam na različitim scenarijima. U petom poglavlju donosimo zaključak.

1. Dosadašnji radovi i novi pristup

1.1. Dosadašnji radovi

Geolokacija fotografija nije nova tema i postoji velik broj pristupa i radova koji se njome bave. Raniji pristupi se generalno baziraju na usporedbi slike s bazom geografski označenih slika (engl. *geo-tagged*) i dohvat najsličnijih slika [6]. Usporedba se mogla vršiti na razne načine, ali obično na bazi nepromjenjivih značajki. Ovaj pristup ne radi dobro za pretragu nad velikim područjem – veliko područje zahtijeva veliku bazu slika, što algoritam ne može podnijeti [4]. U [4] se predlaže poboljšanje koje povećava mogući broj slika, ali se i dalje ograničuje na područje grada. No možda veći problem svih algoritama koji uspoređuju sliku s bazom označenih slika je to što imaju ograničeni broj diskretnih lokacija koje mogu pronaći – samo one koje su u bazi. Možda mogu pronaći približnu lokaciju, bliska mjesta će često imati slične značajke, ali to onda nije egzaktna geolokacija.

Jedan od novijih pristupa je korištenje kartografskih podataka, a ne samo podataka iz označenih slika. Panphattarasap i Calway u [6] koriste panoramske fotografije i identificira prisutnost semantičkih obilježja u 4 smjera (naprijed, nazad, lijevo, desno). Ovi podaci se zatim uspoređuju s podacima u OSM (*OpenStreetMap*) bazi podataka. [7], umjesto ekstrakcije semantičkih značajki slike, trenira ugrađivanje i sijamsku neuronsku mrežu koja sama uči usporediti panoramske fotografije i OSM podatke. Oba rada postižu veliku točnost, ali su ograničeni time što ne mogu geolocirati jednu sliku, zahtijevaju seriju slika niz rutu duljine oko 200 metara. Dodatno, zahtijevaju panoramske slike. Zbog ovih ograničenja, generalno nisu primjenjivi na geolokaciju objavljenih slika i videa u realnim scenarijima – rijetko tko objavljuje rutu panoramskih slika.

Alternativni pristup problemu je tretirati ga kao klasifikacijski problem. [7] trenira duboku neuronsku mrežu koristeći milijune označenih slika. Zemlja je podijeljena u kvadratne sektore, svaki predstavlja jednu klasu. Dodatno, koristeći LSTM (*long short-term memory*) moguće je iskoristiti vremensku koheziju slika u albumima i postići 50% bolje performanse. Ovaj pristup ima prednosti – primjenjiv je na razne vrste slika, dok su prijašnji primjenjivi većinom na vanjske slike s puno informacija. Glavna mana je to što ne pronalazi točnu lokaciju, već samo približnu.

1.2. Novi pristup

Pristup koji ćemo istražiti u ovom radu ne rješava isti problem kao navedeni radovi, iako ima istu motivaciju. Ulaz algoritma nije slika – već skica lokacije iz ptičje perspektive.

Inicijalna ideja je bila napraviti rekonstrukciju scene iz slike i time automatski napraviti skicu, ali se pokazalo da postojeće metode za rekonstrukciju scene nisu dovoljno precizne i obično zahtijevaju više slika iz različitih kutova. Motivacija nam je bila moći pronaći slike i videozapise iz ratnih zona, pa ovo nije bilo prihvatljivo. Dodatna motivacija za ovaj pristup je što se nadovezuje na pristup koji koriste stvarni stručnjaci za geolokaciju.

Prednost pristupa je što se ovakav algoritam u teoriji može koristiti za razne primjene: slike, videozapisi, serije slika, panoramske slike – dok god je moguće nacrtati skicu. Još jedna prednost je što algoritam traži egzaktnu lokaciju i pretražuje OSM podatke, a ne bazu označenih slika.

Velika mana ovog pristupa je to što ostaje nedefinirano kako dobiti skice, iako predlažemo stručne skice iz radova [1] i [3] kao jednu moguću primjenu. Naravno, u mnogim slučajevima će biti jako teško ili nemoguće napraviti dobru skicu. Zato ne želimo testirati samo efektivnost algoritma koji traži skicu, nego i koliko je moguće napraviti dovoljno preciznu i informativnu skicu u različitim scenarijima.

Testirat ćemo koliko efektivno sami možemo izraditi skice bazirano na različitim izvorima informacija: OSM karta, satelitske snimke, *Google Street View* i stvarni videozapisi nepoznate lokacije. Skice ćemo izraditi u uređivaču koji smo razvili i zatim ih pokušati geolocirati pomoću razvijenog algoritma. Olakšavajući faktor u testiranju je to što čak za videozapise nepoznate lokacije možemo lako ručno provjeriti jesmo li pronašli točnu lokaciju.

Ideja nije konkurirati radovima koji potpuno automatski geolociraju sliku, već vidjeti može li se napraviti algoritam koji ubrzava ručno pretraživanje velikih područja i provjeriti koliko je ovaj pristup efektivan.

2. Podaci i postava

Prvo što moramo definirati je koje podatke ćemo koristiti za pretraživanje i kako ćemo ih koristiti. Slijedi pregled OpenStreetMap baze podataka i kako se ona inkorporira u rad.

2.1. OpenStreetMap podaci

OpeenStreetMap (OSM) je najveća besplatna i otvorena baza kartografskih podataka. Podaci su u XML formatu. Postoje tri vrste objekta:

- *Node* – točka u prostoru i mogući metapodaci o njoj.
- *Way* – slijed *node* objekata koji tvori oblik i metapodaci o njemu. Primjer *waya* koji opisuje D zgradu FER-a je vidljiv na slici Sl. 2.1. Ovaj primjer ima 14 točaka, tj. *node* objekata, ali smo ostavili samo jedan zbog kompaktnosti.
- *Relation* – rijetki tip objekta koji povezuje više *way* objekata u skup i daje metapodatke o skupu.

```
<way id="201559184" visible="true" version="14" changeset="112283849" (...)>
    <nd ref="321539323"/> (...)
    <tag k="addr:housenumber" v="39"/>
    <tag k="addr:street" v="Ulica grada Vukovara"/>
    <tag k="architect" v="Marijan Hržić"/>
    <tag k="building" v="university"/>
    <tag k="building:levels" v="4"/>
    <tag k="building:levels:underground" v="1"/>
    <tag k="building:material" v="concrete"/>
    <tag k="name" v="Zgrada D"/>
    <tag k="operator" v="Fakultet elektrotehnike i računarstva"/>
</way>
```

Sl. 2.1 Primjer *way* objekta

Node objekti, tj. elementi, su jedini koji imaju geografsku poziciju. Pozicija *way* i *relation* objekata je definirana *node* objektima koje sadrže. Metapodaci su zapisani u ključ – vrijednost formatu unutar *tag* elemenata. Metapodaci mogu biti sadržani u sve tri vrste objekta. *Node* može biti samostalni objekt – primjerice drvo, klupa, koš za smeće i slično. U tom slučaju, razni metapodaci će biti sadržani u node objektu. Kada je *node* dio *way* objekta, često neće imati nikakve metapodatke, metapodatci će biti u *way* objektu. Postoje iznimke – primjerice pješački prijelazi, koji mogu biti označeni kao metapodatak unutar *node* objekta koji je dio ceste, tj. *way* objekta koji je cesta. Isto tako, *way* objekti koji su dio *relation* objekta koji predstavlja zgradu sastavljenu od više poligona često nemaju metapodatke, metapodatci su u *relation* objektu.

2.2. Definicija vlastitih objekata

Iz svih ovih metapodataka moramo definirati što ćemo koristiti za algoritam. Ideja je koristiti samo nepromjenjive, lako prepoznatljive objekte za usporedbu. Usporedba će biti bazirana na oblicima objekata i odnosima između njihovih pozicija u svijetu. Moguće bi bilo koristiti više apstraktne podatke, kao što su kućni brojevi, vrste zgrada (stambeni objekt, dućan, itd.), broj katova, itd. Međutim, odlučili smo ne koristiti ove podatke jer su oni iskoristivi samo u određenim situacijama i često nisu prisutni u OSM podacima.

Primjerice, ako zadamo da zgrada koja je na slici ima četiri kata, i tražimo samo takve zgrade, moguće je da zgrada postoji u OSM bazi podataka, ali metapodatak o broju katova ne postoji u bazi. Općenito, ne možemo se osloniti na to da značajke sa slike postoje u OSM bazi podataka i zato ima smisla koristiti samo najosnovnije podatke: poziciju, oblik i jako osnovne metapodatke – je li nešto zgrada, cesta, drvo… Ovi podaci bi trebali postojati za sve objekte u OSM bazi. Naravno, neće svi objekti iz stvarnog svijeta biti u OSM bazi. Dodatno, korištenje ovakvih osnovnih podataka ostavlja prostora za to da se algoritam proširi podacima iz satelitskih snimki za slabo naseljena područja gdje je OSM baza rijetka.

Objekte iz OSM baze preslikavamo u 9 vlastitih vrsta objekata. Svaka vrsta je određena postojanjem određenih ključ – vrijednost parova unutar OSM objekta.

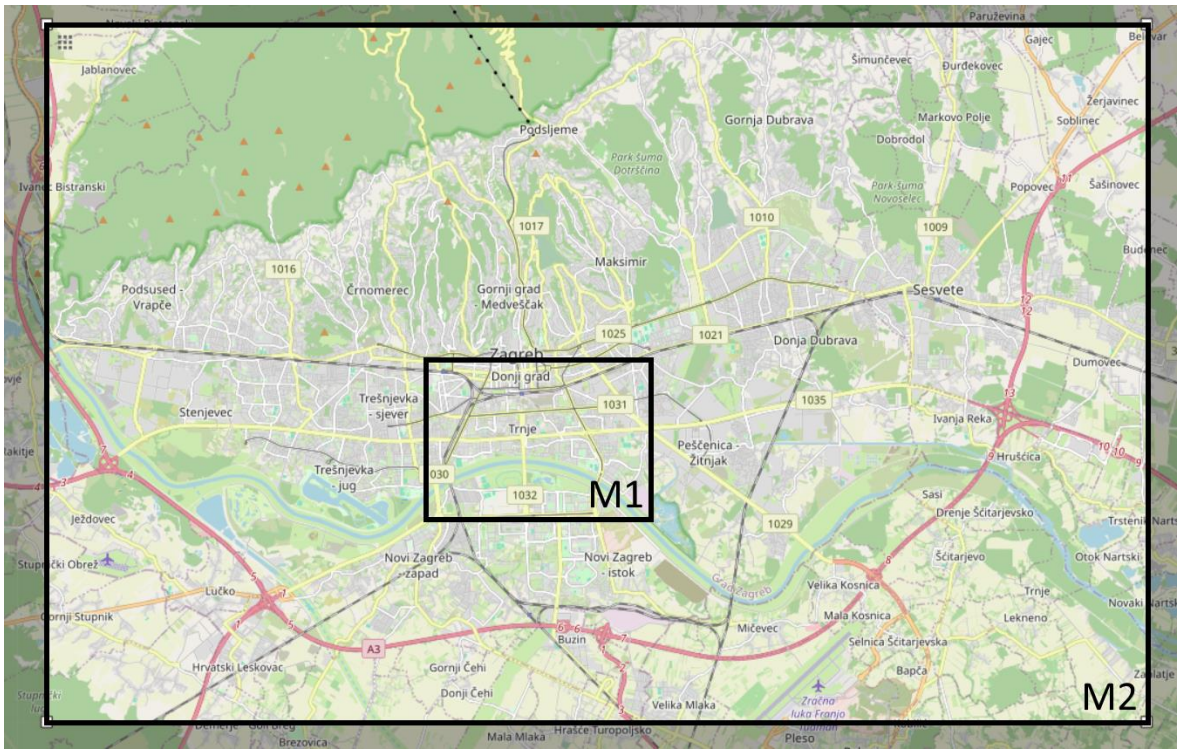
- Točka:
    1. Pješački prijelaz
    2. Drvo
    3. Klupa
    4. Ulična svjetiljka
- Slijed točaka (poligon ili izlomljena linija):
    1. Zgrada
    2. Cesta
    3. Nogostup
    4. Trava
    5. Ograda

Ovi objekti su odabrani jer su najbrojniji u OSM podacima za Zagreb, koji će se najviše koristiti u testiranju. Svrha odabira po učestalosti je da maksimiziramo šansu da objekt sa slike postoji i u bazi podataka.

2.3. Pregled OSM baze za Zagreb

U razvoju i testiranju algoritma većinom koristimo područje Zagreba. Imenovat ćemo dva pravokutna područja u Zagrebu:

- **M1 –** središnji dio Zagreba.
    - Zemljopisna širina: od 45.7775 do 45.8065.
    - Zemljopisna dužina: od 15.9490 do 16.0219.
- **M2 –** većina Zagreba i dio okolice.
    - Zemljopisna širina: od 45.7321 do 45.8852.
    - Zemljopisna dužina: od 15.8296 do 16.1760.



Sl. 2.2 Područja M1 i M2

U tablici (Tablica 2.1) vidimo koliko kojih objekata postoji na odabranim područjima. Jedna bitna stvar za zapaziti je kako u mapi koja pokriva središnji dio Zagreba ima relativno puno više manje bitnih objekata: drva, klupa, svjetiljki. To nam pokazuje da su je OSM karta puno detaljnija u popularnim područjima, tamo ima više kartografa i interesa za popisivanje objekata. Vidimo da kada uključimo širi dio Zagreba, skoro pola objekata postanu zgrade.

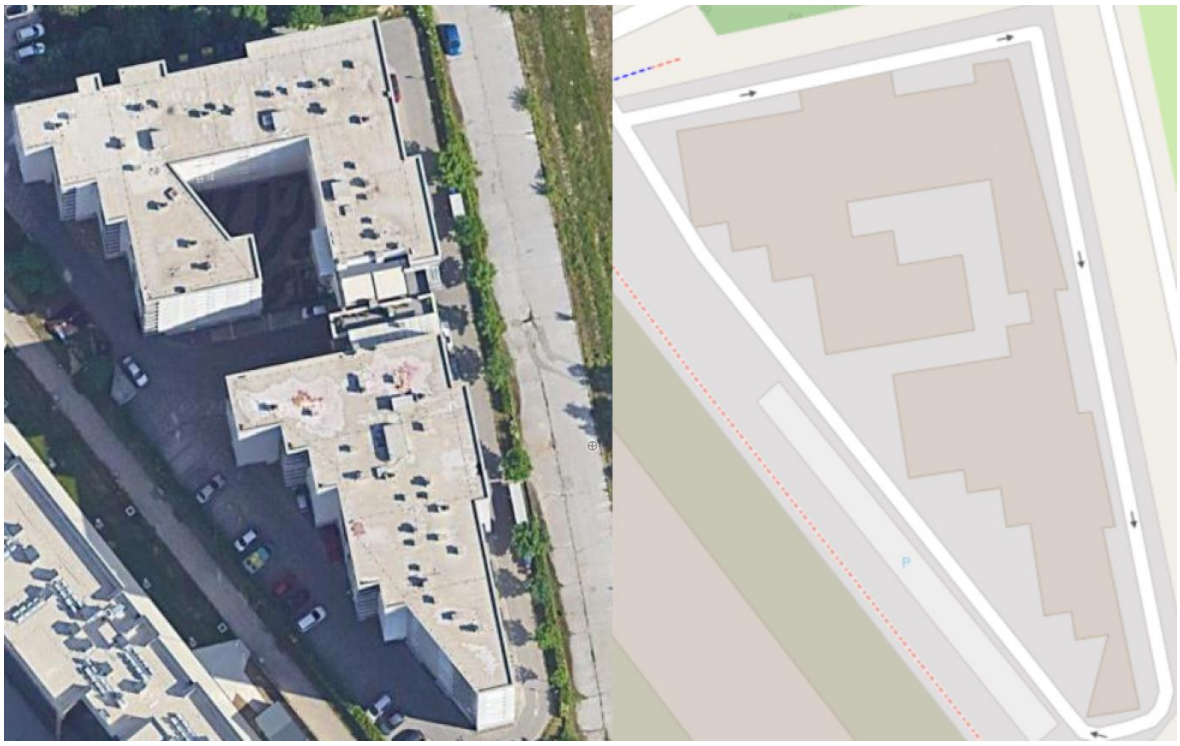
Tablica 2.1 Broj različitih objekata za područja M1 i M2

Vrsta objekta	M1		M2	
	Broj	**Udio**	**Broj**	**Udio**
Zgrada	9522	25%	118901	48%
Cesta	5158	13%	33563	14%
Pješački prijelaz	1376	4%	6914	3%
Drvo	9254	24%	35365	14%
Nogostup	6851	17%	31350	12%
Klupa	821	2%	3204	1%
Trava	2319	6%	9114	4%
Ulična svjetiljka	2629	7%	5205	2%
Ograda	872	2%	4081	2%

To nam govori da će nam zgrade biti najveći oslonac u pretrazi – najveća je šansa da zgrada sa slike postoji u bazi, pogotovo ako pokušamo obaviti pretragu na više ruralnim područjima. Drveće će se pokazati kao vrlo nepouzdani objekt, jako mali udio drveća je označen na karti.

Bitno je i istaknuti da točkasti objekti nose puno manje informacija od objekata koji imaju više bridova. Prosječni broj bridova za objekte u M1 je približno 7.481, a u M2 6.127. Taj podatak nam ukazuje da detaljnost zgrada pada što smo dalje od centra Zagreba, ali ne drastično.

Uz nepostojanje određenih objekata, problem će nam također predstavljati neprecizno označavanje zgrada. Jedan primjer nepreciznosti pronađen tijekom testiranja algoritma je na slici Sl. 2.3. Iako na prvi pogled oblik izgleda slično, velik broj zidova je potpuno krivih proporcija ili pod krivi kutom, određeni zidovi ne postoje, na nekim mjestima su izmišljeni nepostojeći zidovi.



Sl. 2.3 Primjer neprecizno označene zgrade

2.4. Projekcija

Pozicije točaka u OSM bazi zadane su geografskim koordinatama – geografska širina i duljina. S druge strane, algoritam zahtijeva dvodimenzionalne kartezijske koordinate. Geografska širina i duljina se ne mogu koristiti direktno kao kartezijske koordinate. Oko ekvatora pomak po širini i pomak po visini predstavljaju podjednak pomak u svijetu. Ali, što se više približavamo polovima, meridijani su bliži jedni drugima. Zato kako se približavamo polovima, pomak po geografskoj duljini predstavlja sve manji pomak u svijetu.

Kada bi koristili geografske koordinate direktno kao kartezijske, objekti bi nam se izobličili. Zato trebamo napraviti projekciju svijeta na dvodimenzionalnu plohu. Postoje razne projekcije. Bitno nam je svojstvo konformnosti – kutevi između linija prije i poslije projekcije ostaju isti [9].

Jednostavna projekcija koja postiže ovo svojstvo je sferična ekvatorijalna Mercatorova projekcija. Ona pretpostavlja da je Zemlja sfera te postavlja cilindar na ekvator i vrši projekciju točaka na cilindar. Razvučeni cilindar zatim tvori dvodimenzionalnu plohu. Formula za projekciju je sljedeća [9]:

$$\lambda - \text{geografska dužina}$$

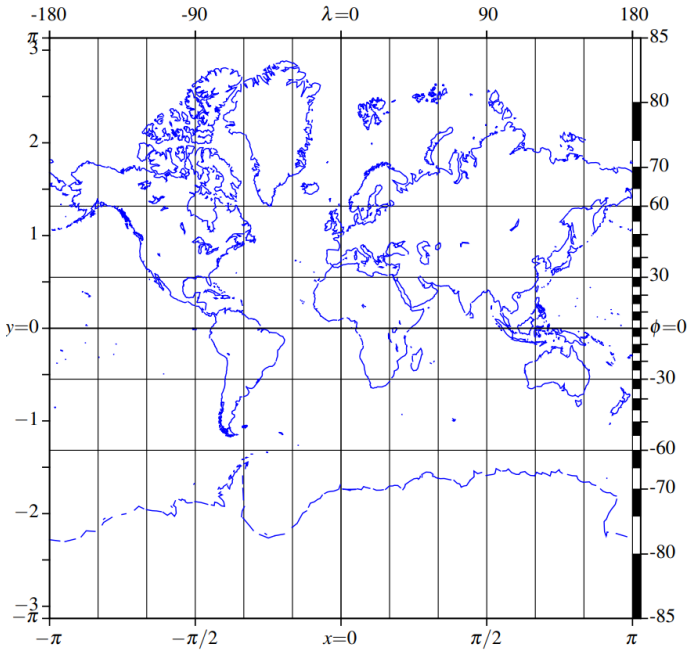
$$\varphi - \text{geografska širina}$$

$$R = 6378137\ m$$

$$x = R\,\lambda \tag{1}$$

$$y = R \ln\left[\tan\left(\frac{\varphi}{2} + \frac{\pi}{4}\right)\right] \tag{2}$$

Iako je projekcija konformna, ima jedan bitan artefakt – ako gledamo projekciju (Sl. 2.4) od ekvatora prema sjeveru (ili jugu), vidimo da je širina svugdje ista, dok se u stvarnosti zemlja sužava.



Sl. 2.4 Mercatorova projekcija [9]

Kako bi se suzbilo to „sužavanje“, oblici na projekciji se povećavaju u svim smjerovima kako bi popunili širinu. Pravokutnici na slici Sl. 2.4 u stvarnosti imaju istu površinu. Kao intuitivno objašnjenje oblika pravokutnika, možemo zamisliti da što smo bliže polovima, to horizontalne stranice pravokutnika postaju manje (jer se Zemlja sužava), dok vertikalne ostaju jednako duge. Zatim, kako bi širina karte bila ista svugdje, trebamo povećati horizontalne stranice tako da budu iste kao one na ekvatoru. Ali, kako se dimenzije ne bi izobličile, moramo jednako povećati i vertikalne stranice. Vertikalne stranice se nisu smanjile, pa kada ih povećamo postanu dulje od onih na ekvatoru, kao što se vidi na slici.

Rezultat ovoga je da se oblici šire u svim smjerovima kako se približavamo polovima. Ovo dovodi do izobličenja kada gledamo jako velike oblike – sjever Grenlanda je mnogo više

proširen nego jug. Kada gledamo male oblike, to proširenje je toliko maleno da oblici efektivno ostaju isti. Zato nam je ova projekcija dovoljno dobra, uz jednu opomenu.

Projekcija nam daje xy koordinate u metrima, i želimo iz njih direktno računati udaljenosti u metrima. Ali, kako se približavamo polovima, udaljenosti postaju preuveličane, stvarne udaljenosti su manje. Unutar algoritma ćemo htjeti imati određene pragove u metrima, pa je potrebno riješiti ovaj problem. Srećom, rješenje je jednostavno, udaljenosti se povećavaju s sljedećim faktorom, kojeg ćemo nazvati *MercatorScale* [9]:

$$\text{MercatorScale}(\varphi) = \frac{1}{\cos \varphi} \tag{3}$$

Kako bi dobili jednako ponašanje na svim geografskim širinama, jednostavno ćemo morati pomnožiti pragove, ili podijeliti udaljenosti s ovim faktorom. Ovo rješenje radi samo za male udaljenosti, jer pretpostavlja da su točke čija se udaljenost računa na istoj geografskoj širini, ali to je dovoljno dobro za naše potrebe.

3. Razvoj sustava

3.1. Hipoteza

Prva hipoteza je da će algoritam moći pronaći lokaciju ako na skici ima dovoljno objekata definiranih u poglavlju 2.2 i ako dovoljan dio tih objekata postoji u OSM bazi podataka, čak ako skica nije potpuno precizna i ako se objekti na skici samo djelomično poklapaju s objektima u OSM bazi.

Druga hipoteza je da će za slike i videozapise iz ptičje perspektive većinom biti moguće nacrtati dovoljno preciznu skicu, dok će za ostale slike i videozapise biti moguće nacrtati skicu, ali će biti znatno manje vjerojatno da je dovoljno precizna. Ovdje pretpostavljamo da skicu izrađujemo sami i nismo stručnjaci navedeni u [1][3].

3.2. Arhitektura sustava

Imat ćemo tri glavne komponente sustava. Prva komponenta je uređivač. Pomoću uređivača možemo stvoriti skice i spremiti ih u *SQLite* bazu podataka. Svaka skica ima svoj identifikator.

Druga komponenta je program koji uzima izvoz (engl. *export*) iz OpenStreetMap baze podataka, izvlači relevantne objekte i sprema ih u odvojenu *SQLite* bazu podataka, sličnog uređenja kao baza za skice. Pomoću ove komponente ćemo stvoriti nekoliko baza, jednu za svaku područje koje pretražujemo.

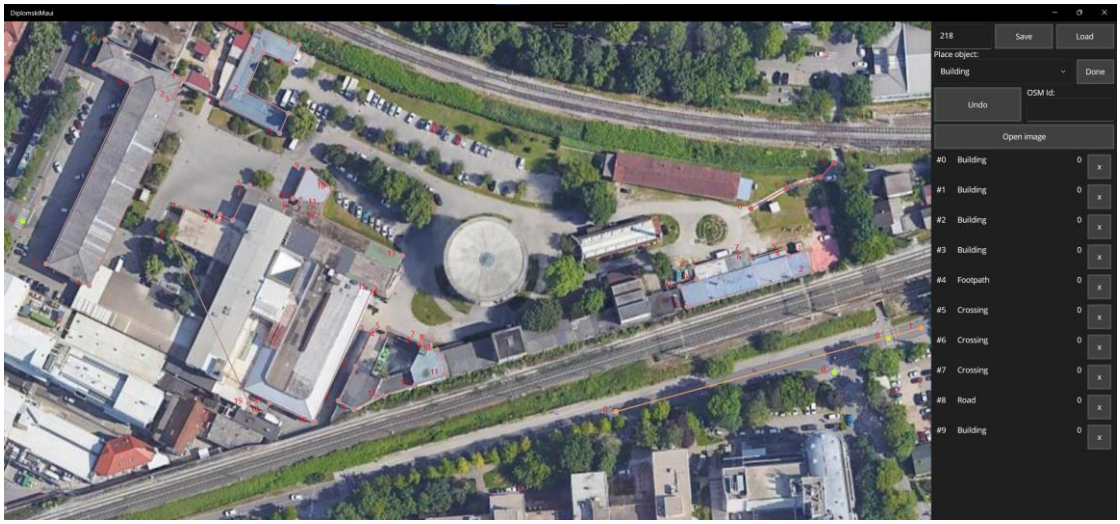
Treća komponenta je algoritam pretrage. On kao ulaz prima identifikator skice i bazu podataka područja u kojem tražimo skicu. Algoritam izvršava pretragu i stvara izvještaj u CSV (*comma-separated values*) formatu.

3.3. Uređivač za skice

Za početak je bilo potrebno definirati točno što će biti ulaz u algoritam. Zato smo prvo razvili uređivač za skice pomoću kojeg se stvaraju ulazni podaci za algoritam.

Uređivač dozvoljava postavljanje objekata definiranih u poglavlju 2.2 na 2D kanvas. Moguće je ucrtati točke, poligone i isprekidane linije. Jedna bitna značajka uređivača je

mogućnost postavljanja pozadinske slike, što se pokazalo vrlo korisnim za stvaranje preciznih skica u slučaju slika iz ptičje perspektive. Primjer skice i izgled uređivača je vidljiv na slici Sl. 3.1.



Sl. 3.1 Izgled uređivača

3.4. Osnovna ideja

Osnovna ideja algoritma je napraviti *fuzzy* pretragu objekata sa skice u OSM bazi podataka baziranu na obliku objekata i njihovim pozicijama. Pretraga treba biti neovisna o skaliranju i rotaciji. Algoritam ćemo podijeliti na dva dijela.

Prvi dio algoritma razmatra objekte od više točaka i traži slične objekte bazirano na obliku. Ova pretraga mora biti brza i otporna na razumne razlike između stvarnog oblika objekta i oblika objekta u OSM bazi podataka.

Drugi dio algoritma uzima potencijalne parove koje prvi dio algoritma pronađe i provjerava ih. Provjera je bazirana na poziciji objekata u odnosu na potencijalni par. Očekivano je da će prvi dio algoritma vratiti velik broj potencijalnih parova – zgrade su često vrlo slične. Zato drugi dio algoritma treba biti relativno jednostavan i brz.

3.5. Prvi dio algoritma

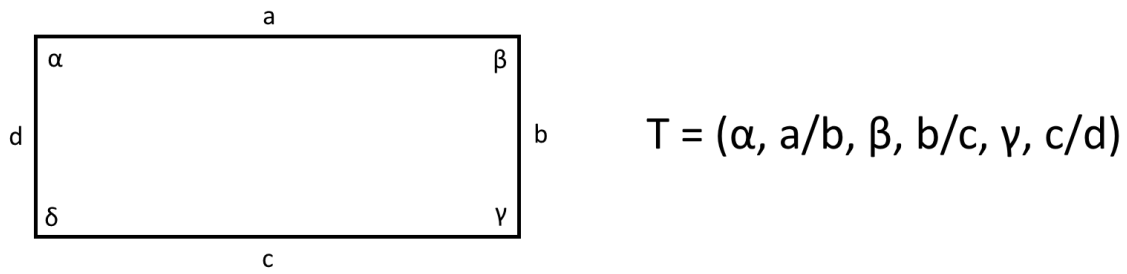
Kako bi mogli brzo pronalaziti objekte sličnog oblika, oblik objekta ćemo definirati kao *k*-dimenzionalnu točku. Točnije, uzeti ćemo skup od n bridova objekta i pretvoriti ih u *k*-dimenzionalnu točku tako da:

$$k = 2(n-1) \tag{4}$$

Za prvu dimenziju točke uzimamo kut između prva dva brida. Za drugu dimenziju uzimamo omjer dužine prvog i drugog brida. To radimo naizmjenično dok ne prođemo sve bridove. Sljedeći pseudokod opisuje ovaj proces:

```
for (i = 0; i < n - 1; i++)
{
   točka[i * 2] = kut_između(brid[i], brid[i + 1])
   točka[i * 2 + 1] = duljina(brid[i]) / duljina(brid[i])
}
```

Slika Sl. 3.2 prikazuje kako uz n = 4 stvaramo 6-dimenzionalnu točku počevši od brida a.



Sl. 3.2 Primjer stvaranja točke

Bitno je zapaziti da je redoslijed obilaska bridova bitan, i da za pravokutnik na slici Sl. 3.2 možemo stvoriti 4 različite točke (iako su neke iste zbog simetričnosti). Za svaki brid možemo stvoriti jednu točku.

Sve objekte iz OSM baze koji imaju barem n bridova pretvaramo u ovakve točke. U teoriji bi mogli objekt od m bridova pretvoriti u $\frac{m}{n}$ točaka. To bi funkcioniralo u slučaju da se skica savršeno poklapa s objektom u bazi. U stvarnosti, moramo napraviti jednu točku za svaki brid u bazi, tj. svaki brid koji ima još barem n – 1 bridova nakon sebe.

Za brzi dohvat ovih točaka koristit ćemo *k*-d stablo. Prosječna složenost dohvata u *k*-d stablu je $O(\log N)$, gdje je N ukupan broj bridova u bazi podataka.

Nakon što imamo stablo, bridove objekata skice na isti način pretvaramo u točke i tražimo ih u stablu. Razmatramo sve točke unutar hiperpravokutnika centriranog oko točke koju dobijemo iz skice. Za kutove dozvoljavamo apsolutno odstupanje za parametar ϵ, pretražujemo interval $< \alpha - \epsilon, \alpha + \epsilon >$. Za ϵ smo većinom uzimali vrijednosti od 10° do 15°, ovisno koliko pouzdana je skica. Za omjere dužina dozvoljavamo odstupanje za postotak d, pretražujemo interval $< \frac{omjer}{1+d},\ (1+d)\ \times\ omjer >$. Za x smo uzimali vrijednosti od 20% do 30%.

Parametar n ima velik utjecaj na ponašanje algoritma. Ako se pitamo kada će algoritam uspješno upariti točne objekte sa skice i iz baze, odgovor je da će uspjeti ako ta dva objekta imaju barem *n* dovoljno sličnih bridova zaredom. Što je *n* veći, to je manja vjerojatnost da ćemo uspješno upariti objekte. Dodatno, ako je *n* veći od broja bridova u objektu, algoritam mora preskočiti taj objekt. S druge strane, što je *n* manji, to ćemo dobiti više lažno pozitivnih objekata. U testiranju većinom koristimo *n* = 4. To je najveći *n* koji i dalje pokriva skoro sve objekte, kao što smo istaknuli u poglavlju 2.3. U nastavku ćemo promotriti koliko su koji oblici česti i kako to utječe na algoritam.

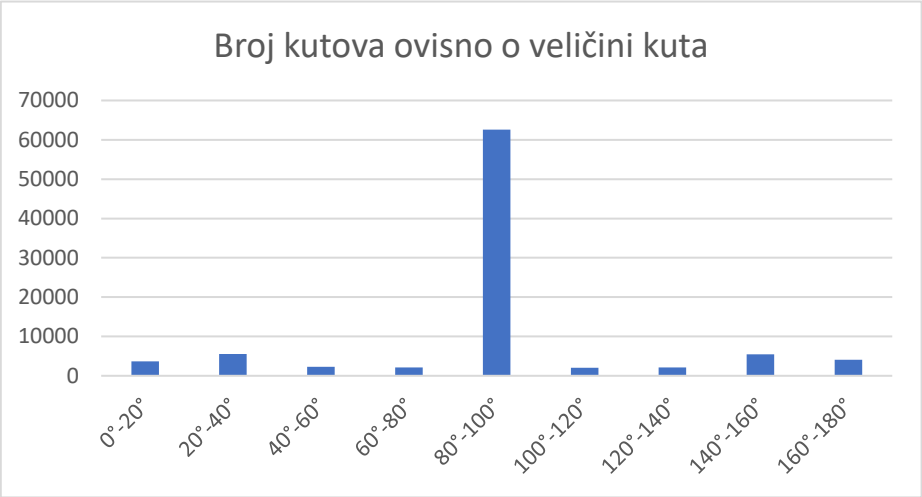
Tijekom razvoja algoritma i ranog testiranja se ispostavilo da ceste i nogostupi nisu pogodni za prvi dio algoritma, samo za drugi. Ceste imaju malen broj bridova, potrebna je slika jako velike površine da bi dobili 4 jasna brida ceste. Dodatno, ceste u OSM bazi podataka su često razlomljene na više objekata koje nije trivijalno sastaviti u cjelinu koja ima veći broj bridova. Zato ćemo ceste i nogostupe koristiti samo u drugom dijelu algoritma.

3.5.1. Učestalost oblika

Jedan od velikih problema ovog algoritma je to što su određeni oblici ekstremno česti. Za početak, pogledajmo koliko je koji kut čest za područje M1 (spomenuto u poglavlju 2.3). Na grafu na slici Sl. 3.3 vidimo da je daleko najviše kutova u blizini 90°. To ima smisla, pravi kutovi su česti u svim vrstama objekta, pogotovo u zgradama, koje čine velik dio baze podataka. Ovo bi nam moglo stvarati probleme zbog premale raznolikosti u *k*-dimenzionalnim točkama koje smo definirali.

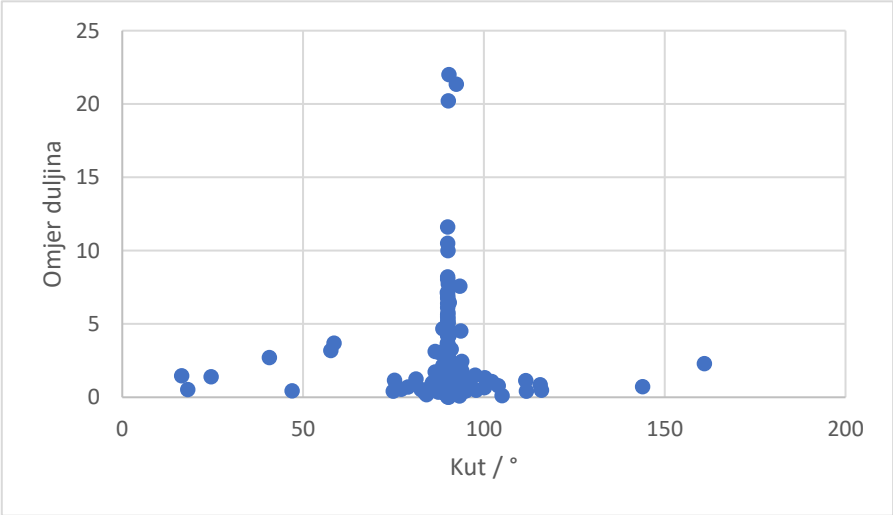
Također ćemo ovdje istaknuti da u stvarnosti ima i puno malih kutova, ali njih uklanjamo radi robusnosti. U OSM bazi često nailazimo na slučajeve gdje se brid nastavlja ravno, ali je podijeljen na dva brida koji su pod malim kutom, ili čak pod kutom od 0°. U ovom slučaju

bi imali problem, jer ako se na skici to nacrta kao jedan brid, ne bi došlo do uparivanja. Zato uklanjamo sve kutove manje 15°.



Sl. 3.3 Graf raspodjele kutova po veličini

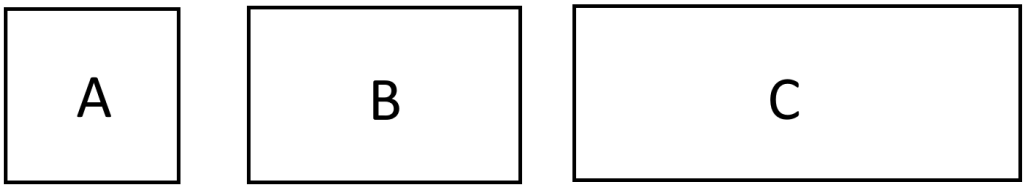
Sljedeće ćemo pogledati graf (Sl. 3.4) koji ilustrira raspodjelu nasumično odabranih točki za $n = 2$. Za ovaj graf smo uzeli samo zgrade, jer su nam one najbitnije. Vidimo da je većina točaka u području točke (90°, 1) – većina susjednih zidova je podjednake duljine i dotiče se pod kutom od 90° stupnjeva. Ovo nam pokazuje važnost toga da parametar n ne bude premalen.



Sl. 3.4 Raspršeni graf 500 nasumičnih točaka za $n = 2$

Ako pretražimo točku (90°, 1) nad zgradama u području M1 uz $\epsilon = 15°, d = 0.3$, dobiti ćemo poklapanje 5512 od 9522 (57.88%) zgrada. U području M2 dobijemo 84383 od 118901 (70.97%) zgrada.

Sljedeće pitanje je koliki je ovo problem za naš odabrani $n = 4$. Jasno je da će najproblematičnije biti zgrade pravokutnog oblika. Zato ćemo prijašnji eksperiment ponoviti na 3 oblika prikazana na slici Sl. 3.5. Koristit ćemo područje M1.



Sl. 3.5 Skice pravokutnih zgrada

Dobijemo sljedeće rezultate:

Skica	Broj poklapanja	Udio poklapanja
A	2143	22.51%
B	2904	30.50%
C	1244	13.06%

Čini se da pravokutne zgrade čine veliki udio svih zgrada. Dodatno, koristili smo prilično velike parametre za dozvoljeno odstupanje (ϵ, d). Ovo će definitivno biti problem algoritmu što se tiče brzine. Problem nije skroz rješiv, ali ćemo komentirati moguće načine ublažavanja problema u nastavku, kad ćemo opisivati drugi dio algoritma.

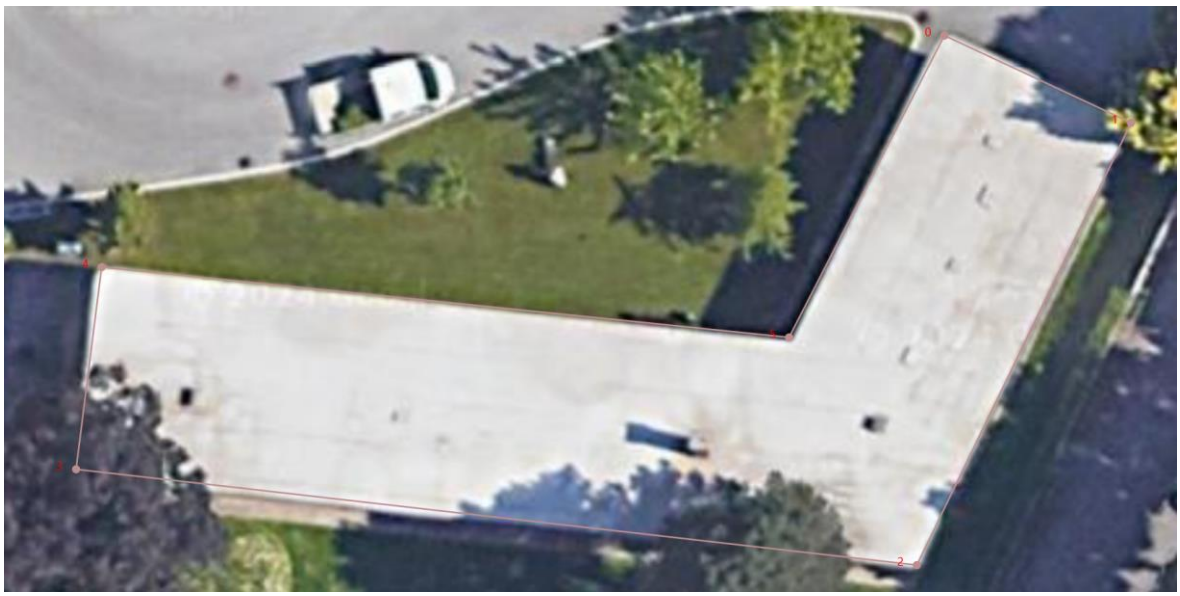
3.5.2. Alternativna primjena prvog dijela algoritma

Zaključili smo da ćemo najrobusniju pretragu postići ako podijelimo objekte na relativno male segmente, od 4 brida, ali da ćemo također imati problema s velikim brojem rezultata pretrage. Sada ćemo razmotriti suprotni smjer razmišljanja – što ako povećamo *n* tako da zahvaća veći dio objekta, ili cijeli objekt?

Kako *n* raste, broj potencijalnih kandidata drastično pada – koje su šanse da objekt od 20 bridova ima istog takvog negdje na karti? Vrlo male ili nikakve, osim ako se radi o nekom generičnom obliku. Nažalost, vjerojatnost da se kompliciran objekt sa skice potpuno poklapa s objektom u OSM bazi je također vrlo malena.

Unatoč tome, postoje određeni slučajevi gdje objekt nema jako velik broj bridova i postoji dobra šansa da se skica i baza poklapaju, ali je i dalje jedinstvenog oblika. Pogledajmo primjer na slici Sl. 3.6. Uz dozvoljeno odstupanje $\epsilon = 15°, d = 0.3$, u cijelom području M2

(većina Zagreba), ne postoji ni jedan objekt dovoljno sličan objektu sa slike da bi ga pretraga pronašla. Dodatno, objekt je precizno označen u bazi i algoritam ga uspješno pronalazi.



Sl. 3.6 Primjer jedinstvene zgrade

Jedinstvene zgrade su zapravo vrlo česte, samo nije često da se dovoljno poklapa stvarni oblik zgrade i oblik u bazi. Često je i samo dio zgrade s većim brojem bridova, ili s čudnom kombinacijom kutova jedinstven.

Iako je samo u određenim situacijama efektivno, korištenje samo prvog dijela algoritma na jedinstvenim ili rijetkim oblicima je vrlo brzo – pretraga je logaritamske složenosti (ovisno o ukupnom broju bridova). Takvu pretragu bi vrlo brzo mogli vršiti za cijeli svijet, samo bi morali implementirati k-d stablo na disku, zbog velikih memorijskih zahtjeva, i izrada stabla bi dugo trajala.

3.6. Drugi dio algoritma

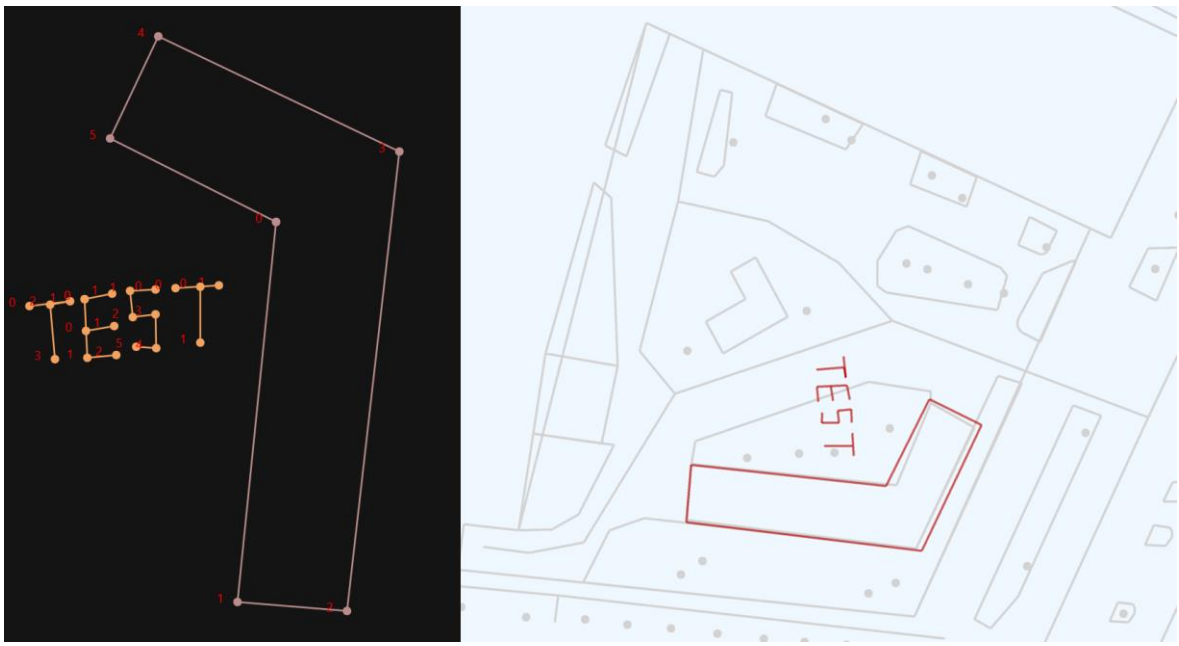
Izlaz prvog dijela algoritma su dva skupa od n bridova koji se poklapaju – jedan sa skice, jedan s karte, tj. OSM baze podataka. Zadatak drugog dijela algoritma je provjeriti koliko dobro se ostatak objekata sa skice poklapa s okolnim objektima karte.

Prvi korak je napraviti transformaciju objekata sa skice na kartu. Transformaciju radimo na sljedeći način:

1. Definirajmo skupove T_{karta} i T_{skica} kao skup od $n + 1$ točaka karte odnosno skice koje su dio bridova koji se poklapaju.

2. Odredimo ishodište skice C_{karta} i C_{skica} kao centroid od T_{karta} odnosno T_{skica}. Izračunamo skupove vektora V_{karta} i V_{skica} kao vektore točaka u T_{karta} odnosno T_{skica} uzevši C_{karta} odnosno C_{skica} kao ishodište.
3. Odredimo faktor skaliranja $s = \frac{ukupna\ duljina\ bridova\ karte}{ukupna\ duljina\ bridova\ skice}$.
4. Odredimo rotaciju transformacije r tako da uzmemo medijan svih kutova između vektora V_{karta} i V_{skica}.
   4.1. Preciznije, uzimamo kutove između vektora na istim pozicijama u skupovima – kut između prvog vektora u V_{karta} i prvog vektora u V_{skica}. Time dobijemo skup od n kutova i uzmemo njegov medijan. Kodomena računanja kuta nam je $(0°, 360°]$, pa nije ispravno uzeti prosjek, jer je moguće da su neki kutovi 1°, a neki 360°. To su skoro isti kutovi, ali ako uzmemo prosjek dobijemo potpuno krivi kut. Stoga kad uzimamo medijan skupa koji ima parni broj članova ne smijemo uzeti prosjek dva središnja elementa, nego jedan od tih elemenata.
5. Preslikavamo sve objekte sa skice na kartu. Za svaku točku svakog objekta skice uzmemo vektor od C_{skica} do te točke, rotiramo ga za r i skaliramo ga za s.

U svrhu prikaza rezultata i otklanjanja grešaka, implementirali smo iscrtavanje karte i transformirane skice. Možemo pogledati primjer na slici Sl. 3.7.



Sl. 3.7 Skica u uređivaču (lijevo) i skica transformirana na mapu (desno)

Sljedeći korak je pronaći sve objekte s karte koji su blizu našoj skici. To radimo pomoću odvojenog k-d stabla, gdje k = 2, koje sadrži sve objekte karte.

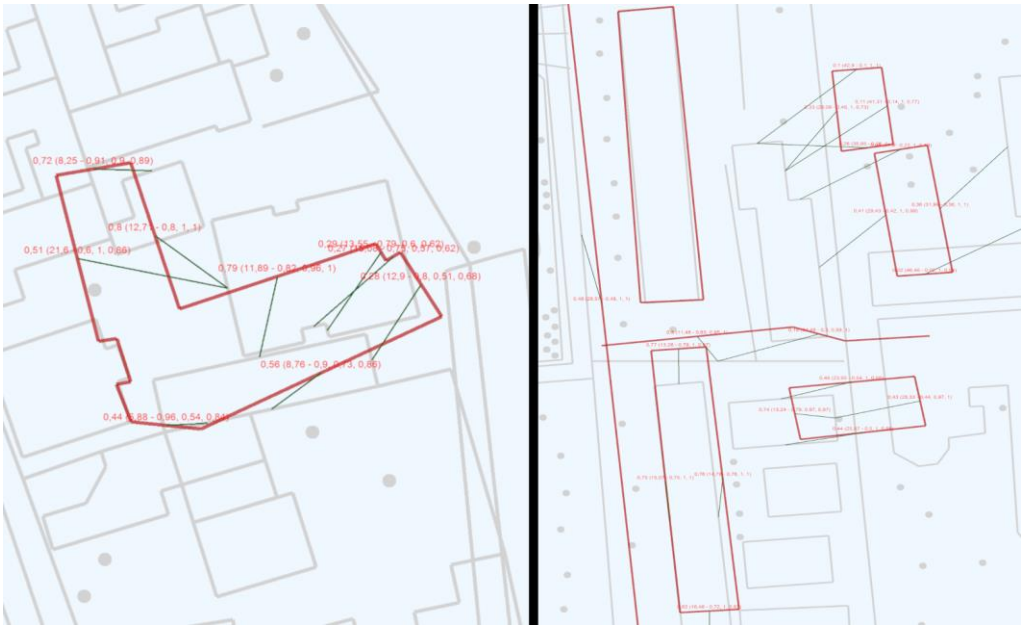
Sada za svaki objekt sa skice trebamo vidjeti postoji li dovoljno slični objekt na karti. Za točkaste objekte je ovo vrlo jednostavno, samo trebamo vidjeti imali li na karti dovoljno blizu točkasti objekt iste vrste.

Za objekte od više točaka je nešto složenije. Prvo što moramo imati na umu je da nije efektivno uspoređivati objekt s objektom, jer kada crtamo skicu, ne možemo znati gdje su granice između objekata. Ako se dvije zgrade dodiruju, ponekad će biti jedan objekt, ponekad dva. Ako popločeni put prelazi preko travnjaka, ponekad će travnjak biti jedan objekt, a ponekad će biti presječen na dva.

Zato ćemo uspoređivati bridove s bridovima, a ne objekte s objektima. Bridove ćemo uspoređivati po duljini, kutu i udaljenosti središta brida. Za ceste i pješačke staze ćemo imati posebni slučaj – na slikama se često ne vidi cijela cesta, pa će na skici biti samo dio ceste. Zato za ceste i pješačke staze ne gledamo udaljenost od središta brida skice do središta brida karte, nego udaljenost od središta brida skice do najbliže točke brida karte.

Inicijalna ideja je bila da za svaku točku i brid imamo ocjenu poklapanja koja eksponencijalno pada s udaljenosti, razlikom u kutu i duljini. Ovime bi imali jako visoku ocjenu ako je skica precizna, ali bi i dalje imali nekakvu ocjenu ako nije vrlo precizna.

Nažalost, ovo se pokazalo neuspješnim. Najčešći objekti su nam zgrade – cijela karta je prepuna bridova od zgrada. Ispostavilo se da vrlo često dobijemo do 70% poklapanja na lažno pozitivnim slučajevima, dok za nepreciznu skicu dobijemo negdje 30-50% poklapanja. Jedan ovakav slučaj je prikazan na slici Sl. 3.8.



Sl. 3.8 Kriva lokacija s visokom ocjenom (63%) i točna lokacija s niskom ocjenom (34%)

Uz neku kompliciraniju usporedbu, koja bi vjerojatno uključivala strojno učenje, bi bilo moguće napraviti da se neprecizne skice prepoznaju kao točna lokacija. Međutim, nemamo primjere za učenje i cilj nam je da ova usporedba bude vrlo brza. Zato smo odlučili odustati od jako neprecizno nacrtanih objekata i fokusirati se na one koji jesu precizni.

Umjesto kontinuirane ocjene poklapanja, koristimo binarnu ocjenu – ili se poklapa ili ne. Kriterij za poklapanje je vrlo strog, te je stroži za češće objekte. Za bridove zgrada je kriterij najstroži, generalno koristimo dozvoljeno odstupanje od 3 do 5 metara. Za neke rjeđe objekte dozvoljavamo odstupanje od 10 do 15 metara. Sva dozvoljena odstupanja se množe s MercatorScale iz izraza (3). Za kutove između bridova je dozvoljeno odstupanje 7.5°, a za duljinu bridova 25%. Za ceste i pješačke staze se duljina ne ocjenjuje iz ranije navedenih razloga. Ne dajemo točne brojeve jer su ovo podesivi parametri algoritma.

Ovakav pristup se pokazao mnogo uspješnijim u testiranju od prijašnje opisanog. Oslanja se na to da strogi uvjet poklapanja stvara mali broj lažno pozitivnih poklapanja, pa je dovoljno dobro da samo dio objekata bude precizno nacrtan. Ipak, djelomično smo odustali od djela hipoteze koji tvrdi da je moguće prepoznati i nepreciznu skicu.

Izlaz algoritma su potencijalne lokacije poredane po broju poklapanja. Za najbolje lokacije iscrtavaju slike kakve smo vidjeli ranije (Sl. 3.8 i Sl. 3.7), obično prvih 25 ili 50 lokacija. Iz tih slika je lako vidjeti jesmo li pronašli točnu lokaciju te koja je po redu među kandidatima.

3.6.1. Složenost drugog dijela algoritma

Za točkaste objekte, samo moramo provjeriti postoji li drugi točkasti objekt iste vrste unutar dozvoljenog odstupanja. Složenost provjere je $O(m \log N)$, gdje je N ukupan broj točkastih elemenata, a m broj točkastih objekata na skici.

Za objekte s bridovima uzimamo sve obližnje bridove karte i poredamo ih po kutu. Zatim radimo binarnu pretragu kako bi pronašli sve bridove unutar dozvoljenog odstupanja kuta i provjerimo jesu li duljina i udaljenost unutar dozvoljenog odstupanja. Ukupna složenost je $O(k \times l)$, gdje je k broj bridova skice, a l broj obližnjih bridove karte. Varijabla l je obično manja od 1000, ali ovisi o veličini skice i gustoći objekata karte na pronađenoj lokaciji. Iako ne možemo reći da složenost logaritamski ovisi l, konstanta je mala jer binarnom pretragom po kutu eliminiramo većinu bridova. Dodatno, provjeravaju se samo bridovi iste vrste objekta, što dodatno smanjuje konstantu.

3.6.2. Broj provjera

Kao što smo spomenuli u poglavlju 3.5.1, neki oblici su jako česti i zahtijevaju velik broj pokretanja drugog dijela algoritma. U teoretskom najgorem slučaju, broj provjera može biti $O(Bb)$, gdje je B broj bridova karte, a b broj bridova skice. U praksi, pretraživanje najčešćih oblika rezultira brojem provjera manjim od $0.1B$.

Prvi pristup smanjivanju broja provjera je provjeravanje samo lokacija gdje imamo veći broj poklapanja u blizini. Ovo se nije pokazalo učinkovitim jer se česti oblici obično pojavljuju zajedno, dok stvarna lokacija ne mora nužno imati veći broj poklapanja u blizini. Pokazalo se da ovakav pristup povećava brzinu, ali znatno smanjuje vjerojatnost da pronađemo stvarnu lokaciju.

Sljedeći pristup je stati s provjeravanjem poklapanja ako pronađemo dovoljno dobru lokaciju. Provjeravanje bi vršili u redoslijedu prema rijetkosti oblika, tako da stanemo prije nego što dođemo do čestih oblika. Glavni problem ovog pristupa je što je teško postaviti prag koji definira dovoljno dobar slučaj. Ako je prag previsok, nećemo stati rano i nećemo dobiti na brzini. Ako je prenizak, moguće je da ćemo stati na lažno pozitivnoj lokaciji. Ipak, ovaj pristup ima potencijala i isprobat ćemo ga prilikom evaluacije algoritma.

Jedna mala preinaka može eliminirati velik broj slučajeva. Kada imamo objekt od četiri brida iste ili bliske duljine, on će se 16 puta upariti sa svim drugim objektima koji imaju iste takve bridove. Ovo se događa zbog toga što imamo četiri pretrage, jednu za svaki brid objekta skice. Svaka pretraga pronađe četiri rezultata, jedan za svaki brid objekta karte. U poglavlju 3.5 smo opisali zašto vršimo jednu pretragu za svaki brid – radi povećanja robusnosti u slučajevima gdje nisu jasne granice između objekata. Međutim, za objekte od 4 brida nije nužno raditi pretragu za svaki brid jer nema mogućih problema s granicama objekata. Zato za njih možemo vršiti samo jednu pretragu.

Još jedan pristup je izbaciti objekte od 4 brida iz pretrage ili postaviti parametar n, tj. broj pretraživanih bridova u prvom dijelu algoritma, na 5. Dodatno, u uređivač smo dodali opciju za označiti objekt tako da se ne koristi u prvom dijelu algoritma, nego samo u drugom. S ovakvim pristupom prvi dio algoritma često neće pronaći ništa, ali će broj provjera biti drastično manji.

4. Testiranje i rezultati

Za evaluaciju algoritma imat ćemo 4 odvojena scenarija koji testiraju različita svojstva algoritma:

1. Precrtavanje OSM Karte – testira ispravnost algoritma.
2. Precrtavanje satelitskih snimki – testira performanse algoritma uz preciznu skicu.
3. Crtanje skice iz *Google Street Viewa* – testira koliko je teško napraviti skicu bez ptičje perspektive i koliko dobro algoritam radi na manje preciznim skicama.
4. Crtanje skice na bazi ratnih videa iz Pojasa Gaze – testira primjenjivost cijelog sustava.

Testiranje neće biti savršeno. Nećemo imati velike količine primjera za testiranje jer svaki mora biti ručno izrađen. Dodatno, nemoguće je potpuno objektivno napraviti skicu s time da ih ručno izrađujemo. Ipak, za svaki scenarij ćemo imati pravila pomoću kojih nastojimo povećati objektivnost evaluacije.

Imat ćemo sljedeće stupce u tablicama rezultata:

- **REZ** – najbolji rang točne lokacije prema drugom dijelu algoritma. Ako je **REZ** 1, pronašli smo točnu lokaciju. Ako je -1, točna lokacija nije pronađena u prvom dijelu algoritma.
- **T1** – vrijeme izvođenja prvog dijela algoritma.
- **T2** – vrijeme izvođenja drugog dijela algoritma.
- **POKL** – broj poklapanja u prvom dijelu algoritma.
- **PROV** – broj poklapanja iz prvog dijela algoritma koje smo morali provjeriti u drugom dijelu algoritma. Kasnije ćemo uvesti uvjet ranog zaustavljanja provjere**.**
- **N** – broj točaka i bridova na skici koje provjeravamo u drugom dijelu algoritma. Ovo gledamo kao slučajeve za svrhu stupaca **TP** i **FP**.
- **TP** – broj točaka i bridova koji se poklapaju za najbolju točnu lokaciju. Ovo možemo gledati kao točno pozitivne slučajeve.
- **FP** – broj točaka i bridova koji se poklapaju za najbolju krivu lokaciju. Ovo možemo gledati kao lažno pozitivne slučajeve.
- **TPR & FPR** – udio točno pozitivnih, odnosno lažno pozitivnih slučajeva.

Mjerenja vršimo na procesoru s prosječnom frekvencijom od 3.6 GHz. Algoritam je implementiran u jeziku C#. Algoritam se izvodi na jednoj dretvi. Ako se ne spominje drugačije, za parametre koristimo: $\epsilon = 10°, d = 0.25, n = 4$.

4.1. OSM Karta

U ovom scenariju ćemo ručno precrtavati objekte sa slike OSM karte. Time imamo najlakši scenarij za algoritam – ne postoje (vidljive) razlike između slike i OSM baze, te je slika iz ptičje perspektive, što čini crtanje skice laganim. Prva svrha ovog scenarija je provjeriti da algoritam radi ispravno u skoro savršenim uvjetima. Dodatna svrha je analizirati lažno pozitivne lokacije i vrijeme izvođenja.

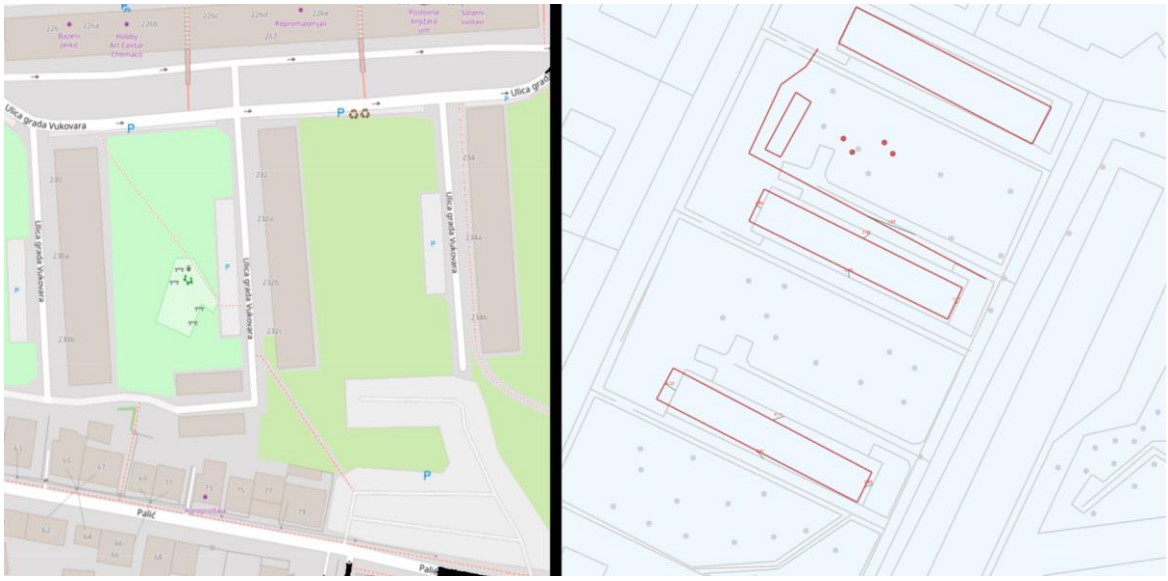
Imat ćemo 10 slučajeva, lokacije će biti nasumično odabrane u području M1. Prilikom crtanja skice nećemo crtati sve objekte, pokušat ćemo imati različite brojeve objekata i raznolik odabir objekata.

#	REZ	T1	T2	PK	PR	N	TP	FP	TPR	FPR
1	1	0.02	0.30	407	407	39	39	6	1.00	0.15
2	1	0.02	0.88	956	956	36	31	6	0.86	0.17
3	1	0.06	0.50	1521	1521	43	43	7	1.00	0.16
4	1	0.02	0.17	411	411	39	36	9	0.92	0.23
5	1	0.02	0.00	83	83	16	16	4	1.00	0.25
6	1	0.00	1.02	5929	5929	17	14	5	0.82	0.29
7	1	0.05	1.45	3443	3443	27	24	9	0.89	0.33
8	1	0.00	0.16	610	610	19	18	6	0.95	0.32
9	1	0.00	0.19	1060	1060	21	19	7	0.90	0.33
10	1	0.00	0.77	2159	2159	20	19	9	0.95	0.45

Kao što smo očekivali, u svim testovima smo uspjeli pronaći točnu lokaciju. U prva 4 testa ima više objekata na skici, dok sljedećih 6 imaju nešto manje. Rezultati impliciraju da **FP** nije ovisan o **N,** imamo podjednaku količinu lažno pozitivnih poklapanja za manje i za veće **N**-ove. Iz toga slijedi da je **FPR** negativno ovisan o **N**. Ovo nije velik broj testova, pa nije moguće zaključiti da to generalno vrijedi, ali možemo pretpostaviti da je bolje ucrtati što više objekata u skicu.

Pogledajmo što se dogodilo u slučaju 10, gdje je **FPR** bio najveći (45%). Slučaj vidimo na slici Sl. 4.1. Vidimo da je stvarna lokacija vrlo slična lažno pozitivnoj lokaciji, ima 3 zgrade istog oblika i međusobne udaljenosti. Međutim, ostali objekti, osim te 3 zgrade i jedne ceste, nisu isti. Ovaj slučaj nam pokazuje kako bi dodavanje još objekata na skicu, tj. povećanje **N**, smanjilo **FPR**.

Ovo ima smisla – što je više objekata, to je teže pronaći lažno pozitivnu lokaciju koja ih slučajno sve sadrži. Slično je povećanju parametra *n* u prvom dijelu algoritma. S druge strane, ako saturiramo skicu objektima, sigurno ćemo dobiti nešto više lažno pozitivnih slučajeva. Međutim, čini se da je to povećanje sublinearno.



Sl. 4.1 Stvarna lokacija i lažno pozitivna lokacija iz slučaja 10

Prvi dio algoritma je bio brz za sve slučajeve. Drugi dio algoritma je trebao više od sekunde za neke slučajeve, gdje je trebalo provjeriti više tisuća potencijalnih lokacija.

FPR i **TPR** su nam bitni za ubrzanje algoritma. Spomenuli smo u poglavlju 3.6.2 da može ranije stati s drugim dijelom algoritma, ako pronađemo dovoljno dobru lokaciju. Prema ovim podacima, mogli bi za uvjet zaustavljanja postaviti poklapanje od 50%. Uz taj uvjet dobijemo sljedeće rezultate:

Tablica 4.1 Vrijeme izvođenja prije i poslije dodavanja ranog zaustavljanja

#	1	2	3	4	5	6	7	8	9	10
POKL	407	956	1521	411	83	5929	3443	610	1060	2159
PROV	6	26	911	16	1	1120	502	83	292	103
T2	0.30	0.88	0.50	0.17	0.00	1.02	1.45	0.16	0.19	0.77
T2 uz zaustavljanje	0.02	0.03	0.59	0.02	0.00	0.17	0.17	0.02	0.08	0.05

Uvelike smo smanjili količinu lokacija koje je trebalo provjeriti. U petom slučaju je točna lokacija odmah pronađena. S druge strane, u drugom slučaju nije eliminirano ni pola lokacija. Svi slučajevi su i dalje uspješno pronašli lokaciju, ali u stvarnom scenariju nećemo znati točno koji prag možemo postaviti tako da izbjegnemo sve lažno pozitivne lokacije.

4.2. Satelitske snimke

Za sljedeći scenarij ćemo crtati objekte sa satelitskih snimki nasumično odabranih lokacija unutar područja M1. Slike će biti približno dimenzija 300 m × 200 m. Imat ćemo 20 slučajeva. Nećemo gledati OSM kartu – morat ćemo pogađati jesu li dvije zgrade koje se dodiruju jedan ili dva objekta u OSM bazi. Nećemo crtati sve objekte, crtat ćemo samo one u koje imamo najviše povjerenja. Izbjegavat ćemo zgrade oblika pravokutnika gdje je moguće.

Jedna nasumična lokacija je završila u Savi i nije sadržavala ni jedan objekt, pa smo ju isključili iz rezultata.

#	REZ	T1	T2	POKL	PROV	N	TP	FP	TPR	FPR
1	1	0.03	1.34	3706	3706	49	46	12	0.94	0.24
2	1	0.00	0.13	296	296	38	22	6	0.58	0.16
3	1	0.03	0.69	739	739	40	25	12	0.63	0.30
4	1	0.00	0.30	743	743	24	20	9	0.83	0.38
5	1	0.03	0.53	535	535	55	29	9	0.53	0.16
6	1	0.03	2.36	4402	4402	41	24	7	0.59	0.17
7	1	0.02	0.64	2086	2086	36	13	6	0.36	0.17
8	1	0.02	0.88	5414	5414	30	12	6	0.40	0.20
9	1	0.03	0.30	317	317	57	10	8	0.18	0.14
10	1	0.03	0.09	465	465	31	22	7	0.71	0.23
11	1	0.02	0.19	323	323	38	28	7	0.74	0.18
12	1	0.00	0.17	286	286	34	15	9	0.44	0.26
13	1	0.02	0.30	521	521	48	15	9	0.31	0.19
14	1	0.00	0.73	6195	6195	47	15	7	0.32	0.15
15	1	0.03	0.67	801	801	61	29	13	0.48	0.21
16	1	0.05	0.94	795	795	70	7	7	0.10	0.10
17	1	0.02	0.41	967	967	67	28	12	0.42	0.18
18	2	0.03	1.52	2774	2774	50	12	15	0.24	0.30
19	1	0.02	0.30	374	374	59	28	8	0.47	0.14
20	1	0.03	0.34	453	453	42	13	6	0.31	0.14

Rezultati su iznenađujuće pozitivni. U svim slučajevima osim 18. slučaja je najbolje ocijenjena točna lokacija. U 18. slučaju je točna lokacija bila druga po redu, i dalje možemo smatrati da smo pronašli točnu lokaciju. Vrijeme izvođenja drugog dijela algoritma je bilo većinom ispod jedne sekunde, uz par iznimki u slučajevima 1, 6 i 18.

Pogledajmo što se dogodilo u slučaju 18 (Sl. 4.2). Vidimo da ima puno bliskih zgrada s nejasnim oblicima. Zato smo ucrtali najjasnije oblike i izbjegavali zgrade oblika pravokutnika. Nažalost, objekti u OSM bazi su bili manje detaljni nego na skici i nismo imali

dovoljno poklapanja. Kad dodamo još par objekata, točna lokacija postane najbolje ocijenjena.

Sl. 4.2 Slučaj 18

Slučaj 16 također nije vrlo uspješan, dobivamo poklapanje **TPR** od samo 10%. Već smo se susreli s razlogom zašto ovaj primjer nije uspio, ovo je primjer sa slike Sl. 2.3. Većina bridova skice je jako neprecizno označena u OSM bazi. Unatoč tome, lažno pozitivne lokacije također nisu vrlo uspješne, pa je točna lokacija i dalje pronađena.

Rano zaustavljanje algoritma uz prag od 60% bi pomoglo u samo 5 od 20 slučajeva. Ni jedan slučaj ne bi postao lažno pozitivan zbog ranog zaustavljanja, ali ne bi ni bilo previše korisno. Najveći **FPR** je bio 38% u 4. primjeru, pa bi uz niži prag imali više uspjeha – za prag od 40% bi rano stali u 13 od 20 slučajeva. Sve u svemu, rano zaustavljanje se čini previše riskantnim i nepotrebnim, osim ako pretražujemo jako veliko područje i sigurni smo u preciznost skice.

Sljedeće ćemo razmotriti performanse algoritma na području M2, koje ima 10 puta više objekata od područje M1. Rezultati su sljedeći:

Tablica 4.2 Rezultati za područje M2

#	REZ	T1	T2	POKL	PROV	N	TP	FP	TPR	FPR
1	1	0.13	8.64	39863	39863	49	46	12	0.94	0.24
2	1	0.05	0.80	1758	1758	38	22	8	0.58	0.21
3	1	0.09	4.64	5233	5233	40	25	12	0.63	0.30
4	1	0.02	0.94	3278	3278	24	20	13	0.83	0.54
5	1	0.06	1.53	2659	2659	55	30	11	0.55	0.20
6	1	0.11	12.75	56926	56926	41	24	7	0.59	0.17
7	1	0.06	3.86	17112	17112	36	13	6	0.36	0.17
8	1	0.14	7.19	57883	57883	30	12	6	0.40	0.20
9	1	0.08	2.47	2029	2029	57	10	9	0.18	0.16
10	1	0.13	0.58	2457	2457	31	20	8	0.65	0.26
11	1	0.05	0.91	1826	1826	38	28	10	0.74	0.26
12	1	0.05	1.22	1819	1819	34	15	9	0.44	0.26
13	1	0.08	2.36	3518	3518	48	16	9	0.33	0.19
14	1	0.22	11.63	131215	131215	47	15	8	0.32	0.17
15	1	0.17	3.75	5169	5169	61	29	13	0.48	0.21
16	4	0.14	5.58	5440	5440	70	7	9	0.10	0.13
17	1	0.09	2.20	6395	6395	67	28	12	0.42	0.18
18	2	0.20	13.78	32085	32085	50	12	15	0.24	0.30
19	1	0.06	1.61	2321	2321	59	28	10	0.47	0.17
20	1	0.06	1.20	2242	2242	42	13	6	0.31	0.14

Rezultati su i dalje dobri. Vidimo da se jedino rezultat 16. primjera promijenio, sada je točna lokacija četvrta po redu. To je očekivano, kao što smo već objasnili, 16. primjer ima jako nisko poklapanje zbog nepreciznosti u OSM bazi podataka. Čak iako je četvrta najbolja lokacija, i dalje možemo smatrati da je točna lokacija pronađena.

Uspoređujući vremena izvođenja, vidimo da je **T1** nešto porastao. Iako je prvi dio algoritma logaritamske složenosti, kopiranje rezultata u listu je linearne složenosti, ovisno o broju rezultata. Zato vidimo malo veća vremena kod primjera s velikim **POKL**. Broj objekata je porastao 10 puta, dok je **T2** porastao za prosječno 6.51 puta. Porast je približno linearan, što je prihvatljivo.

Sljedeće ćemo testirati što se dogodi ako promijenimo parametar *n* s 4 na 5. Rezultati su sljedeći:

#	REZ	T1	T2	POKL	PROV	N	TP	FP	TPR	FPR
1	1	0.03	0.13	94	94	48	45	7	0.94	0.15
2	1	0.03	0.08	100	100	37	9	4	0.24	0.11
3	1	0.03	0.61	639	639	39	24	11	0.62	0.28
4	**-1**	0.00	0.00	0	0	0	0	0	0.00	0.00
5	1	0.02	0.16	348	348	54	30	11	0.56	0.20
6	1	0.02	0.23	218	218	40	29	5	0.73	0.13
7	1	0.00	0.06	75	75	35	11	5	0.31	0.14
8	1	0.02	0.03	110	110	29	14	4	0.48	0.14
9	1	0.02	0.28	277	277	56	11	7	0.20	0.13
10	1	0.05	0.03	274	274	30	17	5	0.57	0.17
11	1	0.02	0.08	251	251	37	26	7	0.70	0.19
12	1	0.03	0.16	284	284	33	14	7	0.42	0.21
13	1	0.03	0.14	357	357	47	15	6	0.32	0.13
14	1	0.03	0.02	35	35	46	15	5	0.33	0.11
15	1	0.05	0.67	924	924	60	27	10	0.45	0.17
16	10	0.05	0.58	669	669	69	4	7	0.06	0.10
17	1	0.06	0.31	944	944	66	29	9	0.44	0.14
18	21	0.08	1.84	3248	3248	49	6	10	0.12	0.20
19	1	0.03	0.08	118	118	58	29	6	0.50	0.10
20	1	0.03	0.16	171	171	41	6	5	0.15	0.12

Primjeri 16. i 18. su se pogoršali. Primjer 4 je potpuno ispao jer skica nije imala ni jedan objekt s više od 4 brida. S druge strane, vrijeme izvođenja drugog dijela algoritma je drastično palo.

4.3. Google Street View testiranje

Sljedeće ćemo testirati puno teži scenarij: crtanje skice koristeći *Google Street View*. U ovom scenariju testiramo drugu hipotezu – koliko je moguće napraviti dovoljno preciznu skicu za slike koje nisu iz ptičje perspektive.

Nasumično ćemo odabrati 10 lokacija. Imat ćemo nekoliko pravila koja simuliraju osobu koja stvarno pokušava pronaći lokaciju:

1. Nije dozvoljeno gledanje lokacije iz zraka.
2. Nećemo se pomaknuti više od 100 metara od nasumično odabrane lokacije.
3. Algoritam stvara slike najboljih kandidata (npr. Sl. 3.8), te je dozvoljeno na bazi ovoga provjeriti uspješnost pretrage i doraditi skicu, uz vremensko ograničenje od 20 minuta.
4. Ako pronađemo točnu lokaciju, ali nije prva po redu, nećemo je dorađivati. Dorađujemo samo ako lokacije nema u slikama najboljih kandidata.

Koristit ćemo popustljivije parametre: $\epsilon = 20°, d = 0.40$. Rezultati su sljedeći:

#	REZ	T1	T2	POKL	PROV	N	TP	FP	TPR	FPR
1	-1	0.03	1.27	10320	10320	23	0	8	0.00	0.35
2	13	0.02	0.13	429	429	24	6	9	0.25	0.38
3	1	0.00	0.09	514	514	25	17	7	0.68	0.28
4	-1	0.03	0.88	3937	3937	17	0	7	0.00	0.41
5	28	0.00	0.17	586	586	32	7	10	0.22	0.31
6	1	0.02	0.09	611	611	26	8	7	0.31	0.27
7	-1	0.02	0.00	20	20	14	0	2	0.00	0.14
8	2	0.02	0.11	663	663	11	8	9	0.73	0.82
9	36	0.02	2.09	17364	17364	14	6	9	0.43	0.64
10	2	0.00	0.06	235	235	8	3	4	0.38	0.50

Slučajeve 1, 6, 8 i 9 smatramo potpuno uspješnima. Slučajevi 2, 5 i 9 su djelomično uspješni, mogla bi se ručno pronaći lokacija pregledavanje slika koje algoritam stvara. Primjeri 1, 4 i 7 su potpuno neuspješni, lokacija nije pronađena. Sve u svemu, ovo je dobar rezultat. Jako je teško napraviti preciznu skicu kada nemamo ptičju perspektivu.

U neuspjelim slučajevima je većinom problem bio što nismo mogli pronaći niti jedan objekt za koji možemo precizno ucrtati barem 4 brida. Ili to, ili objekti nisu postojali u OSM bazi. U djelomično uspjelim slučajevima je većinom problem bio što nismo mogli ucrtati dovoljno velik broj objekata.

4.4. Stvarni slučajevi

Za kraj ćemo pokušati pronaći lokacije nekih stvarnih videozapisa. S time da smo spominjali ratno novinarstvo kao jednu od motivacija, isprobat ćemo algoritam na videozapisima iz Pojasa Gaze. Generalno ćemo se fokusirati na videozapise snimljene dronom, jer su iz ptičje perspektive, što će olakšati izradu skica.

Pojas Gaze nije prevelik, možemo ga ubaciti u algoritam bez problema. Međutim, imat ćemo nekoliko velikih problema:

1. Većina objekata su zgrade i ceste, svih ostalih objekata ima manje od 1500 (Tablica 4.3). Dodatno, zgrade su jako gusto zbijene i većinom u obliku pravokutnika. Zbog ovoga će biti jako teško razaznati točnu lokaciju među lažno pozitivnim.
2. Većina videozapisa nije usmjerena ravno dolje, već pod kutom, što otežava izradu skice.
3. Podaci u OSM bazi podataka nisu vrlo detaljni, pogotovo za područja na vanjskim dijelovima gradova.

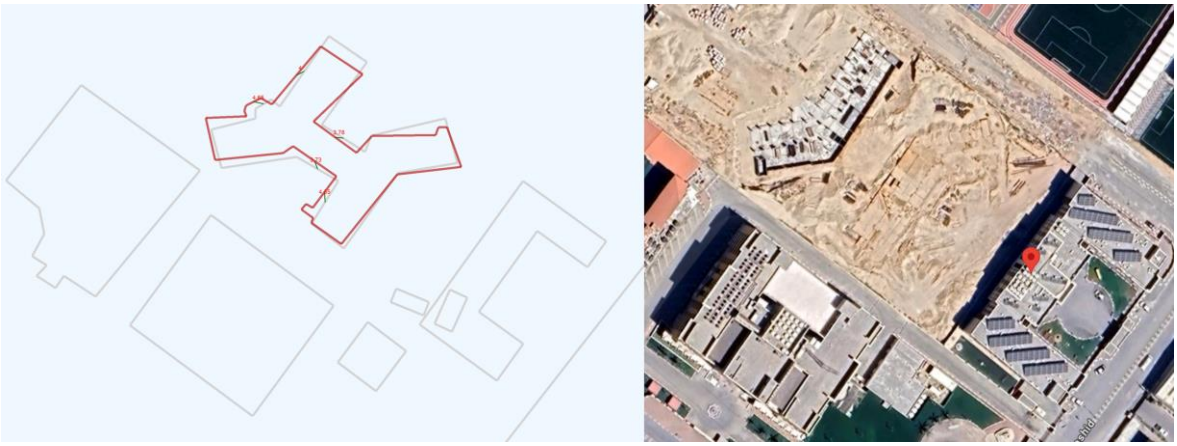
Tablica 4.3 Raspodjela objekata u Pojasu Gaze

Vrsta	Zgrada	Cesta	P. prijelaz	Drvo	Nogostup	Klupa	Trava	U. svjetiljka	Ograda
Broj	296224	23353	52	480	788	3	87	4	73
Udio	0.923	0.073	0.000	0.001	0.002	0.000	0.000	0.000	0.000

Pogledajmo par slika iz videozapisa koje ćemo pokušavati pronaći:

Sl. 4.3 Slike vojnih akcija u Pojasu Gaze

Od 15 različitih lokacija koje smo pokušali pronaći, uspjeli smo pronaći dvije. Prva pronađena lokacija je na slici Sl. 4.4, a slika po kojoj je pronađena je u donjem desnom kutu slike Sl. 4.3. Druga pronađena lokacija je na slici Sl. 4.5. Zanimljivo je da zgrada još nije izgrađena na satelitskoj snimci.



Sl. 4.4 Prva uspješno pronađena lokacija (31.5476676, 34.4582099)

Sl. 4.5 Druga uspješno pronađena lokacija (31.2666405, 34.2842932). Lijevo – slika iz videozapisa; sredina – skica; desno – lokacija na *Google Maps*

Pronalaženje prve lokacije je bilo lagano – imali smo vrlo dobru sliku iz ptičje perspektive. Pronalaženje druge lokacije je zahtijevalo nekoliko prepravljanja skice. Kada smo napokon pronašli lokaciju, provjerili smo zašto ranije iteracije skice nisu pronađene. Razlog je bio u tome što zgrade nisu bile dovoljno precizno ucrtane. Uvjet za poklapanje bridova zgrade u drugom dijelu algoritma je strog jer su zgrade vrlo česte i dovele bi do puno lažno pozitivnih poklapanja kada uvjet ne bi bio strog. U Gazi je to još veći problem, jer su zgrade puno gušće nego u Zagrebu.

Sve u svemu, pokazali smo da je moguće pronaći ovakve lokacije, iako nije velika vjerojatnost da ćemo moći napraviti dovoljno dobru skicu.

4.5. Moguća poboljšanja

Crtanje skice je najslabija karika ovog pristupa. Uređivač ima samo osnovne funkcije, bilo bi vrlo korisno imati određene alate za olakšano stvaranje skice. Primjerice, bilo bi vrlo korisno imati pomoćni algoritam koji objekte slikane pod kutom ispravlja u njihov originalni oblik, iz ptičje perspektive.

Još jedno potencijalno poboljšanje bi bilo zamijeniti ili nadograditi OSM bazu podataka pomoću ekstrakcije objekata iz satelitskih snimki. Ova ekstrakcije bi ponekad bila točnija, ponekad manje točna od OSM podataka – vjerojatno bi prosječno bila manje točna. Međutim, bila bi konzistentnije netočna, i znali bi da za svaki objekt u svijetu postoji barem nešto u bazi podataka. Algoritam pretrage bi se morao promijeniti, ali bi imao više mogućnosti jer se može osloniti na postojanje objekata.

Zaključak

Rezultati djelomično potvrđuju prvu hipotezu. Prilikom testiranja algoritma na satelitskim snimkama smo vidjeli da je vrlo lako pronaći točnu lokaciju, unatoč znatnim razlikama između svijeta i OSM baze. Međutim, nismo postigli željenu otpornost na nepreciznosti u skici. Algoritam i dalje uspijeva ako je samo dio skice neprecizan, ali ne uspijeva ako je velika većina skice neprecizna, čak ako je ta nepreciznost relativno mala u usporedbi s nepreciznosti koja bi učinila da čovjek ne može prepoznati lokaciju.

Rezultati potvrđuju drugu hipotezu. Slike iz ptičje perspektive smo većinom uspješno pronašli, dok smo za ostale perspektive imali miješane rezultate.

Brzina algoritma je bila prihvatljiva – uspjeli smo obaviti pretragu na cijelom Zagrebu i kasnije na cijelom Pojasu Gaze unutar minute, čak za najsporije slučajeve opisane u poglavlju 3.5.1.

Sukladno motivaciji, isprobali smo algoritam na stvarnim ratnim videima iz Pojasa Gaze. Gaza se pokazala problematičnom za algoritam – OSM podaci nisu precizni i sadrže većinom zgrade, a te zgrade su jednolične i zbijene. Uz to, videozapisi su većinom bili niske rezolucije i pod lošim kutom. Uspjeli smo pronaći dvije od 15 lokacija na kojima smo isprobali algoritam. Ovo nije velik udio, ali je zadovoljavajuće s obzirom na okolnosti.

Literatura

[1] Venkatagiri S., Thebault-Spieker J., Kohler R., Purviance J., Sabbir Mansur R., I Luther K. *GroundTruth: Augmenting Expert Image Geolocation with Crowdsourcing and Shared Representations.* In Proceedings of the ACM on Human-Computer Interaction, New York, (2019), vol. 3, članak 107. Poveznica: https://doi.org/10.1145/3359209.

[2] Trench Selfies: Tracking A Russian Military Unit By Frontline Social Media Photos. Poveznica: https://www.rferl.org/a/trench-selfies-tracking-russia-military-frontline-social-media-/32462632.html, pristupljeno 16.6.2024.

[3] Kohler R., Purviance J., Luther K. *Supporting Image Geolocation with Diagramming and Crowdsourcing*. Proceedings of the Fifth Conference on Human Computation and Crowdsourcing, (2017). Poveznica: https://cdn.aaai.org/ojs/13296/13296-64-16813-1-2-20201228.pdf, pristupljeno 16.6.2024.

[4] Schindler G, Brown M., Szeliski R. *City-Scale Location Recognition*. 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, (2007). Poveznica: https://ieeexplore.ieee.org/abstract/document/4270175?section=abstract&signout=success.

[5] Mousavian A., Košecká J., *Semantic Image Based Geolocation Given a Map*. Poveznica: https://arxiv.org/abs/1609.00278

[6] Panphattarasap P., Calway A. *Automated map reading: Image based localisation in 2-d maps using binary semantic descriptors*. In Proc. IEEE/RSJ Int Conf on Intelligent Robots and Systems, (2018). Poveznica: https://arxiv.org/abs/1803.00788.

[7] Samano N, Zhou M., Calway A. *You Are Here*: *Geolocation by Embedding Maps and Images*. Poveznica: https://arxiv.org/abs/1911.08797.

[8] Weyand T., Kostrikov I., Philbin J. *PlaNet - Photo Geolocation with Convolutional Neural Networks*. Poveznica: https://arxiv.org/abs/1602.05314.

[9] Osborne P. *The Mercator Projections*. Edinburgh: 2013. Poveznica: https://web.archive.org/web/20130930144834/http://www.mercator99.webspace.virginmedia.com/mercator.pdf, pristupljeno 16.6.2024.

Sažetak

Geolokacija fotografija i videosnimki bazirana na prostornim i geometrijskim obilježjima

Sažetak

Geolokacija medijskih sadržaja objavljenih na društvenim mrežama i *chat* grupama se pokazala bitnom za vojne svrhe, novinarstvo i istraživanje ratnih zločina. Postoji velik broj radova na temu geolokacije fotografija, ali generalno imaju preduvjete zbog kojih nisu primjenjivi za ovu svrhu, ili ne pronalaze točnu, već približnu lokaciju. U praksi se geolokacija ovakvih medijskih sadržaja obavlja ručno – često pretraživanjem karti ili satelitskih snimki na bazi stručno konstruirane skice lokacije iz ptičje perspektive.

U radu ispitujemo mogućnost automatskog pretraživanja ovakvih skica na *OpenStreetMap* podacima i konstruiramo algoritam pretrage. Iako nismo postigli željenu otpornost na nepreciznosti skice, ovaj pristup se pokazao efektivnim za fotografije iz ptičje perspektive, iz kojih je lako napraviti preciznu skicu. Pristup smo isprobali na stvarnim videima iz Pojasa Gaze i uspjeli pronaći lokacije dva videozapisa.

…

Ključne riječi: geolokacija fotografija, algoritam pretraživanja

Summary

Geolocation of photos and videos based on geometrical and spatial features

Abstract

Geolocating media posted on social media networks and chat groups has proven important for military purposes, human rights and war crime investigation, and journalism in general. Existing methods usually require conditions which aren't met for social media posts. In practice, journalists manually locate images, often resorting to drawing up a sketch of the location and searching for it on maps or satellite images.

We investigate the possibility of automatically finding such sketches in OpenStreetMap data and construct an algorithm that performs this task. Although requiring a precise sketch to function, the algorithm is effective for bird's-eye view images, where it's easy to construct a precise sketch. We applied this method to drone videos in Gaza and geolocated two videos.

Keywords: image geolocation, search algorithm