Usklađivanje traga kretanja cestovnih vozila primjenom geoprostornih operatora u bazi podataka PostGIS

Krehula, Matej

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:787214

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 461

USKLAĐIVANJE TRAGA KRETANJA CESTOVNIH VOZILA PRIMJENOM GEOPROSTORNIH OPERATORA U BAZI PODATAKA POSTGIS

Matej Krehula

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 461

USKLAĐIVANJE TRAGA KRETANJA CESTOVNIH VOZILA PRIMJENOM GEOPROSTORNIH OPERATORA U BAZI PODATAKA POSTGIS

Matej Krehula

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 461

Pristupnik:	**Matej Krehula (0036514991)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Dejan Škvorc

Zadatak: **Usklađivanje traga kretanja cestovnih vozila primjenom geoprostornih operatora u bazi podataka PostGIS**

Opis zadatka:

Usklađivanje traga kretanja je postupak koji se provodi u geografskim informacijskim sustavima da bi se niz zemljopisnih koordinata koje opisuju trag kretanja pokretljivog objekta uskladio s logičkim modelom zemljopisne karte. Usklađivanje traga kretanja potrebno je provoditi zbog pogreške u određivanju položaja pokretljivog objekta koju unosi uređaj za određivanje položaja, primjerice GPS prijemnik. U ovom radu potrebno je proučiti i opisati postojeće algoritme za usklađivanje traga kretanja te ih međusobno usporediti s obzirom na složenost, točnost i brzinu rada. S druge strane, potrebno je proučiti geoprostornu bazu podataka PostGIS i geoprostorne operatore dostupne u toj bazi. Oblikovati postupak za usklađivanje traga kretanja primjenjujući dostupne geoprostorne operatore u bazi podataka PostGIS. Programski ostvariti sustav za usklađivanje traga kretanja cestovnih vozila primjenom geoprostornih operatora u bazi podataka PostGIS zasnovan na zemljopisnoj karti OpenStreetMap. Ulazni podaci su logički model prometnica preuzet iz zemljopisne karte OpenStreetMap i trag kretanja vozila u obliku niza zemljopisnih koordinata prikupljenih iz GPS prijemnika. Izlazni podaci su usklađeni trag kretanja vozila i oznaka cestovnog segmenta na koji je preslikan položaj vozila za svaku pojedinačnu točku iz traga kretanja. Pokusnim ispitivanjem odrediti točnost i brzinu rada ostvarenog sustava. Točnost usklađivanja traga kretanja provjeriti za različite konfiguracije prometnica, kao što su gusta gradska mreža, prorijeđena ruralna mreža te raskršća. Opisati arhitekturu i programsko ostvarenje sustava te rezultate ispitivanja.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

1. Uvod

Usklađivanje traga kretanja (engl. *mapmatching*) je postupak u kojem se niz zemljopisnih koordinata koje opisuju kretanja nekog objekta usklađuje s logičkim modelom zemljopisne karte. Taj postupak je nužan u širokom spektru primjena u suvremenom svijetu, od korištenja navigacijskih sustava do naručivanja hrane ili prijevoznih usluga. Ovaj rad će se fokusirati na primjenu geoprostornih operatora unutar sustava za upravljanje bazom podataka (engl. *database management system*) PostgreSQL s proširenjem PostGIS kako bi se poboljšala preciznost traga kretanja cestovnih vozila.

S obzirom na to da uređaji za određivanje položaja, kao što su GPS prijamnici, često imaju greške u lociranju. U kontekstu GPS prijemnika one se mogu klasificirati u četiri kategorije: satelitske, koje su uzrokovane orbitom, satom i antenom GPS satelita; propagacijske, koje nastaju zbog refrakcije, ionizacije i treperenja GPS signala dok putuju kroz atmosferu; prijamničke, koje uzrokuju šum, pristranost i drift GPS prijemnika i antena; te višeputne, koje su rezultat refleksije, difrakcije i raspršenja GPS signala od strane obližnjih objekata [1]. Na primjer, mobilni uređaji opremljeni GPS-om obično su točni unutar radijusa od 4,9 metara na otvorenom prostoru [2]. Međutim, njihova točnost se pogoršava u blizini zgrada, mostova i drveća. Zbog tih grešaka lokacija vozila može biti pomaknuta na netočnu poziciju.

Ovaj rad stoga ima cilj analizirati i razviti sustav za usklađivanje traga kretanja koristeći geoprostorne operatore korištenjem proširenja PostGIS. Sustav će se temeljiti na logičkom modelu prometnica preuzetom iz zemljopisne karte OpenStreetMap, a njegova funkcionalnost će biti testirana.

U drugom poglavlju opisani su postojeći algoritmi za usklađivanje traga kretanja vozila. Treće poglavlje posvećeno je geoprostornim bazama podataka i PostGIS-u, njihovim

karakteristikama i primjeni u kontekstu usklađivanja traga kretanja. Četvrto poglavlje opisuje arhitekturu sustava razvijenog za usklađivanje traga kretanja. U petom poglavlju opisana je implementacija sustava, uključujući ključne tehnologije i metodologije koje su korištene. Šesto poglavlje bavi se ispitivanjem sustava, pri čemu su predstavljeni rezultati i analiza dobivenih podataka. Zaključak je iznesen u sedmom poglavlju, gdje su sumirani glavni rezultati rada.

2. Algoritmi usklađivanja traga kretanja vozila

Praćenje kretanja vozila pomoću GPS podataka često pati od pogrešaka uzrokovanih raznim faktorima kao što su smetnje signala, atmosferski uvjeti i urbana mreža prometnica. Algoritmi za usklađivanje traga kretanja vozila (engl. *mapmatching algorithms*) pomažu u ispravljanju tih pogrešaka i omogućuju precizno praćenje vozila na cestovnoj mreži. Ovi algoritmi su ključna komponenta za poboljšanje performansi sustava koji podržavaju navigacijske funkcije.

Prema starijoj klasifikaciji [3] algoritmi za usklađivanje kretanja vozila se dijele u četiri kategorije:

- **Geometrijske metode**: Fokusiraju se samo na udaljenost između elemenata traga vozila kojega se želi uskladiti i cestovne mreže. Ove metode su jednostavne i brze za implementaciju, ali često pate od niske točnosti, posebno u urbanim sredinama gdje je mreža cesta kompleksna. Primjeri uključuju metode najbliže točke i najkraće udaljenosti.

- **Topološke metode**: Uzimaju u obzir povezanost i sličnost oblika. Ove metode ne samo da gledaju udaljenost između točaka, već i kako su ti segmenti povezani. Time omogućuju bolje rezultate u složenim mrežama. Primjeri uključuju algoritme bazirane na grafovima koji koriste informacije o cestovnoj mreži za poboljšanje točnosti.

- **Probabilističke metode**: Modeliraju nesigurnost traga vozila kojega se želi uskladiti, uključujući pogreške mjerenja i nepoznate putove, s ciljem pronalaska puta s najvećom vjerojatnošću generiranja zadanog traga. Ove metode koriste statističke modele, poput skrivenih Markovljevih modela (HMM), za predviđanje najvjerojat-

nije rute vozila. Prednosti uključuju veću točnost, ali su složenije i zahtijevaju više računalnih resursa.

- **Napredne metode**: Temelje se na naprednim modelima kao što su Kalmanov filter, partikularni filter i fuzzy logika. Ove metode kombiniraju više izvora podataka i koriste sofisticirane algoritme za postizanje visoke točnosti. Na primjer, Kalmanov filter koristi se za integraciju GPS podataka s podacima iz drugih senzora kako bi se smanjila pogreška u pozicioniranju.

Prema [4] uspostavljena je nova klasifikacija algoritama za usklađivanje traga kretanja vozila prema njihovom jezgrenom modelu usklađivanja. U algoritmu za usklađivanje traga vozila, model je opći radni okvir (engl. *framowork*) ili princip izvođenja algoritma korišten za proces usklađivanja traga vozila. Model se najčešće sastoji od seta računskih komponenti (engl. *computation components*), kao što je izračun udaljenosti, prijelaz i modeliranje korisničkog ponašanja, te tijek izvođenja koji ih povezuje. Te komponente su fiksirane i implementacije im variraju između različitih metoda usklađivanja traga. Postojeći modeli za usklađivanje traga mogu se klasifirati u četiri kategorije: model sličnosti (engl. *similarity model*), model prijelaza stanja (engl. *state-transition model*), model evolucije kandidata (engl. *candidate-evolving model*) i model bodovanja (engl. *scoring model*). Modeli bazirani na sčičnosti su najjedostavniji jer uziamju samo točke bez konteksta između njih, dok ostali modeli imaju komplekcije implementacije. U nastavku će svaki od navedenih modela biti detaljnije opisan.

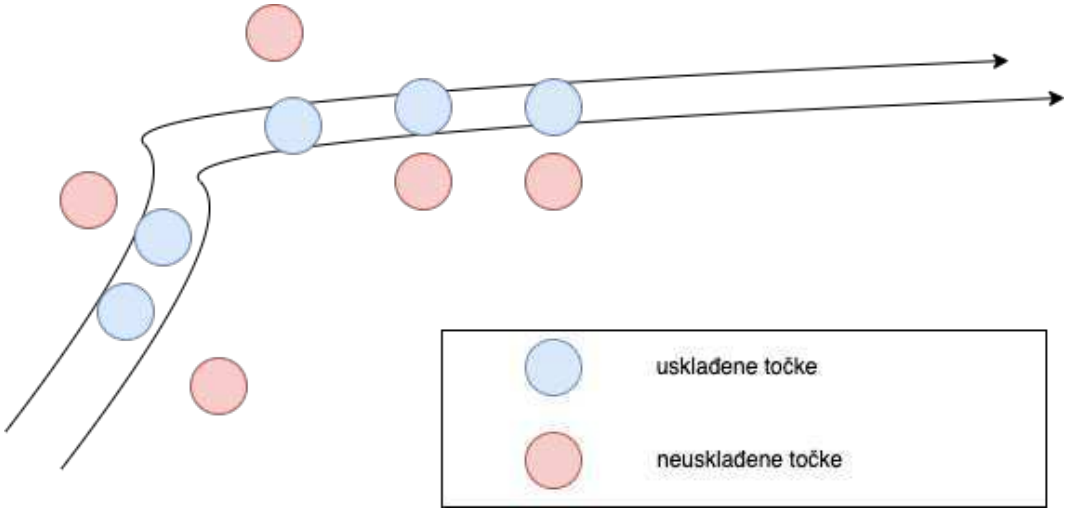
2.1. Model sličnosti

Model sličnosti odnosi se na opći pristup koji vraća vrhove/bridove koji su najbliži putanji vozila geometrijski i/ili topološki. Intuitivno, budući da kretanje vozila uvijek prati topologiju osnovne cestovne mreže, te vozilo nikada ne može preskočiti s jednog segmenta na drugi, putanja bi također trebala biti slična stvarnoj putanji na karti. Stoga je glavni fokus u ovoj kategoriji kako definirati mjeru blizine.

Modeli mogu biti bazirani na udaljenosti ili bazirani na uzorcima.

- **Bazirani na udaljenosti** - Najraniji algoritmi za podudaranje točke s krivuljom (engl.*point to curve*) i krivulje s krivuljom (engl.*curve to curve*) projiciraju točke ili

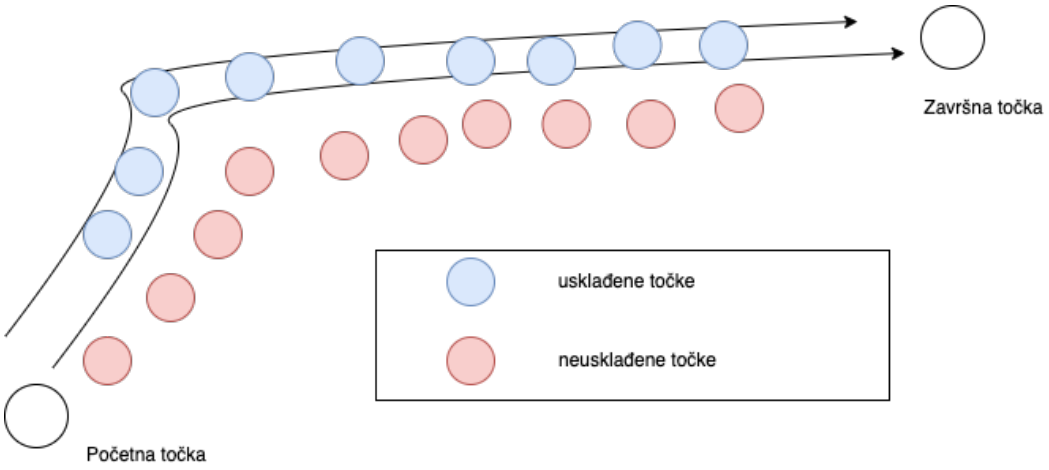
segmente trajektorije na geometrijski najbliže bridove. Fréchetova udaljenost je često korištena za mjerenje sličnosti jer uzima u obzir monotonost i kontinuitet krivulja, ali je osjetljiva na ekstremne vrijednosti.



Slika 2.1. Algoritam modela sličnosti baziran na udaljenosti

Slika 2.1. Prikazuje algoritam za usklađivanje traga kretanja čija je funkcija za mjeru blizine definirana kao najbliža udaljenost do određenog brida, u ovom slučaju ceste. Neusklađene točke se projiciraju na točku na cesti koja im je geometrijski najbliža.

- **Bazirani na uzorcima** - Algoritmi temeljeni na uzorcima koriste povijesne podatke o mapiranju kako bi odgovorili na nove upite o mapiranju pronalazeći slične obrasce putovanja. Pretpostavka je da ljudi obično putuju istim putevima s obzirom na par početnih i odredišnih točaka. Pozivanjem na povijesne trajektorije koje su slične upitu, mogu se dobiti kandidati za putanju bez brige o tome koliko su putanje rijetke. Konkretno, povijesna trajektorija ili trajektorija dobivena spajanjem više povijesnih trajektorija bit će korištena ako je svaka točka te trajektorije unutar sigurnog područja oko upitne trajektorije. Algoritam na kraju koristi funkciju bodovanja za određivanje optimalne rute. Međutim, zbog rijetkosti i nejednakosti povijesnih podataka, putanja možda neće biti potpuno pokrivena povijesnim putanjama, osobito u rijetko posjećenim regijama, što dovodi do korištenja nekakvog drugog pristupa.

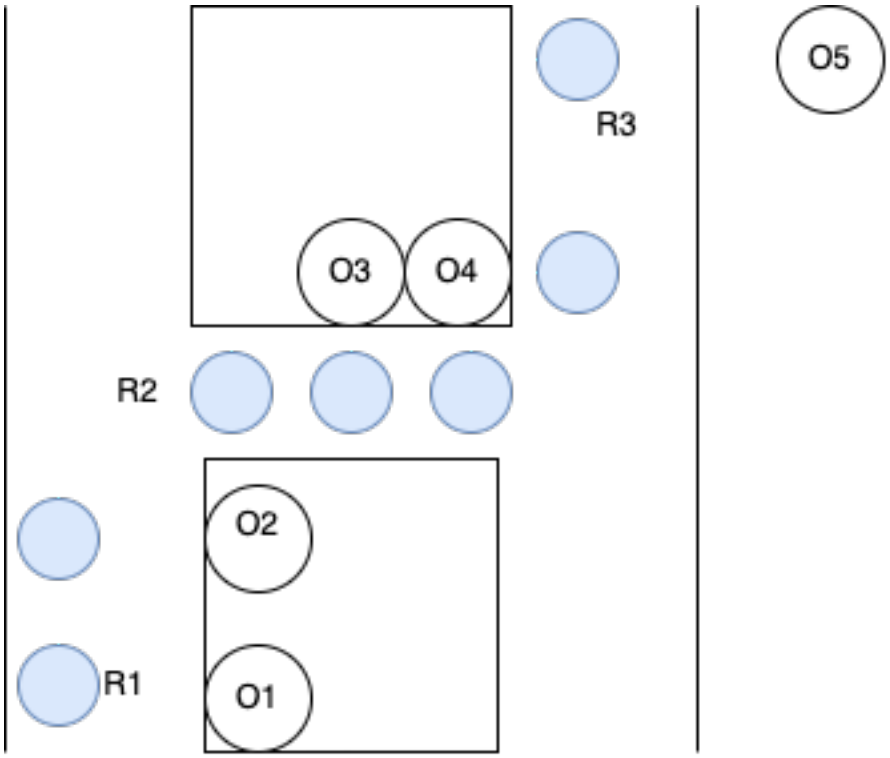


Slika 2.2. Algoritam modela sličnosti baziran na uzorcima

Slika 2.2. prikazuje model sličnosti baziran na uzorcima. On koristi povijesne podatke o tragovima kretanja da bi uskladio trag kretanja između početne i završne točke. Tako u ovom primjeru algoritam vec sadržava podatke o kretanju između prikazanih točaka te onda mapira točke na povijesnu putanju.

2.2. Model prijelaza stanja

Modeli prijelaza stanja grade otežani topološki graf koji sadrži sve moguće rute kojima vozilo može putovati. U ovom grafu, vrhovi predstavljaju moguća stanja u kojima se vozilo može nalaziti u određenom trenutku, dok bridovi predstavljaju prijelaze između stanja u različitim vremenskim oznakama. Za razliku od cestovne mreže, težina elementa grafa predstavlja mogućnost stanja ili prijelaza, a najbolji rezultati podudaranja dolaze od optimalnog puta u grafu na globalnoj razini. Postoje tri glavna načina za izgradnju grafa i rješavanje problema optimalnog puta, a to su skriveni Markovljev model (HMM) (engl.*Hidden Markov model*), uvjetno slučajno polje (CRF) (engl.*Conditional Random Field*) i tehnika otežanog grafa (WGT) (engl.*Weighted Graph Technique*).



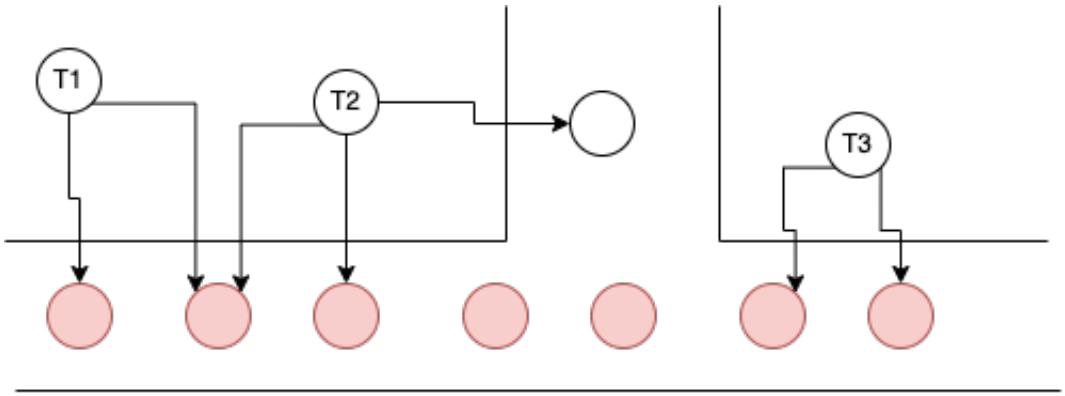
Slika 2.3. Skriveni markovljev model

Slika 2.3. prikazuje primjer skrivenih markovljevih modela. Cilj ovog algoritma je naći najvjerojatnije segmente koji bi proizveli opažene GPS točke. R1,R2,R3 predstavljaju cestovne segmente, točke O1,O2,O3,O4,O5 opažena GPS očitanja. Prvi dio algoritma je za svako GPS opažanje otkriti koji segment ceste koji ga je mogao generirati. Za svako GPS očitanje se pronalazi najbliži cestovni segment. Nakon što je pronađen najbliži segment onda se pronalaze projekcije na segmente (krugovi označeni plavom bojom). Nakon traženja najbližeg segmenta, gradi se model markovljevog modela. Računaju se vjerojatnosti toga da je opažen određeni GPS signal dok vozilo prometuje po segmentu ulice pomoću formula koje ovise o udaljenosti segmenta ulice od opažene GPS točke. Tako se izgrađuje model te se pomoću vjerojatnosti prijelaza određuje najvjerojatnija putanja.

2.3. Model evolucije kandidata

Model evolucije kandidata odnosi se na model koji tijekom procesa usklađivanja traga vozila održava skup kandidata, koje su još poznate i kao čestice ili hipoteze. Skup kandidata se inicijalno formira na temelju prvog uzorka putanje i nastavlja se razvijati dodavanjem novih kandidata koji se propagiraju od starih kandidata koji su po nekakvoj mjeri

blizu najnovijih mjerenja, dok se nerelevantni kandidati izbacuju iz skupa kandidata. Interpretirajući kandidata kao glas, algoritmi održavajući skup kandidata mogu pronaći segment s najviše glasova, čime se određuje put koji se najviše podudara. U usporedbi s modelom prijelaza stanja, model evolucije kandidata je otporniji na problem podudaranja izvan staze jer je trenutačno podudaranje pod utjecajem ne samo prethodno definiranog rješenja, već i drugih kandidata. Tehnika filtriranja čestica (PF) (engl. *The particle filter*) i tehnika višestrukih hipoteza (MHT) (engl. *Multiple Hypothesis Technique*) dva su reprezentativna modela.



Slika 2.4. Model evolucije kandidata

Na slici 2.4. crvene točke izglasavaju najvjerojatniju putanju. Dolaskom novih točaka, vjerojatniji je donji put.

2.4. Model bodovanja

Model bodovanje je grupa algoritama koja koristi težine bez specifičnog modela, dodjeljujući kandidate svakom segmentu putanje i pronalazeći cestovni brid koji maksimizira definiranu funkciju bodovanja. U scenarijima gdje podatci dolaze jedan po jedan, segmenti se vraćaju odmah, dok se u scenarijima gdje su svi podatci odmah dostupni, čekaju na spajanje s drugim segmentima. Najnoviji pristupi postižu mapiranje na razini cestovne trake, identificirajući cestovne trake prema širini ceste i bodujući mreže kandidata. Funkcija bodovanja koristi četiri značajke: blizinu mreže i uzorka putanje, procijenjenu buduću lokaciju vozila, mogućnost dosegnuća i namjeru skretanja. Ove značajke se otežavaju različito u funkciji bodovanja, s koeficijentima dobivenim kroz proces treniranja prije mapiranja.

3. Geoprostorne baze podataka i PostGIS

U ovom poglavlju opisane su geoprostorne baze podataka i baza podataka PostGIS. U poglavlju 3.1 opisane su geoprostorne baze podataka na općenit način. U sljedećem poglavlju, 3.2 detaljno je opisana baza podataka PostGIS i dostupni geoprostorni operatori.

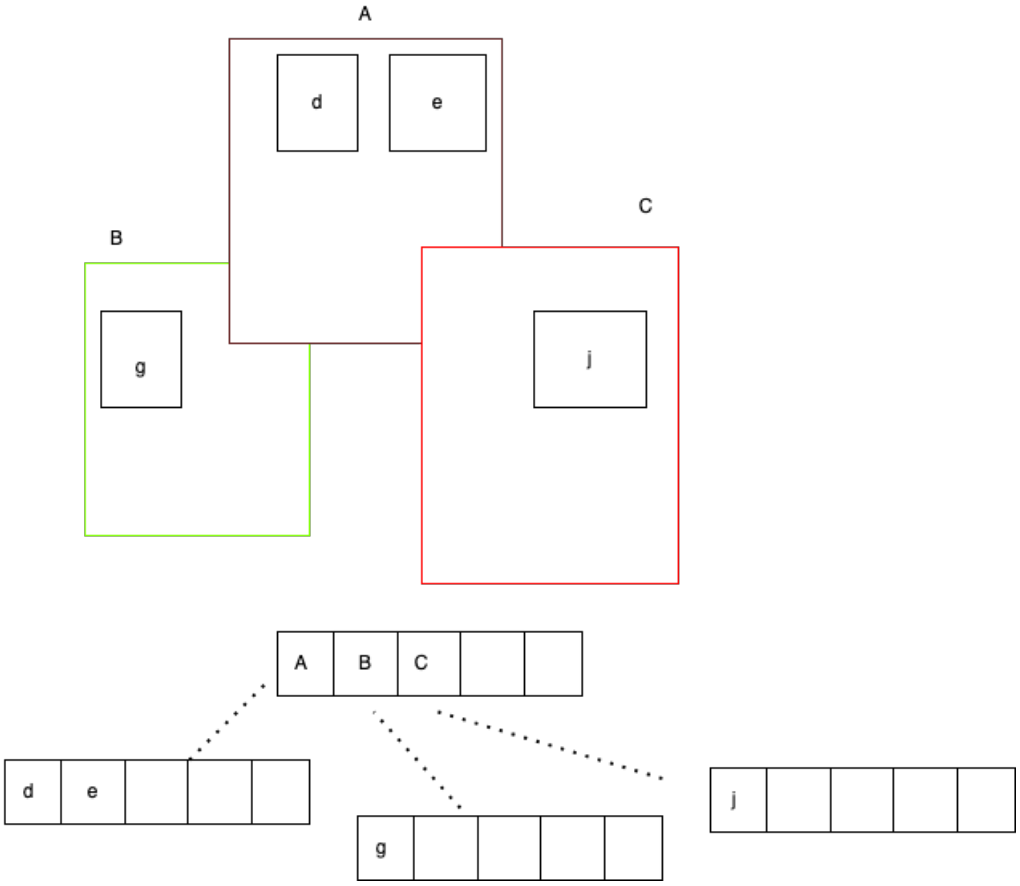
3.1. Geoprostorne baze podataka

Baza podataka je velika kolekcija povezanih podataka pohranjenih unutar računalnog sustava. U tom sustavu podatci su postojani, što znači da podatci ostaju sačuvani i dostupni čak i nakon što se baza podataka zatvori ili se dogodi neki problem. Za razliku od podataka kojima upravljaju obični programski jezici, podatci su većeg obujma i postojani. U modernim računalnim sustavima pojavile su se mnoga "nestandardna" korištenja baza podataka kao što su pohrana prostornih podataka, slika, bioinformatičkih podataka i drugih... Sustav za upravljanje bazama podataka (engl.*Database managment system*) je skup programa koji upravljaju strukturom baze podataka i kontroliraju pristup podatcima pohranjenima u bazi podataka. Generalno sustav za upravljanje bazom podataka se brine oko procesa definiranja baze podataka, konstruiranja baze podataka, manipuliranja bazom podataka, slanjem upita bazi podataka i promjenom podataka u bazi podataka. [5]

Strukture podataka, ograničenja i operacije koje podržava sustav za upravljanje bazama podataka definiran je logičkim modelom podataka koji se pohranjuju u bazi podataka. Danas je najpopularniji relacijski model podataka. U relacijskom okolišu, reprezentacija podataka se odvija korištenjem tablica (engl.*table*). A tablice se sastoje od redaka u kojima se pohranjuju podatci. Manipulacija podataka se odvija korištenjem jezika za manipuliranje podataka (engl.*data manipulation language*), od kojih je najpo-

pularniji SQL (engl.*structured query language*). [5]

Tako se dolazi i do pojma geoprostorne baza podataka. Geoprostorne baze podataka služe za prikupljanje i pohranu točaka, linija i područja kartografskih informacija koje nazivamo prostornim podacima. Ove baze podataka omogućuju korisnicima da izvrše složene prostorne analize i upite na velikim količinama geoprostornih podataka, integrirajući ih s drugim vrstama podataka kako bi pružili vrijedne uvide korisnicima podataka. Geoprostorni podaci se koriste u različitim industrijama, uključujući urbanističko planiranje, upravljanje resursima, transport, telekomunikacije, poljoprivredu, okolišnu znanost i mnoge druge. Osim toga, geoprostorne baze podataka omogućuju praćenje promjena u geografskom prostoru tijekom vremena, što je od ključne važnosti za promatranje okoliša, upravljanje katastrofama i planiranje razvoja. Korištenjem naprednih alata za vizualizaciju, kao što su GIS (engl.*Geographic Information System*) aplikacije, korisnici mogu prikazati složene prostorne podatke u obliku koji je lako razumljiv ljudima. čime se olakšava donošenje odluka. Geoprostorne baze podataka podržavaju razne prostorne operacije kao što su prostorna pretraživanja, operacije spajanja, mjerenje udaljenosti i područja, te analize susjedstva prostornih elemenata. Također prostorne baze podataka koriste i prostorne indekse radi poboljšanja performansi sustava. Ove operacije omogućuju detaljnije analize i bolje razumijevanje geografskih odnosa i obrazaca. Prostorni indeksi omogućuju brzo pretraživanje prostornih podataka, koji se potom koriste u prostornim operacijama. Najvažniji primjeri geoprostornih struktura podataka koji se koriste u svrhu prostornih indeksa su R-stabla, KD-stablo, GeoHash i drugi.



Slika 3.1. R-stablo

Slika 3.1. prikazuje strukturu prostornog indeksa R-stablo. Svaki čvor R-stabla predstavlja pravokutnik koji obuhvaća sve pravokutnike njegovih podređenih čvorova, stvarajući hijerarhiju graničnih okvira 3.5.

Implementacija geoprostornih baza podataka zahtijeva specifične tehnologije i formate podataka, kao što su PostGIS za PostgreSQL baze podataka ili Oracle Spatial. Ovi dodaci omogućuju pohranu, upit i analizu geoprostornih podataka unutar relacijskih baza podataka, čime se proširuju mogućnosti tradicionalnih baza podataka. Korištenjem geoprostornih baza podataka, organizacije mogu učinkovitije upravljati svojim geografskim informacijama, omogućujući bolje donošenje odluka i planiranje[6].

3.2. PostGIS

PostGIS je geoprostorno proširenje za sustav za upravljanje bazom podataka PostgreSQL. PostgreSQL je popularan relacijski sustav za upravljanje bazama podataka otvorenog koda (engl.*open source*), što znači da ga održava globalna zajednica razvojnih inženjera, a ne pojedina kompanija. PostgreSQL je također razvijan s mogućnošću dodavanja proširenja, to omogućava da proširenja mogu lagano dodavati svoje tipove, funkcije i indekse koji se integriraju s bazom podataka. PostGIS pretvara PostgreSQL sustav za upravljanje bazom u prostornu bazu dodajući 3 značajke prostornih baza podataka:

- prostorne tipove
- prostorne indekse
- prostorne operatore

PostGIS je usklađen s `Simple Features for SQl` specifikacijom. Ta specifikacija definirana od strane `Open Geospatial Consortium` [7] organizacije koja definira specifikacije za geoprostorne servise definira tipove i funkcije koja sačinjavaju standardnu prostornu bazu podataka. Svaka PostGIS baza podataka sadrži 2 tablice koje definiraju tipove podataka koji su dostupni u bazi podataka:

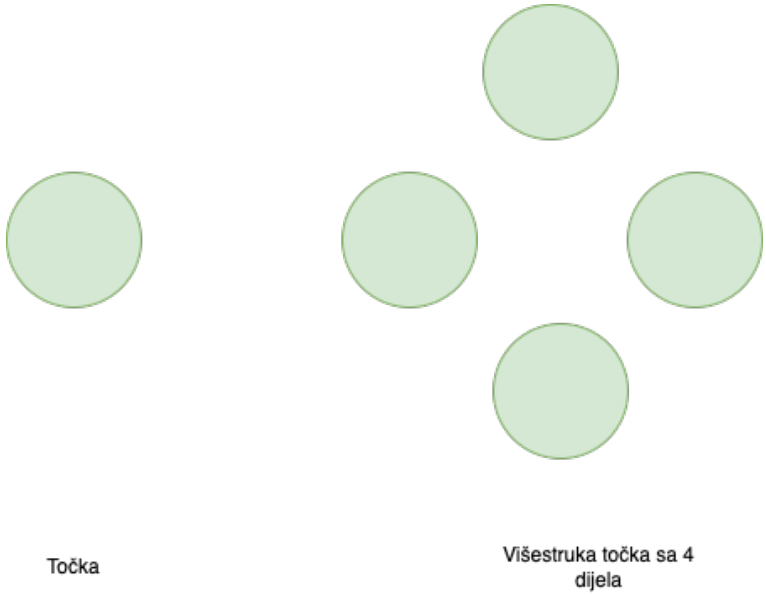
- **spatial_ref_sys** definira sve prostorne referentne sustave poznate bazi podataka.
- **geometry_columns** pruža popis svih "obilježja" (definiranih kao objekti s geometrijskim atributima) i osnovne detalje tih obilježja.

Upitom prema ovoj tablici, GIS klijenti i biblioteke mogu odrediti što očekivati prilikom dohvaćanja podataka te mogu izvršiti potrebne projekcije, obrade ili prikazivanja bez potrebe za inspekcijom svake geometrije.

`Simple Features for SQl` specifikacija, definira kako su stvarni objekti reprezentirani. Ta specifikacija barata samo s dvodimenzionalnim objektima, ali PostGIS to proširuje i na više dimenzionalne reprezentacije. U PostGIS-u imamo više tipova geometrija, informacije o podatcima oko određenih geometrija možemo dobiti korištenjem ugrađenih funkcija koje čitaju metapodatke:

- **ST_GeometryType(geometry)** vraća tip geometrije.
- **ST_NDims(geometry)** vraća broj dimenzija geometrije.
- **ST_SRID(geometry)** vraća identifikacijski broj prostornog referentnog sustava geometrije.

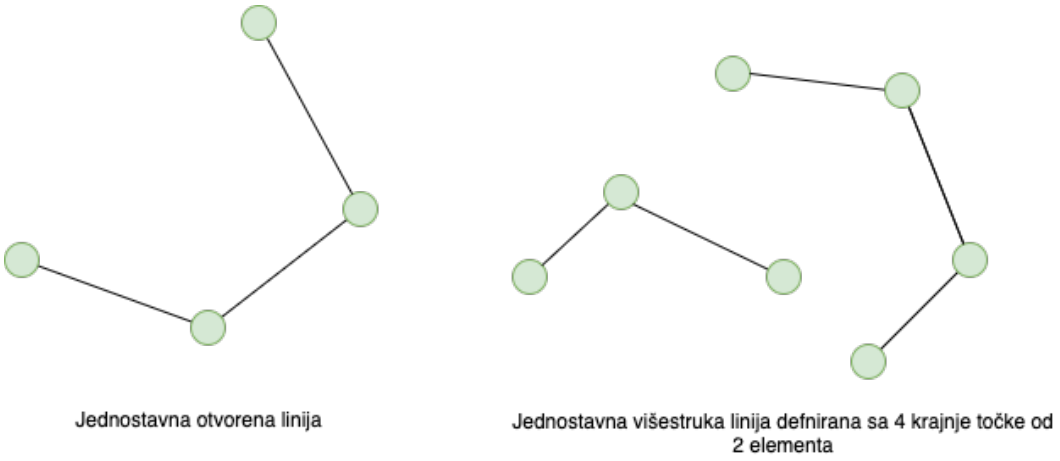
Prva geometrija s kojom se susrećemo je točka (engl. *Point*).



Slika 3.2. Točka

Slika 3.2. prikazuje točku koja predstavlja jednu lokaciju na Zemlji. Točka je reprezentirana kao jedna koordinata u određenoj dimenziji, bilo da je ona dvodimenzionalna, trodimenzionalna ili višedimenzionalna. Specifične funkcije za rad s točkama su:

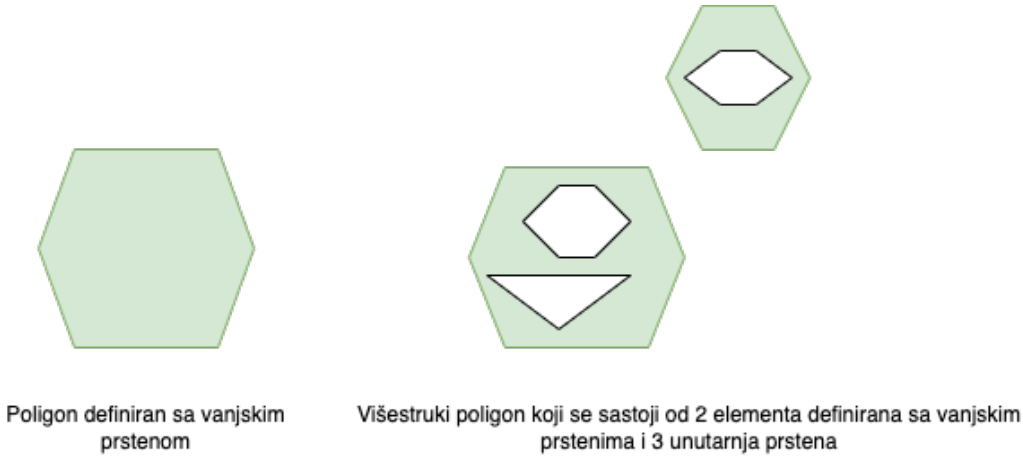
- **ST_X(geometry)** koja vraća prvu koordinatu
- **ST_Y(geometry)** koja vraća drugu koordinatu



Slika 3.3. Linija

Slika 3.3. prikazuje liniju (engl. *linestring*). Linija predstavlja put između dvije točke. Sastoji se od više poredanih točaka. Ceste, rijeke, potoke se najčešće prikazuje s linijama. Za liniju kažemo da je zatvorena (engl. *closed*) ako počinje i završava u istoj točci. Za liniju kažemo da je jednostavna (engl. *simple*) ako se ne dira ili prelazi sama preko sebe osim u krajnjim točkama. Specifične funkcije za rad s linijama su:

- **ST_Length(geometry)** vraća duljinu linije.
- **ST_StartPoint(geometry)** vraća prvu krajnju koordinatu kao točku.
- **ST_EndPoint(geometry)** vraća zadnju krajnju koordinatu kao točku.
- **ST_NPoints(geometry)** vraća broj koordinata u liniji.



Slika 3.4. Poligon

Poligon (engl. *Polygon*) 3.4. je geometrija koja predstavlja nekakvu površinu. Vanjska granica poligona je reprezentirana sa prstenom. Prsten je linija koja je zatvorena i jednostavna. Praznine unutar poligona su isto reprezentirane pomoću linija. Poligoni se koriste kada se reprezentiraju objekti čija je površina i oblik bitan. Primjeri poligona su gradovi, parkovi, države itd... Specifične funkcije za rad s poligonima su:

- **ST_Area(geometry)** vraća površinu poligona.
- **ST_NRings(geometry)** vraća broj prstenova.
- **ST_ExteriorRing(geometry)** vraća vanjski prsten kao liniju.
- **ST_InteriorRingN(geometry, n)** vraća određeni unutarnji prsten kao liniju
- **ST_Perimeter(geometry)** vraća duljinu svih prstenova.

Kolekcije su setovi u kojima se nalaze jednostavnije geometrije, one su prikazane u 3.2., 3.3. i 3.4. Postoje četiri tipa kolekcija:

- **MultiPoint**, kolekcija točaka.
- **MultiLineString**, kolekcija linija.
- **MultiPolygon**, kolekcija poligona.
- **GeometryCollection**, heterogena kolekcija bilo koje geometrije (uključujući druge kolekcije.

Prostorne baze podataka osim što mogu skladištiti geometrije, one mogu također i uspoređivati veze između geometrija. Tako PostGIS podržava operacije:

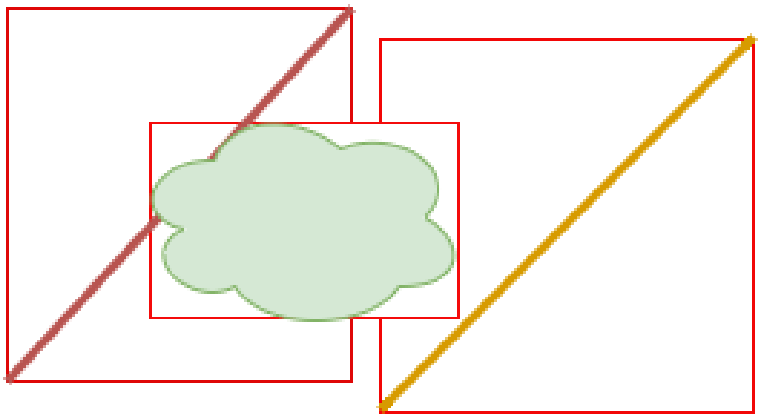
- **ST_Equals(geometry A, geometry B)** provjerava prostornu jednakost dviju geometrija. Vraća TRUE ako obje geometrije imaju identične x,y koordinate. Na primjer, dvije točke se smatraju jednakima ako su njihove koordinate identične.
- **ST_Intersects(geometry A, geometry B)** vraća TRUE ako dvije geometrije dijele bilo koji prostor, tj. ako se njihove granice ili unutrašnjosti sijeku. Na primjer, linija koja prolazi kroz poligon će rezultirati vrijednošću TRUE.

- **ST_Disjoint(geometry A, geometry B)** vraća TRUE ako se dvije geometrije ne sijeku, što znači da nemaju zajednički prostor. Ova funkcija je suprotna od `ST_Intersects`.
- **ST_Crosses(geometry A, geometry B)** vraća TRUE ako se dvije geometrije sijeku tako da sjecište ima dimenziju manju od maksimalne dimenzije dviju geometrija. Na primjer, linija koja prelazi poligon će rezultirati vrijednošću TRUE.
- **ST_Overlaps(geometry A, geometry B)** uspoređuje dvije geometrije iste dimenzije i vraća TRUE ako njihov zajednički dio tvori novu geometriju različitu od obje izvorne, ali iste dimenzije. Na primjer, dva poligona koji dijele dio prostora, ali nisu potpuno sadržani jedan u drugom.
- **ST_Touches(geometry A, geometry B)** vraća TRUE ako se dvije geometrije dodiruju samo na njihovim granicama, ali se ne sijeku u unutrašnjostima. Na primjer, dva poligona koji dijele samo zajedničku rubnu liniju.
- **ST_Within(geometry A, geometry B)** vraća TRUE ako je prva geometrija potpuno unutar druge geometrije. Na primjer, točka unutar poligona ili manji poligon unutar većeg poligona.
- **ST_Contains(geometry A, geometry B)** vraća TRUE ako druga geometrija u potpunosti leži unutar prve geometrije. Ova funkcija je suprotna od `ST_Within`.
- **ST_Distance(geometry A, geometry B)** izračunava najkraću udaljenost između dviju geometrija i vraća je kao broj s pomičnim zarezom. Na primjer, udaljenost između točke i linije.
- **ST_DWithin(geometry A, geometry B, distance)** vraća TRUE ako su dvije geometrije unutar zadane udaljenosti jedna od druge. Na primjer, određivanje svih objekata unutar 100 metara od određene točke.

Kao i uobičajene baze podataka PostGIS podržava različite operacija spajanja. Tako se u PostGIS-u koriste operacije prostornog spajanja. Na taj način se kombiniraju podatci iz različitih tablica gdje se operacija spajanja obavlja pomoću neke od funkcija za prostorno uspoređivanje. Svaka od funkcija koja vraća istinito/lažno vezu se može koristiti za spajanje dvaju tablica, ali najčešće korištene su `ST_Intersects`, `ST_Contains` i

`ST_DWithin`.

Prostorni indeksi su jedan od glavnih značajki koji definiraju prostorne baze podataka. U klasičnim bazama podataka indeksi omogućuju brže dohvaćanje podataka, koristeći određene struktura podataka kao što su B-stabla. Indeksi omogućuju korištenje velikih prostornih podataka mogućim. Bez korištenja indeksa, svaki upit bi se morao odvijati sekvencijalno. Indeksiranje ubrzava pretraživanje prostornih podataka tako da organizira podatke u stabla koja mogu biti brzo pretražena.



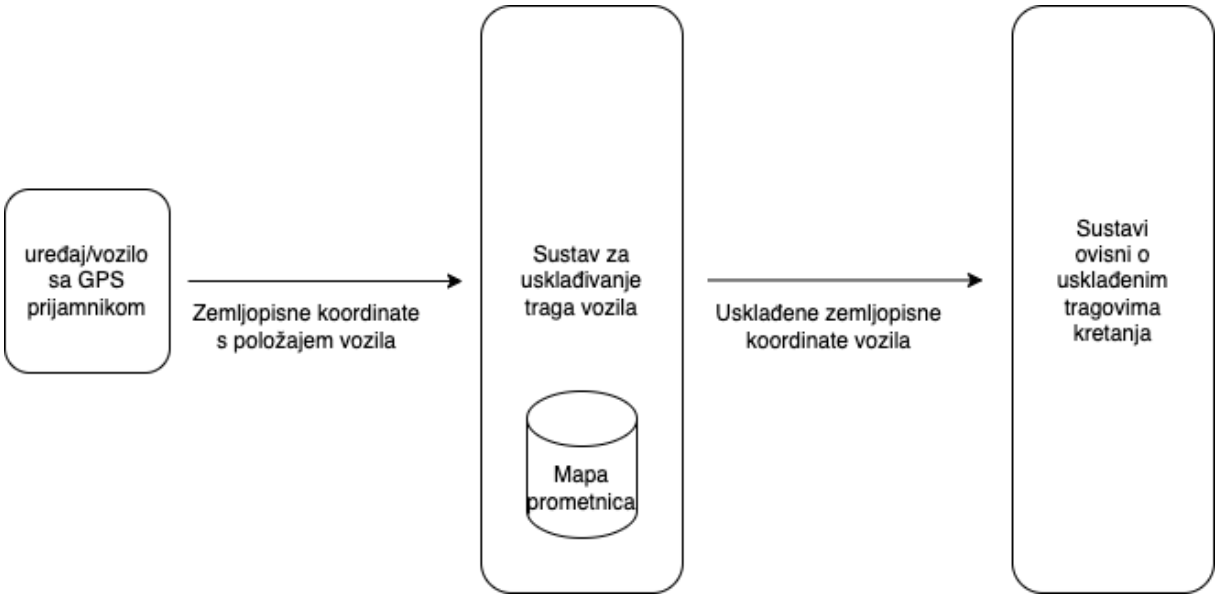
Slika 3.5. Granični okvir

Standardne baze podataka kreiraju hijerarhijska stabla utemeljena na vrijednostima indeksiranih stupaca. Prostorni indeksi su drugačiji, oni ne mogu indeksirati geometrijske strukture kao i obične podatke. Za indeksiranje podataka u prostornim bazama se koriste granični okviri (engl. *bounding box*). Granični okviri je pravokutni oblik koji potpuno okružuje nekakav geometrijski oblik. Na slici 3.5. su prikazani granični okviri za geometrijske oblike označeni crvenom bojom. Broj linija koje sijeku zeleni oblak je jedna, ali ako se gledaju brojevi graničnih okvira koji sijeku zeleni oblak dolazi se do broja dva. Na taj način baza podataka efikasno vraća objekte koji sijeku druge objekte. Prvo se odvija upit koji nalazi broj objekata čiji granični okviri sijeku granični okvir objekta za koji je postavljen upit, a tek potom se pretražuju stvarne granice objekta. Na taj način se mnogostruko smanjuje broj kalkulacija potrebnih za izvođenje određenog upita. PostGIS svoje indekse bazira na R-stablima [8]. R-stabla dijele podatke na pravokutnike, pod-pravokutnike i manje pravokutne blokove omogućujući efikasno prostorno pretraživanje [9].

Zbog zakrivljenosti zemlje u stvarnom svijetu, Zemljinu površinu nije jednostavno

postaviti na ravnu površinu. Razvijene su razne projekcije za taj zadatak, svaka sa svojim prednostima i nedostacima. Neke projekcije zadržavaju površine, omogućujući očuvanje relativne veličine objekata, druge projekcije kao što je Mercatorova, zadržavaju točnost kuteva. Zajedničko svim projekcijama je to da transformiraju sferičnu površinu zemlje na ravnu površinu Kartezijevog koordinatnog sustava. U PostGIS bazi podataka geografski koordinatni sustav i projekcija su definirane svojim SRID-om (engl.*Spatial Reference IDentifier*), što je jedinstveni identifikator koji kombinira sve informacije o projekciji u jedan broj [10]. SRID i koordinata zajedno definiraju jednu lokaciju na zemljinoj površini, bez SRID-a koordinata je samo apstraktna notacija. Tijekom učitavanja podataka podatci mogu biti učitani u projekciji koja nije odgovarajuća za operacije koje se žele izvesti. PostGIS zato omogućuje transformacija podataka iz jednog SRID-a u drugi odgovarajući SRID funkcijom `ST_Transform(geometry, srid)`.

4. Arhitektura sustava



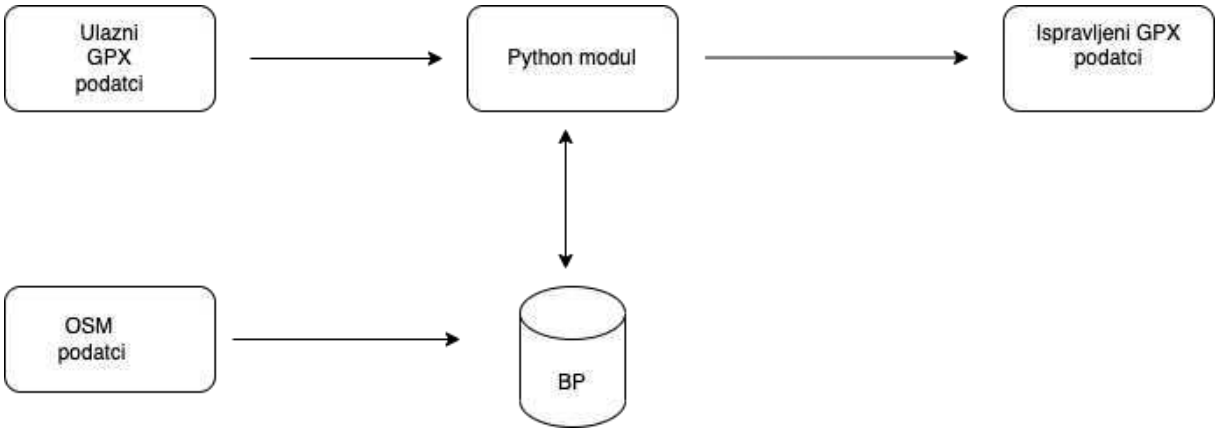
Slika 4.1. Arhitektura općeg sustava

Slika 4.1. prikazuje arhitekturu općeg sustava kod kojega bi ovaj rad bio samo jedan dio većeg sustava. Sustav započinje s ulaznim GPS podacima, koji dolaze iz GPS uređaja i predstavljaju sirove podatke o lokaciji vozila koje imaju greške, to jest nisu još ispravljene da bi pratile pravu lokaciju prometnica. Ti podatci ulaze u sustav za usklađivanje traga vozila, koji je u središtu dijagrama i predstavlja implementaciju iz rada. Clj ovog sustava je ispraviti eventualne pogreške i netočnosti u ulaznim GPS podacima te ih uskladiti s cestovnom mrežom, osiguravajući precizne i pouzdane podatke o kretanju vozila. Ispravljeni podatci čine ulaz za sustave kojima su potrebni točne lokacije vozila za obavljanje svoje svrhe.

5. Ostvarenje sustava za usklađivanje traga kretanja cestovnih vozila

U ovom poglavlju bit će prikazano programsko ostvarenje sustava za usklađivanje traga kretanja cestovnih vozila korištenjem PostgreSQL baze podataka s PostGIS bazom podataka. Programsko ostvarenje je napisano u programskom jeziku Python uz korištenje SQL (engl. *Structured query language*) naredbi za interakciju s bazom podataka. Razvijeni sustav koristi GPS podatke traga vozila u `.gpx` formatu i prostorne operatore kako bi precizno mapirao kretanje vozila na cestovnu mrežu. Ovaj proces uključuje parsiranje GPS datoteka, korištenje SQL upita za manipulaciju podacima, te primjenu PostGIS prostornih operatora za usklađivanje traga korištenjem podataka iz logičkog modela prometnica preuzetog iz zemljopisne karte OpenStreetMap. Implementacija će biti detaljno opisana kroz nekoliko potpoglavlja. U poglavlju 5.1 je opisano oblikovanje programskog rješenja. U poglavlju 5.2 će biti opisan rad sa prostornim podatcima. U poglavlju 5.3 je prikazana programska implementacija rješenja u programskom jeziku Python.

5.1. Oblikovanje programskog ostvarenje

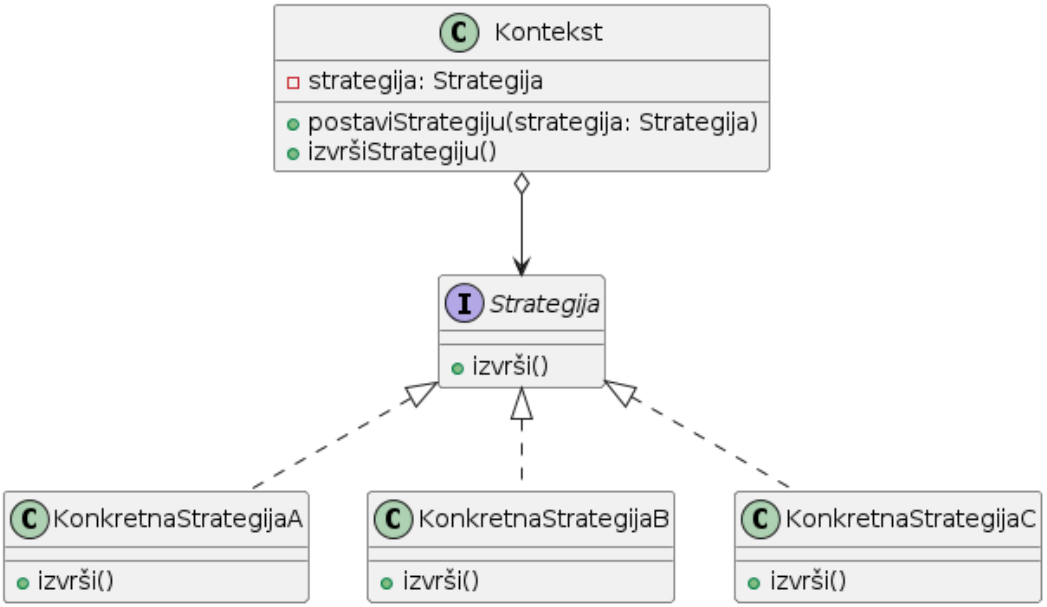


Slika 5.1. Programsko ostvarenje

Slika 5.1. prikazuje programsko ostvarenje sustava. Ulazi za programsko ostvarenje su GPX podatci koji se nalaze u datoteci i OSM podatci koji se učitavaju u bazu podataka. Python modul koristi te podatke za algoritam usklađivanja kretanja traga cestovnim vozila. Izlaz programa su ispravljeni tragovi kretanja u formatu GPX datoteke.

Programsko rješenje je oblikovano korištenjem oblikovnog obrasca strategija (engl. *Strategy*). Oblikovni obrasci su tipična rješenja za probleme koji se često pojavljuju u dizajnu programskih rješenja. Oni su poput unaprijed izrađenih nacrta koji se prilagođavaju rješenju specifičnog problema. Oblikovni obrazac, za razliku od algoritma, predstavlja apstraktniji i visokorazinski opis rješenja [11].

Slika 5.2. prikazuje uml dijagram oblikovnog obrasca strategija. Strategija je ponašajni oblikovni obrazac (engl.*behavioral design pattern*) koji omogućuje definiranje familije algoritama koji su međusobno zamjenjivi. Osnovna ideja strategije je da se uzmu klase koje obavljaju specifične zadatke na različite načine i te algoritme izdvoji u zasebne klase, nazvane strategije, koje su međusobno izmjenjive. `Kontekst` je klasa koja drži referencu na jednu od implementacija sučelja `Strategija`. Kontekst delegira izvođenje funkcija jednoj određenoj klasi koja implementira sučelje `strategija` umjesto da sam izvodi te funkcije. `Kontekst` sam ne odabire odgovarajući algoritam za izvođenje operacije, već njega instancira klijent koji koristi klasu. `Kontekst` radi sa svim implementacija sučelja `Strategija` jednako, spremajući referencu na implementaciju i onda pozivajući funkciju koju implementacije sučelja implementiraju. Tako se dolazi do laganog dodavanja novih implementacija sučelja bez mijenjanja koda klase `Kontekst`. Sučelje



Slika 5.2. Uml dijagram oblikovnog obrasca strategija

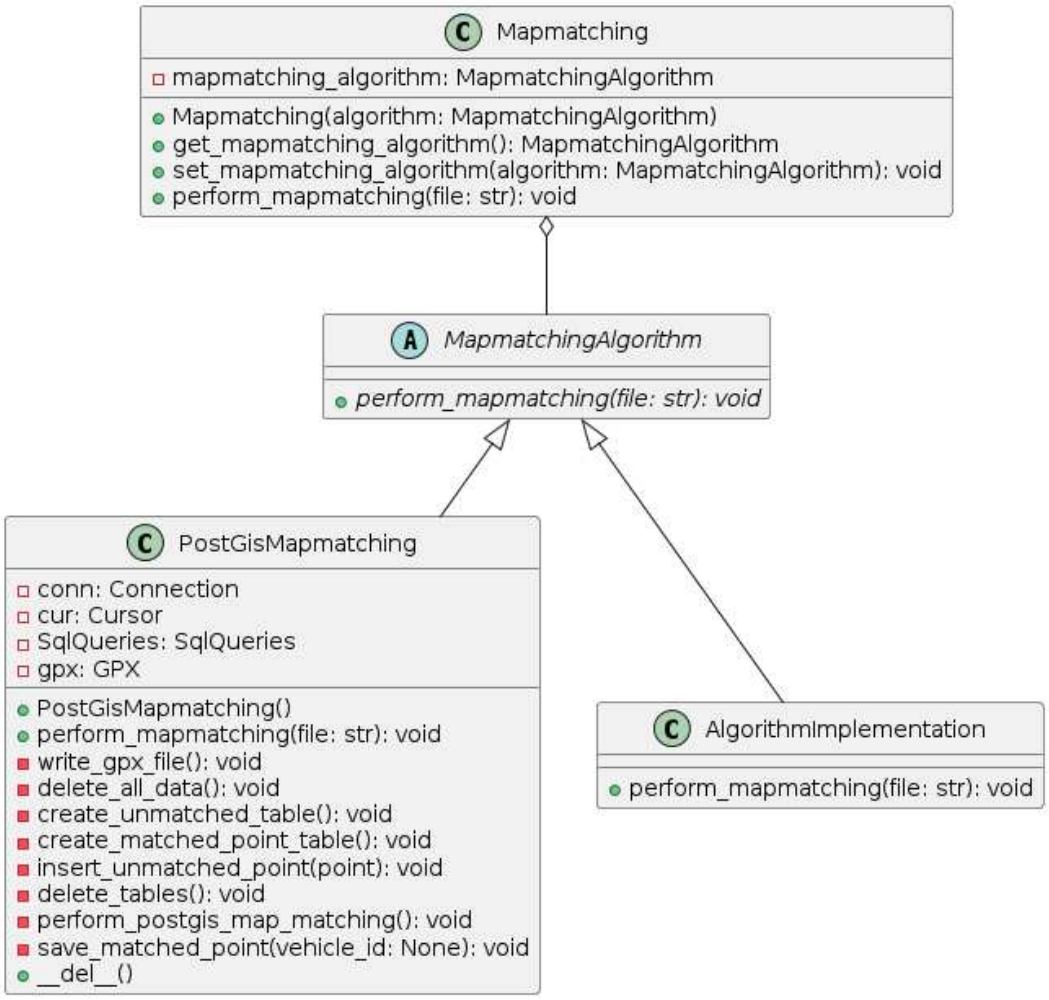
`strategija` definira funkcije koje su zajedničke svim `konkretnim implementacijama`. `Konkretne implementacije` implementiraju sučelje `strategija` svaka na svoj način te se tako dolazi do različitih načina izvođenja neke funkcionalnosti, samo s kreiranjem nove `Konkretne strategije`. Primjena ovog obrasca čini programski kod modularnim i lakšim za održavanje, omogućujući brzu prilagodbu promjenama zahtjeva bez značajnih izmjena u osnovnoj strukturi programa [12].

SLika 5.3. prikazuje uml dijagram izrađenog programskog rješenja u programskom jeziku Python. Ono koristi oblikovni obrazac strategija kako bi se omogućila jednostavna zamjena algoritama za usklađivanje traga kretanja vozila. Dodavanje još jedne implementacije bi se obavimo dodavanjem klase koja implementira `MapmatchingAlgorithm` apstraktnu klasu.

```
class Mapmatching():

    def __init__(self, algorithm: MapmatchingAlgorithm):
        self.mapmatching_algorithm = algorithm

    def get_mapmatching_algorithm(self) -> MapmatchingAlgorithm:
        return self.mapmatching_algorithm
```



Slika 5.3. Uml dijagram izrađenog programskog rješenja

```
def set_mapmatching_algorithm(self, algorithm: MapmatchingAlgorithm):
    self.mapmatching_algorithm = algorithm


def perform_mapmatching(self, file: str):
    self.mapmatching_algorithm.perform_mapmatching(file)
```

Klasa `Mapmatching` u oblikovnom obrascu strategija ima ulogu konteksta. Ona u sebi drži referencu na objekt tipa `MapmatchingAlgorithm`.
Pomoću metode `set_mapmatching_algorithm(self, algorithm: MapmatchingAlgorithm)` omogućeno je mijenjanje reference za algoritam za usklađivanje kretanja vozila, a pomoću metode `get_mapmatching_algorithm(self) -> MapmatchingAlgorithm` omogućuje dohvaćanje objekta implementacije algoritma. Klasa `Mapmatching` delegira izvođenje algoritma objektu na kojega drži referencu preko funkcije `perform_mapmatching(self,`

file:str).

```
class MapmatchingAlgorithm(ABC):

@abstractmethod
def perform_mapmatching(self, file: str):
    pass
```

Klasa `MapmatchingAlgorithm` u oblikovnom obrascu strategija ima ulogu sučelja strategija kojega implementiraju sve konkretne klase. Klasa MapmatchingAlgorithm nasljeđuje ABC (engl. *Abstract Base Class*), što znači da se radi o apstraktnoj klasi. Apstraktne klase ne mogu biti instancirane i služe kao osnovne klase koje druge klase nasljeđuju. Metoda `perform_mapmatching` je označena dekoratorom `@abstractmethod`. To znači da svaka klasa koja nasljeđuje `MapmatchingAlgorithm` mora implementirati ovu metodu. Apstraktne metode definiraju koje metode moraju biti implementirane u potklasama, ali ne daju implementaciju tih metoda.

```
class PostGisMapmatching(MapmatchingAlgorithm):
def __init__(self):
    super().__init__()
    ...

def perform_mapmatching(self, file: str):
    ...
```

Klasa `PostGisMapmatching` u oblikovnom obrascu strategija ima ulogu konkretne strategije. Ova klasa je konkretna implementacija algoritma za usklađivanje traga kretanja vozila, prilagođena za korištenje s PostGIS bazom podataka. Klasa `PostGisMapmatching` nasljeđuje apstraktnu klasu `MapmatchingAlgorithm`, što znači da mora implementirati sve apstraktne metode definirane u `MapmatchingAlgorithm`.
Metoda `perform_mapmatching(self, file: str)` implementira apstraktnu metodu iz `MapmatchingAlgorithm` apstraktne klase.

```
def main():
    algorithm = Mapmatching(PostGisMapmatching())
```

```
        algorithm.perform_mapmatching('input.gpx')
```

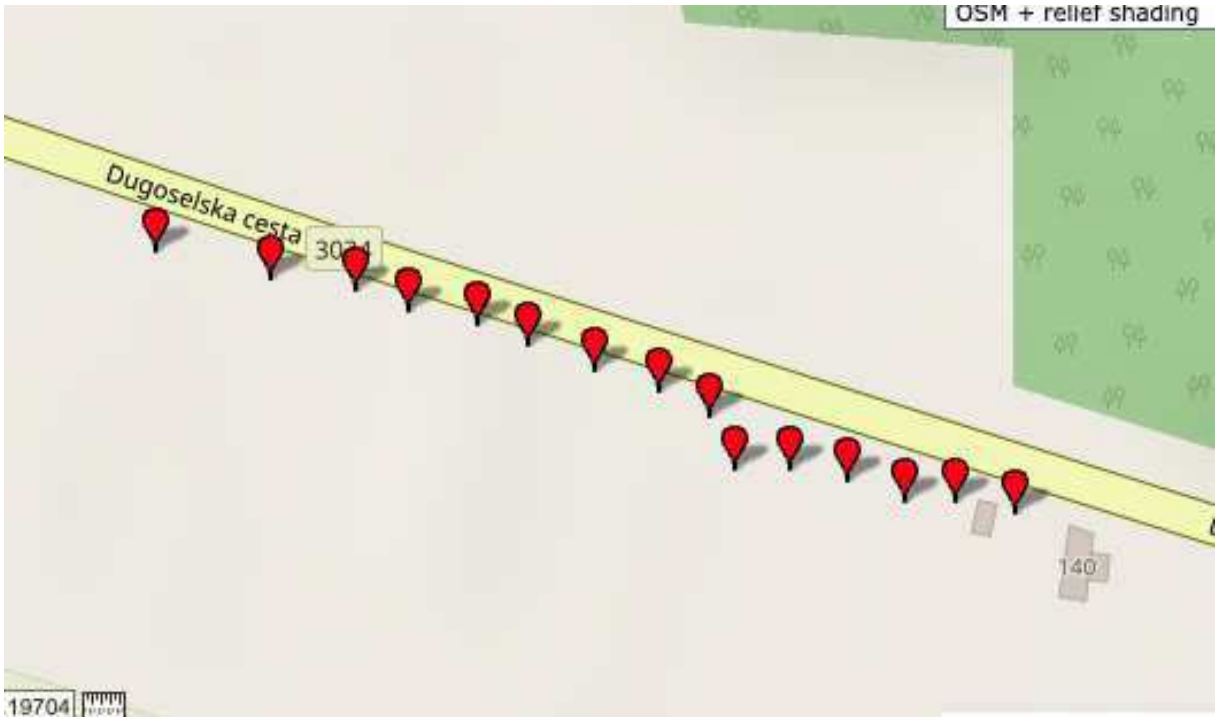
Mijenjanje implementacija se obavlja dodavanjem Konkretne implementacije u instancu MapMatching klase.

5.2. Rad sa prostornim podatcima

Ulazni podatci koji sadrže tragove kretanja vozila za izrađeni sustav su datoteke tipa `.gpx`. GPX (engl.*GPS Exchange Format*) je jednostavni XML format za razmjenu GPS podataka (putne točke, rute i tragovi kretanja) između aplikacija i mrežnih servisa na internetu. GPX datoteke omogućuju interoperabilnost između različitih GPS uređaja i aplikacija, olakšavajući dijeljenje i analizu podataka o kretanju. Svaka GPX datoteka može sadržavati više vrsta podataka, uključujući precizne zemljopisne koordinate, nadmorsku visinu i vremenske oznake, što omogućuje detaljno praćenje putanja. Struktura datoteke je fleksibilna, podržava proširenja i prilagodbe specifičnim potrebama korisnika. Kao rezultat toga, GPX format se široko koristi u različitim aplikacijama, uključujući planiranje putovanja, sportske aktivnosti i mnoge druge [13].

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.1" ...>
        <wpt lat="45.8119236" lon="16.1950386">
         <extensions>
            <vehicleId>12345</vehicleId>
        </extensions>
        </wpt>
        <wpt lat="45.8118338" lon="16.195575">
         <extensions>
            <vehicleId>12345</vehicleId>
        </extensions>
        </wpt>
            .
            .
            .
 </gpx>
```

Prikazani isječak iz ulazne `.gpx` datoteke prikazuje format tragova kretanja vozila. Gpx datoteka započinje sa `XML` deklaracijom koja definira verziju i način kodiranja datoteke. Početak `GPX` dokumenta se označava sa `<gpx>` elementom. Unutar njega se nalaze `<wpt>` elementi koji predstavljaju točke kretanja traga vozila. Svaka točka traga kretanja vozila ima atribute `lat` i `lon`, koji definiraju geografsku širinu i dužinu. Unutar svakog `<wpt>` elementa nalazi se `<extensions>` element koji omogućuje dodavanje dodatnih informacija specifičnih za aplikaciju, u ovom slučaju ID vozila `<vehicleId>`. Ovo omogućava praćenje pojedinačnih vozila kroz njihove jedinstvene identifikatore. Za potrebe rada GPX datoteke se vizualiziraju koristeći mrežni vizualizator GPX datoteka [14].



Slika 5.4. Trag kretanja vozila

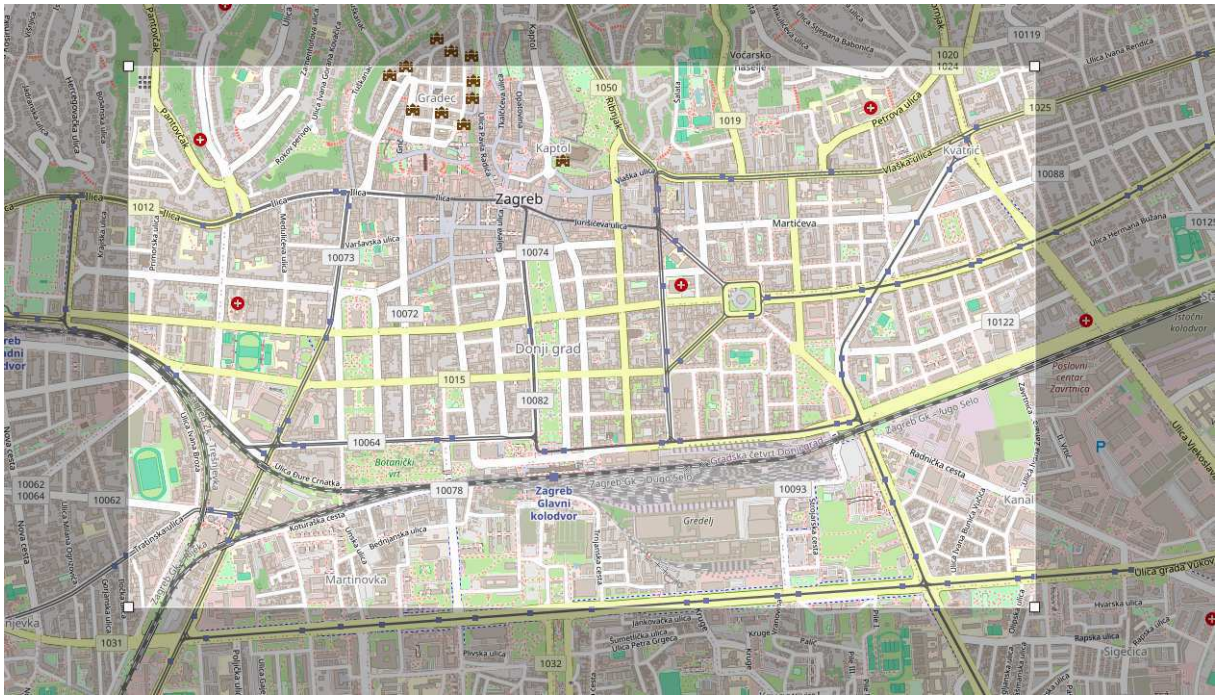
Slika 5.4. prikazuje vizualiziranu GPX datoteku. Ona predstavlja primjer ne usklađenog traga kretanja vozila koje je ulaz za programsko rješenje za usklađivanje traga kretanja vozila.

Logički model zemljine površine je preuzet iz zemljopisne karte `OpenStreetMap`. OpenStreetMap je besplatna mapa cijelog svijeta koju volonteri grade uglavnom od nule i objavljuju pod licencom otvorenog sadržaja (engl.*open-content*). Licenca OpenStreetMap omogućuje besplatan (ili gotovo besplatan) pristup slikama mapa i svim osnovnim podacima. Projekt ima za cilj promicanje novih i zanimljivih načina korištenja tih podataka [15]. OpenStreetMap (OSM) podatci se pohranjuju u XMl formatu, a se sastoje od elemenata, od kojih svi mogu imati pridružene oznake (engl.*tags*). Oznake su parovi `ključ=vrijednost` koji opisuje što je element. Elementi su [16]:

- **Čvorovi (engl. *Nodes*)**: Točke koje označavaju lokacije. Čvorovi mogu biti zasebni ili povezani.
- **Putevi (engl. *Ways*)**: Povezane linije čvorova. Koriste se za kreiranje cesta, staza, rijeka, i slično.
- **Zatvoreni putevi (engl. *Closed ways*)**: Putevi koji formiraju zatvorenu petlju.

Obično definiraju površine.

- **Površine (engl. *Areas*)**: Zatvoreni putevi koji su ispunjeni. Površine su obično implicirane kod zatvorenih puteva.
- **Relacije (engl. *Relations*)**: Koriste se za kreiranje složenijih oblika ili za predstavljanje elemenata koji su povezani na nekakav način, ali nisu fizički povezani.



Slika 5.5. Preuzimanje podataka sa OpenStreetMapa

Preuzimanje podataka iz OpenstreetMapa se može obaviti na više načina, mogu se preuzeti podatci o određenoj geografskoj poziciji ili podatci o cijelom svijetu. Slika 5.5. prikazuje preuzimanje kartografskih podataka o određenoj površini korištenjem mrežnog grafičkog sučelja s mrežne stranice OpenStreetMapa [17]. Preuzeti podatci se nalaze u `.osm` formatu. Podatci se potom učitavaju u instancu PosgreSql baze podataka.

Za potrebe izrade programskog rješenja korištena je baza podataka PostgreSql sa proširenjem PostGIS. Kreiran je PostgreSql server pokrenut na `localhostu` i kreiranja je baza podataka `postGisMapmatching` u kojoj se nalaze podatci potrebni za proces usklađivanja traga kretanja cestovnih vozila. Podatci preuzeti iz OpenStreetMap-a se učitavaju iz `.osm` datoteka u pokrenutu bazu podataka korištenjem alata `osm2pgsql`.

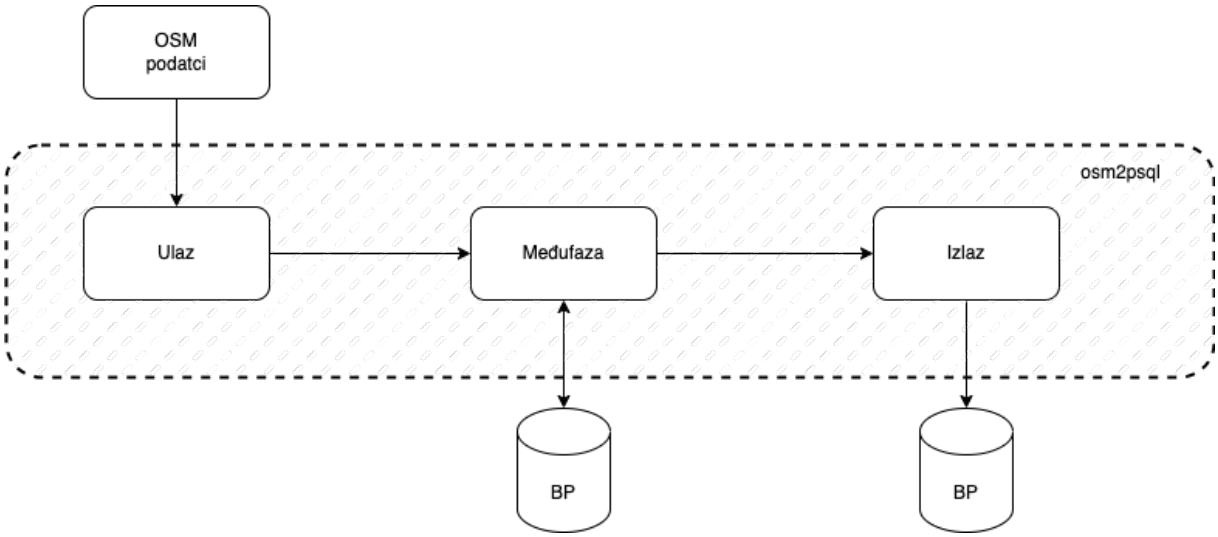
Osm2pgsql se koristi za učitavanje `OSM` podataka u PostgreSQL/PostGIS bazu poda-

taka radi renderiranja mapa i mnogih drugih namjena. Obično je samo dio potrebnih alata, jer je potrebno dodatno progamsko rješenje za stvarno renderiranje (pretvaranje podataka u kartu) ili isporuku mapa korisnicima. Jedan od najvažnijih uloga `osm2psql-a` je pretvaranje OSM textttOSM podataka u PostGIS geometrije koje su opisane u 3.2. Osm2pgsql slaže sve podatke iz povezanih objekata u valjane geometrije.

Tablica 5.1. OSM elementi i ekvivalentne PostGIS geometrije

OSM element	PostGIS geometrija
Točka	Kreirano iz čvorova
Linija	Kreirano iz puteva
Poligon	Kreirano iz zatvorenih puteva ili nekih relacija
Višestruka točka	Kreirano iz čvorova ili nekih relacija
Višestruka linija	Kreirano iz (razdvojenih) puteva ili nekih relacija
Višestruki poligon	Kreirano iz zatvorenih puteva ili nekih relacija
Kolekcija geometrija	Kreirano iz relacija

Tablica 5.1. prikazuje koje OSM elemente osm2psl pretvara u koje PostGIS geometrije. `Osm2pgsql` obično kreira najjednostavniju moguću geometriju za dane podatke. Čvorovi se pretvaraju u točke, putevi u linije ili poligone. Relacije višestrukih poligona mogu se pretvoriti u poligon ili višestruki poligon, ovisno o broju vanjskih prstena. Slično tome, relacija ruta može se pretvoriti u liniju ako je ruta spojena od početka do kraja ili u višestruku liniju ako je ruta nepovezana ili postoje mjesta gdje se ruta dijeli na više dijelova. Tijekom učitavanja alat može doći u doticaj sa nevalidnim elementima koji ne podliježu pravilima, a to onda može uzrokovati probleme tijekom učitavanja OSM podataka. Točke su uvijek valjane sve dok su koordinate unutar ispravnog raspona. Geometrije linija su valjane ako imaju barem dvije različite točke. Osm2pgsql spaja uzastopne identične točke u liniji u jednu točku. Linije koje presijecaju same sebe su još uvijek valjane. Valjanost je složenija za poligone i višestruke poligone. Postoji mnogo načina na koje takva geometrija može biti nevaljana. Na primjer, ako je granica poligona nacrtana u obliku osmice, rezultat neće biti valjan poligon. Baza podataka će pohraniti takve nevaljane poligone, ali to može uzrokovati probleme pri crtanju ili prilikom izvođenja izračuna na temelju nevaljane geometrije. `Osm2pgsql` osigurava da u bazi podataka nema nevaljanih geometrija, bilo da ih ne uvozi uopće ili pomoću provjere `ST_IsValid()` nakon uvoza. U suprotnom će redovi geometrije u bazi sadržavati NULL ili redak neće biti umetnut, a to ovisi o konfiguraciji [18].



Slika 5.6. Procesiranje OSM podataka

Slika 5.6. prikazuje procesiranje OSM podataka koristeći osm2psql alat kroz nekoliko međukoraka. Procesiranje je podijeljeno u nekoliko koraka [18]:

- **Ulaz** (engl.*Input*) čita OSM podatke iz OSM datoteke. OSM datoteka može biti u različitim formatima, poput .osm, .pbf ili .o5m. U ovom koraku, osm2pgsql koristi podatke o čvorovima, putovima i relacijama za daljnju obradu.
- **Međufaza** (engl.*Middle*) pohranjuje sve objekte i prati odnose između objekata. Ovaj korak uključuje održavanje odnosa između čvorova, putova i relacija, te omogućuje optimizirano upravljanje podacima. Osm2pgsql koristi indeksiranje i napredne tehnike za učinkovito praćenje i obradu podataka.
- **Izlaz** (engl.*Output*) transformira podatke i učitava ih u bazu podataka. Ovaj korak uključuje mapiranje OSM podataka na odgovarajuće PostGIS geometrije, kao što su točke, linije i poligoni.

```
osm2pgsql -c -d postgres -U postgres -W -H localhost -C 2048
--hstore --slim -P 5432 map.osm
```

Prikazana naredba je korištena za učitavanje OSM podataka u bazu podataka PostgreSql.

- **-c**: Omogućava kreiranje novih tablica. Ako tablice već postoje, one će biti obrisane.

- **-d postgres**: Određuje naziv PostgreSQL baze podataka. U ovom slučaju, baza se zove `postgres`.
- **-U postgres**: Specifiira korisničko ime za bazu podataka. U ovom slučaju, korisničko ime je `postgres`.
- **-W**: Omogućava unos lozinke za bazu podataka prilikom izvršavanja naredbe.
- **-H localhost**: Određuje host gdje se nalazi PostgreSQL server. U ovom slučaju, to je `localhost`.
- **-C 2048**: Postavlja maksimalnu veličinu radne memorije (engl.*cache*) na 2048 MB.
- **–hstore**: Omogućava korištenje `hstore` PostgreSQL ekstenzije za pohranu tagova.
- **–slim**: Omogućava upotrebu privremenih tablica za pohranu međupodataka tijekom procesa uvoza.
- **-P 5432**: Postavlja port na kojem PostgreSQL server sluša. U ovom slučaju, to je port `5432`.
- **map.osm**: Putanja do OSM datoteke koja se učitava.

Nakon učitavanja OSM podataka sa alatom `osm2psql` u bazi podatka će se pojaviti osam tablica. Svaka od ovih tablica ima specifičnu ulogu i strukturu, koja omogućuje pohranu različitih vrsta geometrijskih podataka i metapodataka. U nastavku su opisane funkcije i sadržaji ovih tablica:

- **osm2pgsql_properties**: Ova tablica pohranjuje metapodatke i svojstva koja su povezana s procesom uvoza podataka pomoću osm2pgsql. Služi za praćenje statusa i postavki uvoza.
- **planet_osm_line**: Sadrži sve linijske objekte iz OSM podataka. To uključuje ceste, rijeke, željezničke pruge i slične linearne značajke koje nisu zatvoreni poligoni.
- **planet_osm_nodes**: Pohranjuje sve čvorove (točke) iz OSM podataka. Čvorovi su osnovne točke koje definiraju lokacije poput točaka interesa, zgrada ili drveća.

- **planet_osm_point**: Sadrži točkaste objekte koji predstavljaju specifične geografske točke, kao što su autobusne stanice, fontane, trgovine, itd.
- **planet_osm_polygon**: Sadrži sve poligonalne objekte iz OSM podataka. To uključuje zatvorene linije koje predstavljaju područja kao što su zgrade, parkovi, jezera i druge površine.
- **planet_osm_rels**: Pohranjuje sve relacije iz OSM podataka. Relacije su skupovi čvorova, putova i drugih relacija koji zajedno definiraju složenije geografske objekte ili logičke grupe.
- **planet_osm_roads**: Sadrži podatke specifično o cestama. To uključuje glavne i sporedne ceste, autoceste, staze i sve ostale vrste prometnica.
- **planet_osm_ways**: Pohranjuje sve puteve iz OSM podataka. Putevi su sekvence čvorova koje definiraju linijske objekte poput cesta i rijeka.

5.3. Programska implementacija rješenja

Programska implementacija je napisana u programskom jeziku Python koristeći alat Visual Studio Code. Implementacija sustava za usklađivanje traga vozila je implementirana u klasi `PostGisMapmatching`.

```
class PostGisMapmatching(MapmatchingAlgorithm):
def __init__(self):
    super().__init__()
    self.conn = psycopg2.connect(
        database=PostgreSql.DATABASE,
        user=PostgreSql.USER,
        password=PostgreSql.PASSWORD,
        host=PostgreSql.HOST,
        port=PostgreSql.PORT
        )
    self.conn.autocommit = True
    self.cur = self.conn.cursor()
```

```
self.SqlQueries = SqlQueries()
self.gpx = gpxpy.gpx.GPX()
.
.
```

U konstruktoru klase se uspostavlja konekcija sa instancom baze podataka. Parametri za uspostavu konekcije se nalaza u konfiguracijskoj datoteci. Konekciju sa bazom podataka omogućuje upravljački program (engl.*driver*) za PostgreSql `psycopg2`.

`Psycopg` je najpopularniji adapter za PostgreSQL bazu podataka za programski jezik Python. Njegove glavne značajke uključuju potpunu implementaciju Python DB API 2.0 specifikacije i sigurnost u više dretvi (više dretvi može dijeliti istu vezu). Dizajniran je za aplikacije s velikim brojem dretvi koje stvaraju i uništavaju mnogo kursora te izvode velik broj istovremenih "INSERT" ili "UPDATE" operacija [19]. Linija koda `self.conn.autocommit = True` omogućuje automatsko izvršavanje SQL naredbi bez potrebe za eksplicitnim pozivanjem metode `commit()`. Kada je autocommit postavljen na True, svaka SQL naredba koja se izvrši putem kursora automatski se potvrđuje i trajno pohranjuje u bazu podataka. Na ovaj način se izbjegava dodatni korak potvrđivanja transakcija. Linija koda `self.cur = self.conn.cursor()` kreira kursor za povezanu bazu podataka. Kursor je objekt koji omogućuje izvršavanje SQL naredbi i dohvaćanje rezultata iz baze podataka. Kroz kursor objekt se izvršavaju Sql upiti.

U kontruktoru se jos postavlja referenca na vlastitu klasu `SqlQueries`. U njoj su navedeni svi SQL upiti koji su korišteni u implementaciji. SQl upiti su navedeni u toj klasi radi bolje organizacije koda u odnosu na pisanje SQL upita direktno u `execute` naredbama.

Linija koda `self.gpx = gpxpy.gpx.GPX()` inicijalizira novi GPX objekt pomoću biblioteke `gpxpy`. To je jednostavna biblioteka za programski jezik Python za parsiranje i manipulaciju GPX datotekama. Korištenjem gpxpy.gpx.GPX() stvara se prazan GPX objekt koji se kasnije može popuniti podacima iz GPS zapisa. Ovaj objekt omogućuje rukovanje GPX podacima unutar programa, uključujući čitanje, pisanje i manipulaciju GPS podacima na strukturiran način.

```
def perform_mapmatching(self, file: str):
```

```
        self.create_unmatched_table()
        self.create_matched_point_table()
        gps_way_points = GPXParser(file).parse_file()
        for point, vehicle_id in gps_way_points:
            self.insert_unmatched_point(point)
            self.perform_postgis_map_matching()
            self.save_matched_point(vehicle_id)
            self.delete_all_data()
        self.delete_tables()
        self.write_gpx_file()
```

Usklađivanje traga vozila se izvodi u funkciji `perform_mapmatching(self, file: str)` ona kao parametar prima putanju do GPX filea sa tragom vozila koje treba uskladiti, a izlaz je stvorena GPX datoteka sa ispravljenom putanjom vozila.

Prvi korak je kreiranje tablica za neispravljene i ispravljane točke.

```
def create_unmatched_point_table(self):
    return """
        CREATE TABLE IF NOT EXISTS unmatched_points(
            id SERIAL PRIMARY KEY,
            point GEOMETRY(Point, 3857)
        );"""
```

Tablica `unmatched_points` u koju se postavljaju neispravljene točke se sastoji od primarnog ključa i neispravljenih podataka PostGIS tipa (engl.*Point*). Točka se sprema u koordinatnom referentnom sustavu SRID 3857. SRID 3857 označava korištenje projiciranog koordinatnog sustava poznatog kao Web Mercator, koji se često koristi u web kartografiji.

```
def create_matched_point_table(self):
    return """
        CREATE TABLE IF NOT EXISTS matched_points(
            id SERIAL PRIMARY KEY,
            point GEOMETRY(Point, 3857),
```

```
            name VARCHAR(255)
        );"""
```

Funkcija `create_matched_point_table(self)` kreira tablicu koja se sastoji od, primarnog ključa, ispravljene točke spremljene u referentnom sustavu SRID 3857, i imena segmenta na kojega će točka iz traga kretanja vozila biti namapirana.

```
gps_way_points = GPXParser(file).parse_file()
```

U prikazanoj liniji pomoću vlastite klase GPXParser parsiraju GPX datoteka i u funkciju vraća lista koja sadrži parsirane točke i identifikatore vozila za svaku točku iz ulazne GPX datoteke.

```
for point, vehicle_id in gps_way_points:
        self.insert_unmatched_point(point)
        self.perform_postgis_map_matching()
        self.save_matched_point(vehicle_id)
        self.delete_all_data()
```

Potom se za svaku točku iz ulazne datoteke izvodi proces usklađivanja kretanja.

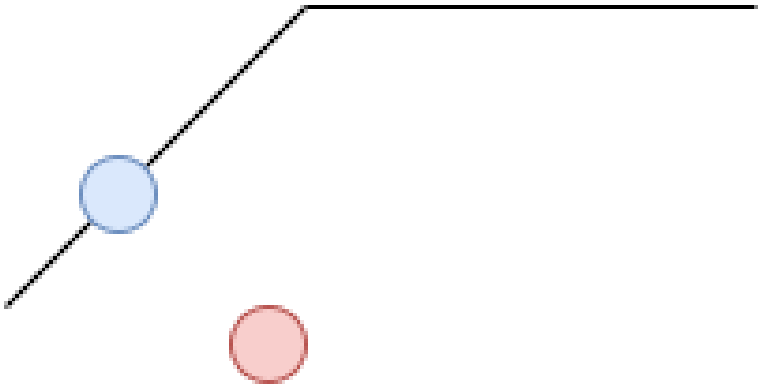
```
def insert_unmatched_point(self, point):
    return ("""
    INSERT INTO unmatched_points(point)
    VALUES(ST_Transform(ST_SetSRID(ST_MakePoint(%s, %s), 4326), 3857));
    """,(point.longitude, point.latitude))
```

U funkciji `insert_unmatched_point(self, point)` izvodi se proces učitavanja neusklađene točke u bazu podataka. SQL naredba za umetanje `INSERT INTO` umeće geometrijsku točku u tablicu `unmatched_points`. Funkcija `ST_SetSRID(ST_MakePoint(%s, %s), 4326)` stvara točku pomoću latitude i longitude vrijednosti koje su zadane kao parametri funkcije i postavlja njihov SRID na 4326, što je standardni referentni koordinatni sustav za GPS podatke. Transformacija se izvodi nad ulaznim točkama tako da bi se slagala sa referentnim sustavom koji smo postavili tijekom definiranja tablice.

```
def perform_map_matching(self):
    return """
        INSERT INTO matched_points(point, name)
        SELECT
            ST_ClosestPoint(road.way, unmatched.point) AS point,
            road.name AS name
        FROM
            unmatched_points AS unmatched,
            LATERAL (
                SELECT way, name
                FROM planet_osm_roads AS road
                WHERE ST_DWithin(road.way, unmatched.point, %s)
                ORDER BY way <-> unmatched.point
                LIMIT 1
        ) AS road
        """ % MAPMATCHING_RADIUS
```

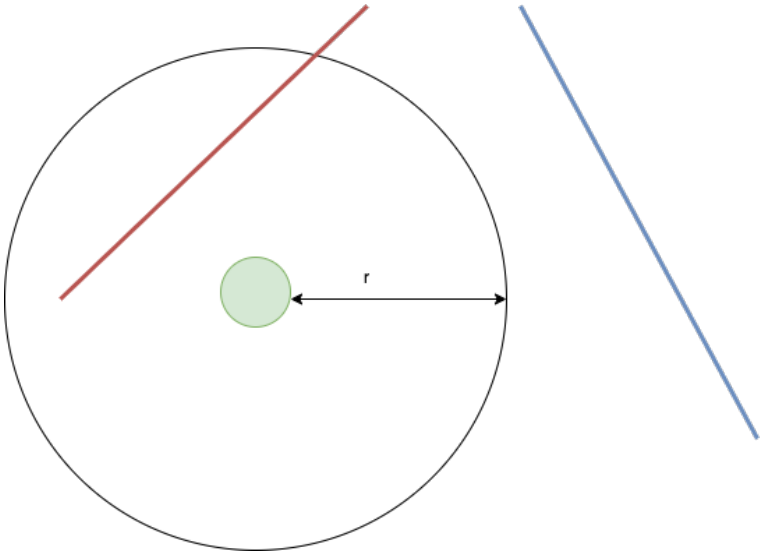
Funkcija `perform_map_matching(self)` vraća SQL upit koji se koristi za mapiranje točaka iz tablice `unmatched_points`, koje sadrže neispravljene točke iz ulazne datoteke koja sadrži trag kretanja vozila, na najbliže ceste iz tablice `planet_osm_roads` korištenjem funkcije `ST_Closestpoint()` . Upit umeće točku i ime ceste u tablicu `matched_points`. U podupitu, funkcija `ST_ClosestPoint(road.way, unmatched.point)` određuje najbližu točku na cesti `road.way` koja je najbliža neispravljenoj točki `unmatched.point`. Tablice `unmatched_points` i `planet_osm_roads` su povezane koristeći lateralni podupit. Lateralni podupit je podupit koji se pojavljuje u klauzuli FROM i koristi ključnu riječ LATERAL kako bi omogućio pristup stupcima prethodnih stavki u klauzuli FROM. Ovaj tip podupita omogućuje korištenje vrijednosti iz prethodnih redaka kao ulaza za podupit, što nije moguće s običnim podupitima[20]. U tom podupitu se za svaku neusklađenu točku bira najbliža cesta unutar zadane udaljenosti definirane konstantom **MAPMATCHING_RADIUS** koristeći funkciju **ST_DWithin**. Ta konstanta je u ovom slučaju postavljena na 50 metara. Podupit vraća geometrijske podatke ceste i ime ceste, koji se zatim koriste za ažuriranje tablice `matched_points`. Ceste su sortirane po udaljenosti do neusklađene točke koristeći operator `<->`, a bira se samo najbliža cesta putem

`LIMIT 1` klauzule.



Slika 5.7. PostGIS operator ST_ClosestPoint

Slika 5.7. prikazuje funkcioniranje PostGIS operator `ST_ClosestPoint(geometry geom1, geometry geom2)`. Operator vraća 2-dimenzionalnu točku na geometriji geom1 koja je najbliža geometriji geom2. To je prva točka najkraće linije između dviju geometrija. Kada su geometrije točka i linija, najbliža točka je sama točka. U slučaju kada su geometrije linija i točka, najbliža točka je točka na liniji [21]. Tako u ovom primjeru najbliža točka crvenoj točki je plava točka koja se nalazi na liniji.



Slika 5.8. PostGIS operator ST_Within

Slika 5.8. predstavlja funkcioniranje PostGIS operatora `ST_DWithin(geometry g1, geometry g2, double precision distance_of_srid)` Operator `ST_DWithin` u PostGIS-u koristi se za određivanje nalazi li se jedan geometrijski objekt unutar određene udaljenosti od drugog geometrijskog objekta. Na slici, zelena točka predstavlja geometrijski objekt čiju udaljenost od drugih objekata se želi provjeriti. Crvena i plava linija pred-

stavljaju ceste. Operator će vratiti TRUE ako je udaljenost između točke i najbliže točke na liniji manja od radijusa. Tako će se u ovom slučaju vratiti TRUE za crvenu liniju.

```
def save_matched_point(self, vehicle_id=None):
    self.cur.execute(self.SqlQueries.get_matched_point())
    point = self.cur.fetchone()
    GPXParser(OUTPUT_FILE)
    .append_point_to_gpx_file(point, self.gpx, vehicle_id)
```

Nakon operacije usklađivanja traga usklađena točka se nadodaje u izlaznu datoteku.

```
def delete_all_data(self):
    return """
        DELETE FROM unmatched_points;
        DELETE FROM matched_points;
        """
```

Funkcija `delete_all_data(self)` vraća SQL naredbu koja briše sve podatke iz tablica `unmatched_points` i `matched_points`. Ova naredba koristi dvije SQL naredbe `DELETE`, pri čemu prva briše sve redove iz tablice `unmatched_points`, a druga briše sve redove iz tablice `matched_points`. Funkcija se koristi za potpuno čišćenje podataka iz ovih tablica, uklanjajući sve prethodno unesene ili obrađene geometrijske točke. Ova operacija se koristi za brisanje podataka u privremenim tablicama između ispravljanja dviju točaka.

```
def delete_tables(self):
    self.cur.execute(self.SqlQueries.delete_matched_point_table())
    self.cur.execute(self.SqlQueries.delete_unmatched_point_table())
```

Nakon ispravljanja svih točaka iz ulazne datoteke, brišu se kreirane tablice pozivom funkcije `delete_tables`. Na kraju se ispravljene točke spremaju u izlaznu datoteku u GPX formatu.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx ...">
  <wpt lat="45.812116937263646" lon="16.19513007744935">
```

```
    <name>Dugoselska cesta</name>
    <extensions>
      <vehicleId>12345</vehicleId>
    </extensions>
  </wpt>
  <wpt lat="45.81199692145615" lon="16.195652180706677">
    <name>Dugoselska cesta</name>
    <extensions>
      <vehicleId>12345</vehicleId>
    </extensions>
  </wpt>
  .
  .
  .
</gpx>
```

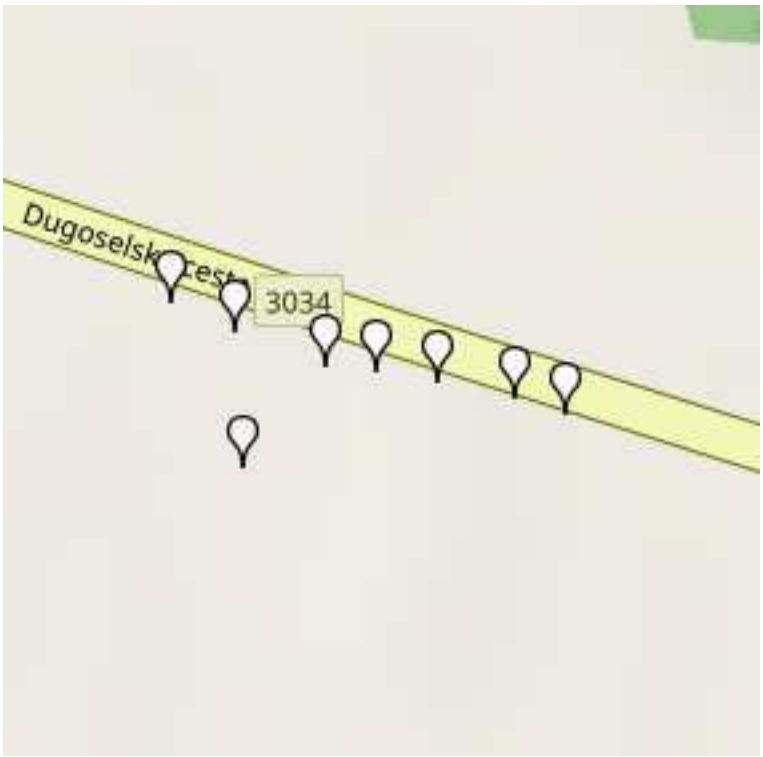
Prikazani isječak prikazuje izlazna datoteku u GPX formatu. Ona sadrži ispravljene točke te je veoma slična ulaznoj datoteci samo što se razlikuje po tome što izlazna datoteka ima i element `name` kojim se označuje naziv segmenta ceste pridružene svakoj točki.

Slika 5.9. Vizualizacija izlazne datoteke

Slika 5.9. prikazuje vizualizaciju izlazne GPX datoteke. Prikazuje ispravljene točke iz ulazne datoteke 5.4. Ulazne točke nakon ispravljanja se nalaze na cesti i imaju pridružen naziv segmenta.

5.4. Implementacija algoritma s povijesnim kontekstom

U klasi `PostgisMapmatchingWithHistory` je implementirana verzija algoritma za usklađivanje traga kretanja cestovnih vozila koja koristi povijesni kontekst u obliku zadnje usklađene točke. Razlika između obične implementacije i ove verzije je u tome da ako je točka udaljena od prošle usklađene točke za više od 50 metara onda se smatra da je došlo do nekog problema u ulaznim podatcima. Smatra se da je nova točka izolirani podatak (engl.*outlier*), te da predstavlja grešku u podatcima pa se takva točka mapira na zadnju poznatu ispravljenu točku.



Slika 5.10. Ulazni podatci sa izoliranim podatkom

Slika 5.10. prikazuje ulazne podatke za algoritam usklađivanja traga kretanja , na njoj se može vidjeti izolirani podatak koji je udaljen od ceste. Implementacija ove verzije se razlikuje od obične implementacije u nekoliko stvari.

```
for point, vehicle_id in gps_way_points:
```

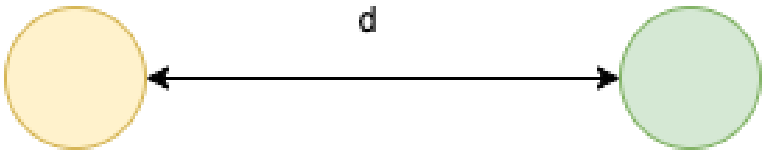
```
            if self.last_matched_point and
            self.is_distance_greater_than(point, self.last_matched_point):
                self.save_matched_point(vehicle_id, self.last_matched_point)
            else:
                self.insert_unmatched_point(point)
                self.perform_postgis_map_matching()
                matched_point = self.save_matched_point(vehicle_id)
                if matched_point:
                    self.last_matched_point = matched_point
            self.delete_all_data()
```

U funkciji za izvršavanje procesa usklađivanja traga provjerava postoji li već usklađena točka i zove funkcija za provjeravanje udaljenosti od nove točke i točke koja je već ranije bila usklađena, ako ona uopće postoji. Ako su uvjeti točni onda se sprema zadnja usklađena točka kao rezultat operacije. A ako je unutar radijusa onda se nastavlja izvedba programa kao i u prijašnjoj verziji.

```
def is_distance_greater_than(self, new_point, last_matched_point):
    self.cur.execute("""
        SELECT ST_Distance(
            ST_Transform(ST_SetSRID(ST_MakePoint(%s, %s), 4326), 3857),
            ST_Transform(ST_SetSRID(ST_MakePoint(%s, %s), 4326), 3857)
        ) > 50
    """,
    (
    new_point.longitude,
    new_point.latitude,
    last_matched_point[1],
    last_matched_point[0])
    )
    return self.cur.fetchone()[0]
```

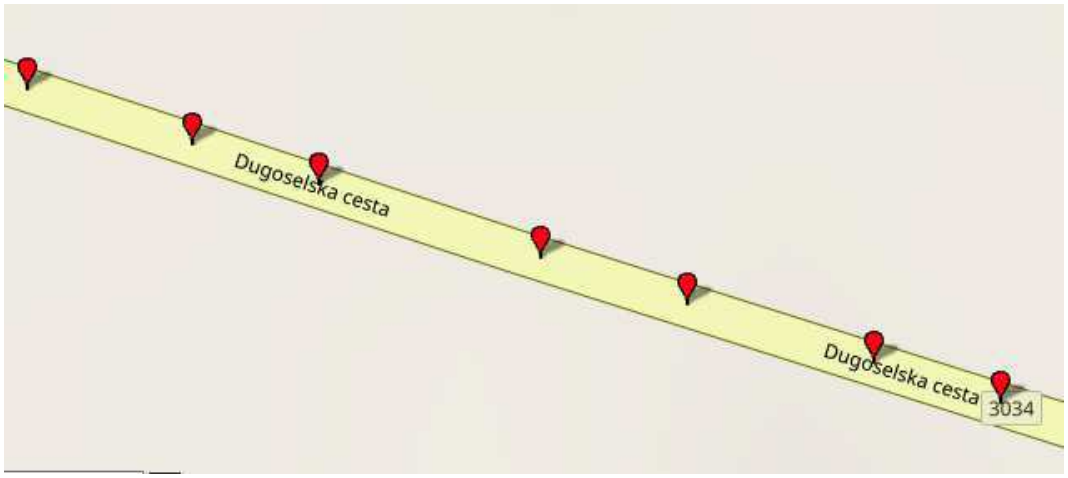
Funkcija `is_distance_greater_than` provjerava je li udaljenost između nove točke (`new_point`) i posljednje usklađene točke (`last_matched_point`) veća od 50 metara.

Prvo transformira koordinate obje točke iz SRID 4326 u SRID 3857 koristeći funkcije `ST_Transform` i `ST_SetSRID`. Zatim izračunava udaljenost između tih točaka koristeći `ST_Distance`. Ako je udaljenost veća od 50 metara, funkcija vraća `True`, inače vraća `False`.



Slika 5.11. PostGIS operator ST_Distance

Slika 5.11. prikazuje funkcioniranje operatora `ST_Distance(geometry g1, geometry g2)`. PostGIS funkcija `ST_Distance` koristi se za izračunavanje minimalne udaljenosti između dvije geometrije. Na primjer, kao što je prikazano na slici, funkcija može izračunati udaljenost između dvije točke, gdje su točke označene žutom i zelenom bojom. Ova funkcija vraća stvarnu udaljenost između središta dviju geometrija u jedinicama koordinatnog sustava u kojem se nalaze. Na slici, funkcija `ST_Distance` bi izračunala udaljenost između žute i zelene točke.



Slika 5.12. Vizualizacija izlazne datoteke

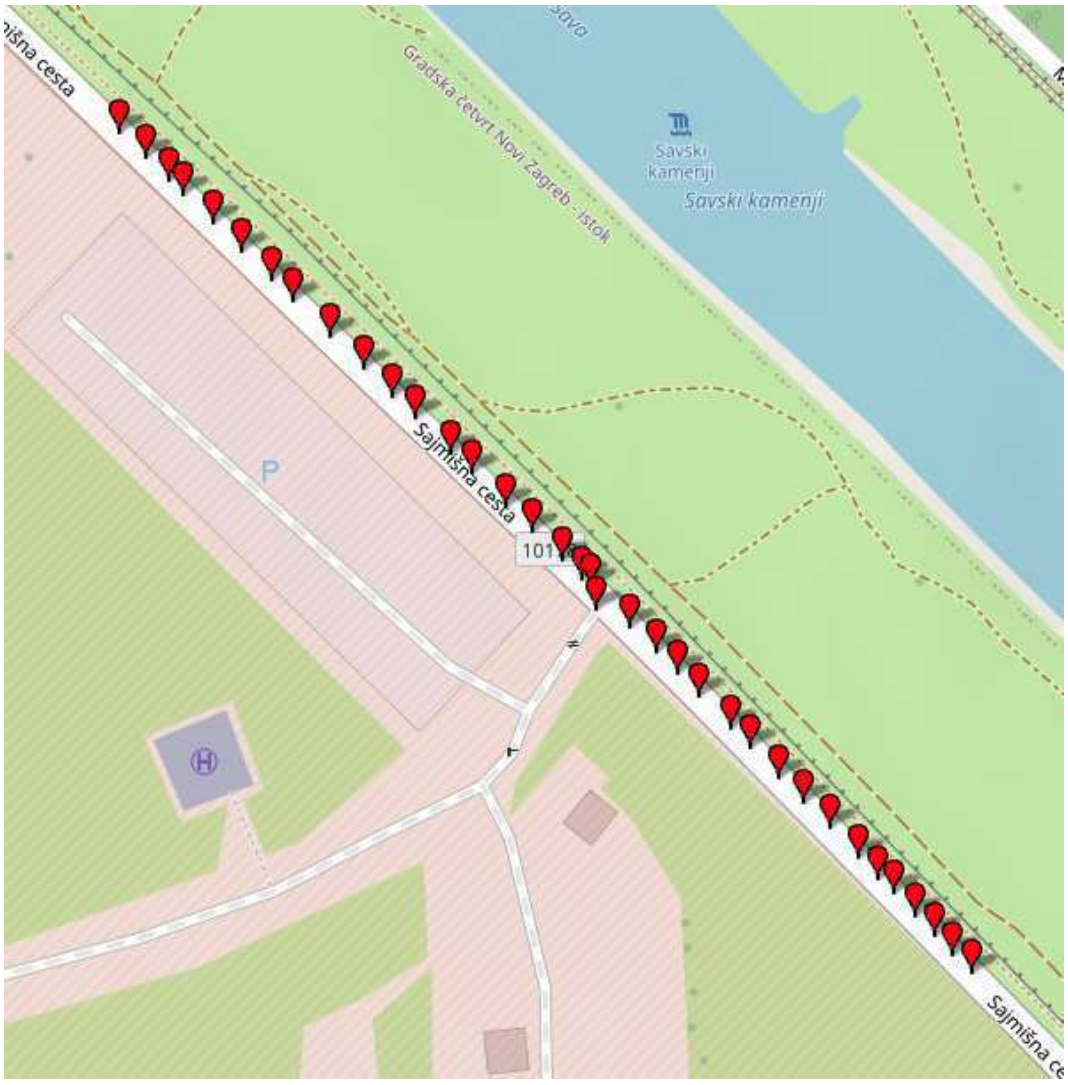
Slika 5.12. prikazuje vizualizaciju usklađenog traga kretanja. Vidi se da se točka koja je bila udaljena od ostalih mapirana u već ispravljenu točku od prije. Na taj način izolirane točke neće biti mapirane na krive segmente ceste.

6. Ispitivanje sustava

Ispitivanje sustava će se odvijati na učitanoj OpenStreetMaps karti grada Zagreba, u nekoliko različitih implementiranih slučajeva, te će biti izmjereno vrijeme izvođenja programa.

6.1. Jednostavna cesta

Slika 6.1. Ulazna datoteka



Slika 6.2. Usklađena izlazna datoteka algoritam bez konteksta

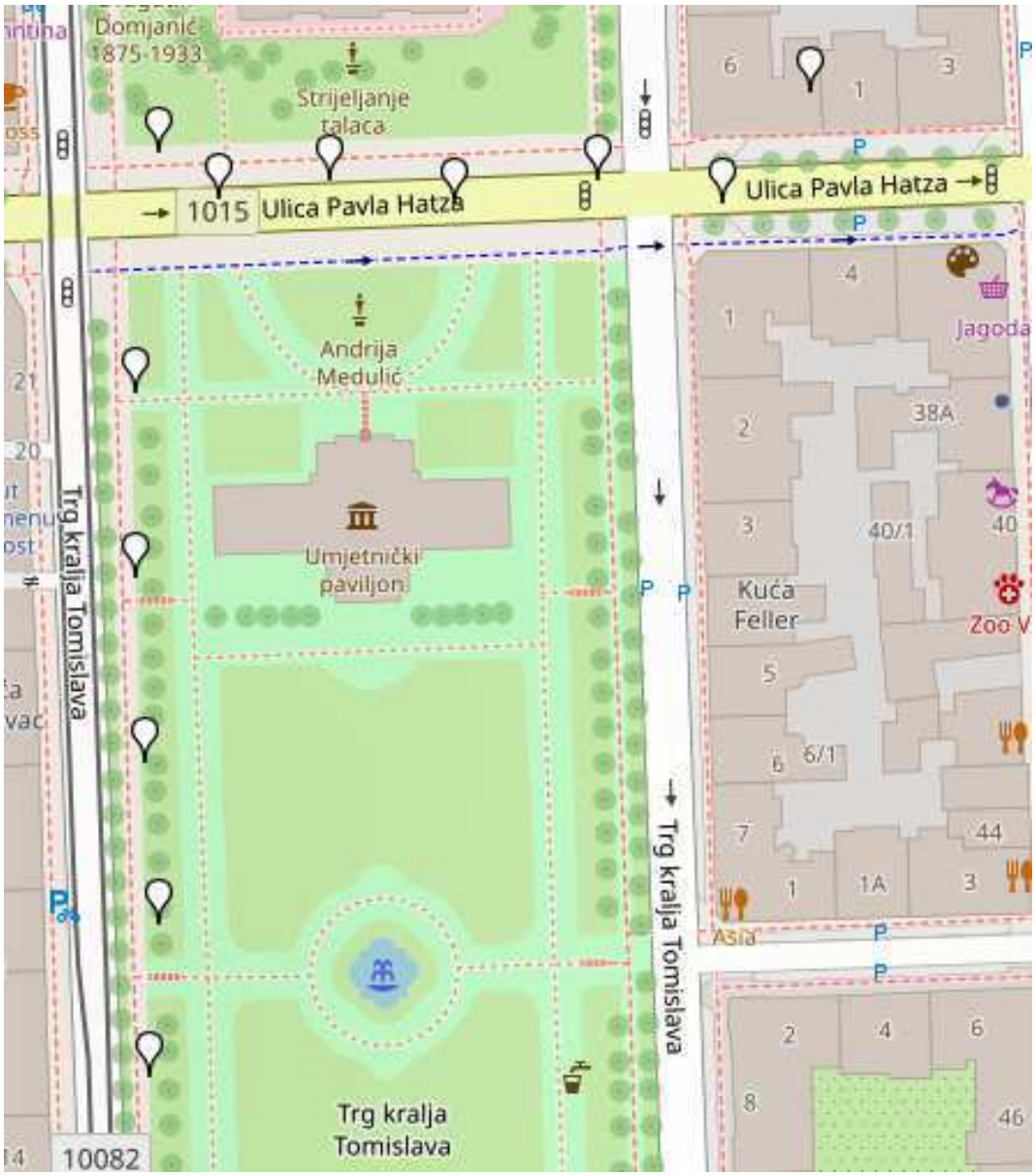
Slika 6.3. Usklađena izlazna datoteka za algoritam sa kontekstom

Slika 6.1. prikazuje kartu jednostavne ravne prometnice i neusklađene GPS točke prikazane bijelim markerima. Na slici 6.2. prikazuje vizualizirani rezultat izvođenja algoritma usklađivanja traga kretanja sa algoritmom bez konteksta, gdje su točke pozicionirane na cestu. Slika 6.3. prikazuje vizualizirani rezultat izvođenja algoritma usklađivanja traga kretanja s algoritmom s kontekstom gdje se može vidjeti da u odnosu na 6.2. točka koja je mnogo udaljenija od ceste nije mapirana na prometnicu. Tablica 6.1. prikazuje vremena izvođenja za podatke za jednostavne ceste.

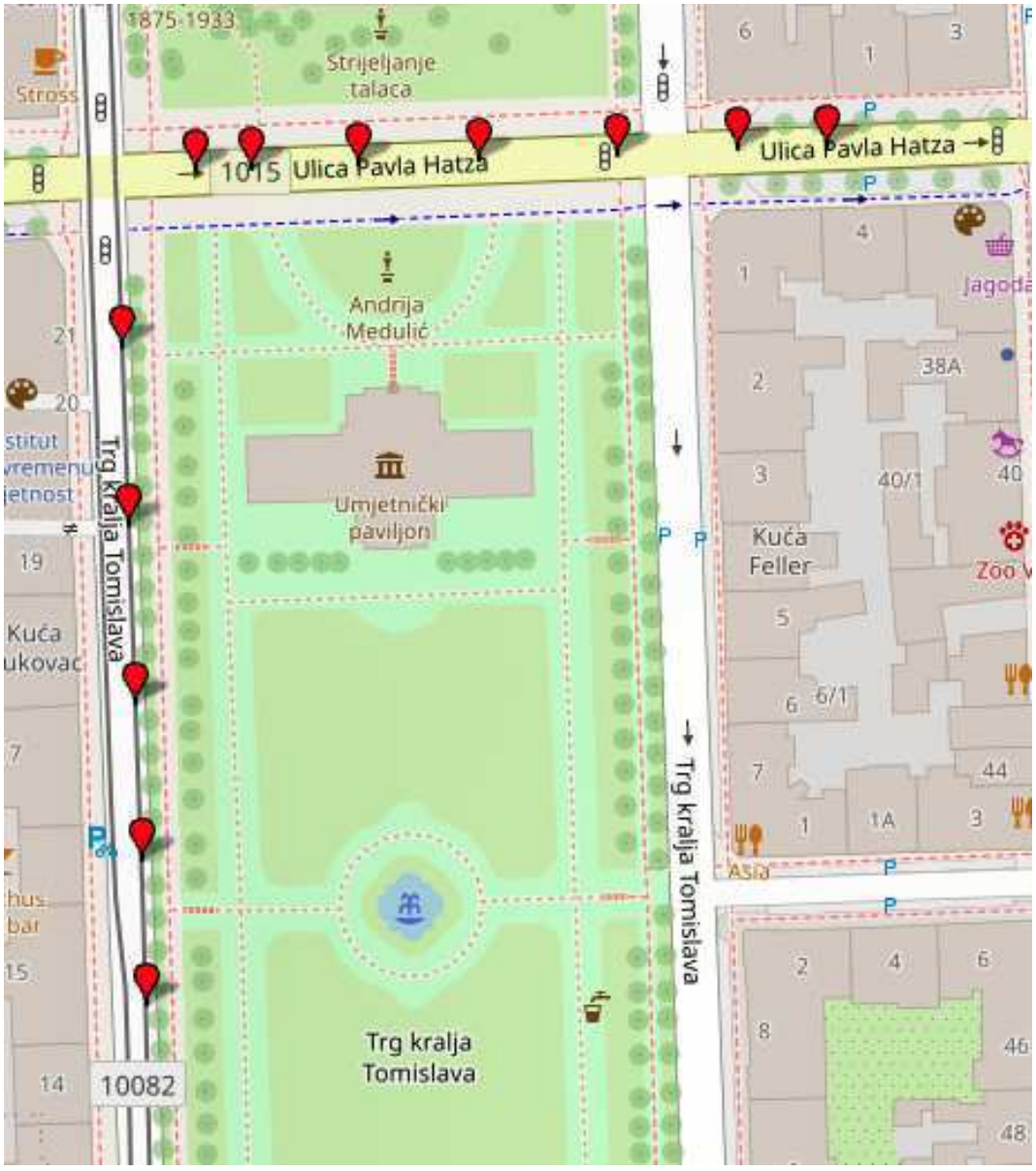
Tablica 6.1. Vrijeme izvođenja različitih algoritama

Algoritam	**Vrijeme izvođenja prosjek (s)**
Algoritam bez filtriranja	0.26
Algoritam sa filtriranjem	0.28
Sa kontekstom	0.3

6.2. Gusta gradska mreža i križanja



Slika 6.4. Ulazna datoteka



Slika 6.5. Usklađena izlazna datoteka algoritam bez konteksta

Slika 6.4. prikazuje kartu guste gradske mreže i neusklađene GPS točke prikazane bijelim markerima. Slika 6.5. prikazuje vizualizirani rezultat izvođenja algoritma usklađivanja traga kretanja s algoritmom bez konteksta, gdje su točke pozicionirane na najbliže ceste. Tablica 6.2. prikazuje vremena izvođenja za podatke s guste gradske mreže i križanja.

Tablica 6.2. Vrijeme izvođenja različitih algoritama

Algoritam	**Vrijeme izvođenja prosjek (s)**
Algoritam bez filtriranja	0.265
Algoritam sa filtriranjem	0.255
Sa kontekstom	0.27

6.3. Testiranje sustava s velikim brojem točaka

Kreirana je gpx datoteka koja sadžava 6000 točaka koja predstavlja ulaz za test sustava sa velikim brojem točaka. Tablica 6.3. prikazuje vremena izvođenja za podatke sa velikim brojem točaka.

Tablica 6.3. Vrijeme izvođenja različitih algoritama

Algoritam	**Vrijeme izvođenja prosjek (s)**
Algoritam bez filtriranja	7.29
Algoritam sa filtriranjem	7.54
Sa kontekstom	7.35

6.4. Zaključak testiranja

Ispitivanje sustava provedeno je na učitanim OpenStreetMap podacima grada Zagreba u različitim scenarijima. Testiranje je uključivalo jednostavne ceste, guste gradske mreže s križanjima te velike skupove podataka. Rezultati testiranja prikazani su kroz različite algoritme i njihovo vrijeme izvođenja.

U testiranju na jednostavnoj cesti, algoritam bez filtriranja pokazao je prosječno vrijeme izvođenja od 0.26 sekundi, dok je algoritam s filtriranjem imao nešto duže vrijeme

od 0.28 sekundi. Algoritam s kontekstom imao je najduže vrijeme izvođenja od 0.3 sekunde.

Za testiranje na gustom gradskom području, algoritam bez filtriranja imao je prosječno vrijeme izvođenja od 0.265 sekundi, algoritam s filtriranjem 0.255 sekundi, a algoritam s kontekstom 0.27 sekundi.

Također je testiran sustav s velikim brojem točaka, gdje je kreirana GPX datoteka s 6000 točaka. Algoritam bez filtriranja imao je prosječno vrijeme izvođenja od 7.29 sekundi, algoritam s filtriranjem 7.54 sekundi, a algoritam s kontekstom 7.35 sekundi.

Iz rezultata je vidljivo da algoritmi s dodatnim filtriranjem i kontekstom zahtijevaju nešto više vremena za obradu, ali ne pokazuju nekakva velika odstupanja pa se ne može sa sigurnošću reći da im izvođenje traje duže. Ukupno gledano, sustav se pokazao funkcionalnim i efikasnim za različite scenarije usklađivanja traga kretanja vozila.

7. Zaključak

U okviru ovog rada izrađen je sustav za usklađivanje traga kretanja cestovnih vozila korištenjem PostgreSQL baze podataka s proširenjem PostGIS. Opisani su ključni koncepti vezani uz geoprostorne baze podataka, algoritme za usklađivanje traga kretanja te implementacija sustava. Korišten je oblikovni obrazac strategija za modularnost i fleksibilnost algoritma. Testiranjem sustava potvrđena je njegova funkcionalnost i izmjerena je brzina rada u različitim scenarijima.

Literatura

[1] "What are the main challenges and solutions in modeling?" https://www.linkedin.com/advice/0/what-main-challenges-solutions-modeling, 2024., pristup: svibanj 2024.

[2] "Gps performance and accuracy", https://www.gps.gov/systems/gps/performance/accuracy/, pristup: svibanj 2024.

[3] M. A. Quddus, W. Y. Ochieng, i R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions", *Transportation Research Part C: Emerging Technologies*, sv. 15, br. 5, str. 312–328, 2007. https://doi.org/https://doi.org/10.1016/j.trc.2007.05.002

[4] P. Chao, Y. Xu, W. Hua, i X. Zhou, "A survey on map-matching algorithms", *Journal of Navigation*, sv. 73, br. 1, str. 45–61, 2020., pristup: lipanj 2024. https://doi.org/10.1017/S0373463319000808

[5] P. Rigaux, M. Scholl, i A. Voisard, *Spatial Databases: With Application to GIS*. San Francisco: Morgan Kaufmann Publishers, 2002.

[6] Oracle. (2024) What is a geospatial database? Pristup: lipanj 2024. [Mrežno]. Adresa: https://www.oracle.com/autonomous-database/what-is-geospatial-database/

[7] Open Geospatial Consortium, "Open geospatial consortium (ogc)", http://opengeospatial.org/, pristup: lipanj 2024.

[8] PostGIS, *PostGIS - R-Tree Spatial Indexing*, 2008., pristup: lipanj 2024. [Mrežno]. Adresa: https://postgis.net/docs/support/rtree.pdf

[9] ——, “Postgis introduction workshop - indexing”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://postgis.net/workshops/postgis-intro/indexing.html

[10] ——, “Postgis introduction workshop - loading data”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://postgis.net/workshops/postgis-intro/loading_data.html#loading-data

[11] R. Guru, “What is a design pattern?” 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://refactoring.guru/design-patterns/what-is-pattern

[12] ——, “Strategy design pattern”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://refactoring.guru/design-patterns/strategy

[13] Topografix, “Gpx specification”, 2024., acPristup: lipanj 2024. [Mrežno]. Adresa: https://www.topografix.com/gpx.asp

[14] G. Visualizer, “Gps visualizer: Free online map creation”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://www.gpsvisualizer.com/map?output_home

[15] OpenStreetMap, “About openstreetmap”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap

[16] ——, “Beginners’ guide 1.3”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://wiki.openstreetmap.org/wiki/Beginners_Guide_1.3

[17] ——, “Openstreetmap: 45.8088/15.9801”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://www.openstreetmap.org/#map=15/45.8088/15.9801

[18] *osm2pgsql Manual*, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://osm2pgsql.org/doc/manual.html#running-osm2pgsql

[19] Psycopg, “Psycopg2: Postgresql database adapter for python”, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://pypi.org/project/psycopg2/

[20] PostgreSQL Global Development Group, *PostgreSQL Documentation: Table Expressions*, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://www.postgresql.org/docs/current/queries-table-expressions.html

[21] PostGIS Development Team, *PostGIS Documentation: ST_ClosestPoint*, 2024., pristup: lipanj 2024. [Mrežno]. Adresa: https://postgis.net/docs/ST_ClosestPoint.html

Sažetak

Usklađivanje traga kretanja cestovnih vozila primjenom geoprostornih operatora u bazi podataka PostGIS

Matej Krehula

Tema ovog diplomskog rada je razvoj sustava za usklađivanje traga kretanja vozila korištenjem PostgreSQL baze podataka s proširenjem PostGIS. Sustav integrira različite verzije algoritma za usklađivanje traga vozila. Sustav obrađuje i analizira GPS podatke iz GPX datoteka te koristi OpenStreetMap podatke kao izvor podataka. Implementacija je detaljno prikazana u Pythonu, demonstrirajući korištenje knjižnica i upita prema bazi podataka. Testiranje je pokazalo učinkovitost sustava u različitim scenarijima, omogućujući precizno praćenje vozila.

Ključne riječi: Usklađivanje traga, PostgreSQL, PostGIS, praćenje vozila, OpenStreetMap

Abstract

Map Matching of Road Vehicles using PostGIS Geospatial Operators

Matej Krehula

The topic of this thesis is the development of a vehicle trajectory matching system using a PostgreSQL database with a PostGIS extension. The system integrates different versions of vehicle trajectory matching algorithms. It processes and analyzes GPS data from GPX files and uses OpenStreetMap data as the data source. The implementation is detailed in Python, demonstrating the use of libraries and database queries. Testing has shown the system's effectiveness in various scenarios, enabling precise vehicle tracking

Keywords: Map-matching, PostgreSQL, PostGIS, Vehicle Tracking, OpenStreetMap