SVEUČILIŠTE U ZAGREBU
GEODETSKI FAKULTET

Ivan Buljan

IZRADA ZIDNE POLITIČKE KARTE SVIJETA ZA 2021. GODINU IZ PODATAKA *OPENSTREETMAPA*

Diplomski rad

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
GEODETSKI FAKULTET

Na temelju članka 19. Etičkog kodeksa Sveučilišta u Zagrebu i Odluke br. 1_349_11 Fakultetskog vijeća Geodetskog fakulteta Sveučilišta u Zagrebu, od 26.10.2017. godine (klasa: 643-03/16-07/03), uređena je obaveza davanja „Izjave o izvornosti“ diplomskog rada koji se vrednuju na diplomskom studiju geodezije i geoinformatike, a u svrhu potvrđivanja da je rad izvorni rezultat rada studenata te da taj rad ne sadržava druge izvore osim onih koji su u njima navedeni.

IZJAVLJUJEM

Ja, **Ivan Buljan**, (JMBAG: 0007069729), rođen dana 20.02.1988. u Dubrovniku, izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi tog rada nisam koristio drugim izvorima osim onih koji su u njemu navedeni.

U Zagrebu, dana 17. rujna 2021.

_____

Potpis studenta

I. AUTOR	
Ime i prezime:	Ivan Buljan
Datum i mjesto rođenja:	20. veljače 1988., Dubrovnik, Republika Hrvatska
II. DIPLOMSKI RAD	
Naslov:	Izrada zidne političke karte svijeta za 2021. godinu iz podataka *OpenStreetMapa*
Broj stranica:	104
Broj tablica:	0
Broj slika:	30
Broj bibliografskih podataka:	13 + 78 URL-a
Ustanova i mjesto gdje je rad izrađen:	Geodetski fakultet Sveučilišta u Zagrebu
Mentor:	doc. dr. sc. Ana Kuveždić Divjak
Komentor:	
Voditelj:	
III. OCJENA I OBRANA	
Datum zadavanja teme:	8.12.2020.
Datum obrane rada:	17.09.2021.
Sastav povjerenstva pred kojim je branjen diplomski rad:	doc. dr. sc. Ana Kuveždić Divjak
	doc. dr. sc. Ivka Kljajić
	dr. sc. Marina Viličić

Izrada zidne političke karte svijeta za 2021. godinu iz podataka OpenStreetMapa

Sažetak**:** *U diplomskom radu izrađena je zidna politička karta svijeta u mjerilu 1:20 000 000 koristeći podatke OpenStreetMapa za 2021. godinu. Za izradu je korišten softver osmišljen 2016. godine na Geodetskom fakultetu Sveučilišta u Zagrebu pod nazivom OSMPoliticalMap. Testirana je funkcionalnost softvera te su neki njegovi dijelovi ažurirani i unaprijeđeni kako bi radili u novim verzijama potrebnih programa. Nakon početnog zadavanja parametara softver automatiziranim postupkom preuzima podatke sa servisa, kreira slojeve i učitava ih na kartu. Stilovi slojeva i pozicije naziva objekata učitani su iz referentne karte koja je također dio softvera. Poslije automatiziranog procesa ručno su provedene manje korekcije i karta je pripremljena za ispis. Takav postupak visoko automatizirane izrade karte, koristeći prostorne podatke s nekog servisa, naziva se servisno orijentirana kartografija.*

Ključne riječi: *automatizacija u kartografiji, OpenStreetMap, politička karta svijeta, otvoreni prostorni podaci, servisno orijentirana kartografija*

The creation of the wall political map of the world from the OpenStreetMap data

Abstract**:** *The task of this thesis was the automated creation of a wall political map of the world at a scale of 1:20 000 000, using OpenStreetMap data for the year 2021. The software used to create the map was developed at the Faculty of Geodesy, University of Zagreb in 2016 under the name OSMPoliticalMap. The software has been tested and some parts have been updated and improved to work in new versions of the required programs. After the initial setting of the parameters, the software automatically retrieves data from the service, creates layers and loads them on the map. Layer styles and object label positions are loaded from the reference map which is also part of the software. After the automated process, minor corrections were made manually and the map was prepared for printing. This process of highly automated map creation, using spatial data from a service, is called service-oriented mapping.*

Keywords: *automated cartography, OpenStreetMap, political map of the world, service-oriented mapping, open spatial data*

SADRŽAJ

1. UVOD

Prema definiciji Međunarodnoga kartografskog društva (engl. *International Cartographic Association*, ICA) karta je simbolizirani prikaz geografske stvarnosti koja predstavlja odabrana obilježja ili karakteristike, nastaje stvaralačkim autorskim izborom i dizajnirana je za uporabu kada su prostorni odnosi od primarne važnosti (URL 1).

U digitalnoj eri karte su sveprisutne u našim životima. Nekad je to u obliku interaktivnog sučelja, a nekad ih nismo ni svjesni jer su u obliku preciznih geoprostornih informacija koje u pozadini utječu na naše svakodnevne uređaje, poput pametnog telefona ili automobila. Pristup kartama se na neki način počinje smatrati osnovnim ljudskim pravom koje se ostvaruje u zamjenu za dobrovoljne geoinformacije (engl. *Volunteered Geographic Information*). Koncepti obrade geoprostornih podataka, stvaranje geoprostornih informacija i prijenos geoprostornog znanja koriste se servisno orijentiranim arhitekturama (engl. *service-oriented architectures*). Trendovi pokazuju da se metodologija „prikupljanje - sastavljanje - sortiranje“ mijenja u metodologiju “ponovna upotreba - sastavljanje - dijeljenje“ koristeći decentraliziran pristup uslugama i izvorima prostornih podataka (Jobst i Gartner, 2019).

Tome je zasigurno doprinjela dostupnost i otvorenost servisa za pohranu sirovih prostornih podataka poput *OpenStreetMapa* (OSM) (URL 2). Podacima s OSM-a moguće je pristupiti u svakom trenutku i iz tih podataka izraditi svoju kartu. Jedan takav projekt pod nazivom „OSMPoliticalMap“ već je rađen 2016. godine na Geodetskom fakultetu Sveučilišta u Zagrebu. Nije bila riječ samo o izradi karte, već se proces probao maksimalno automatizirati, pa možemo reći da je to primjer servisno orijentirane kartografije. Postupak i softver objavljeni su pod otvorenom licencom u digitalnom repozitoriju Github (URL 3) i slobodno su dostupni za preuzimanje. Koristeći navedeni softver trebalo bi biti moguće izraditi vlastitu političku kartu svijeta u proizvoljnom mjerilu.

Predmet ovog rada je provesti testiranje funkcionalnosti softvera i postupka, te po potrebi provesti ažuriranje i unaprijeđenje.

1.1 UOČENI PROBLEMI

S obzirom na vremenski odmak od pet godina, stigle su nove verzije svih korištenih softvera, a postoji mogućnost da se i sintaksa pojedinih naredbi izmjenila. Odlučeno je da će se cijeli postupak ispitati i po potrebi ažurirati tako da bude funkcionalan u najnovijim verzijama potrebnih softvera.

Postupak izrade i probleme s kojima se pritom susreo, opisao je svom radu Jogun (2016), a iz njega je jasno da je najviše bilo problema s podacima *OSM*-a. Ukupno gledajući najveći problem je upitna kvaliteta podataka, odnosno njihova heterogenost. Drugi problem je količina podataka koja se u prošlih pet godina umnogostručila, pa treba vidjeti mogu li softveri pratiti taj porast.

Nadalje, kod Joguna (2016) je opis postupka bio algoritamski orijentiran s pripadajućim dijagramima toka što nije loše za potencijalne naprednije korisnike, međutim za ostale bi takav opis mogao biti nedovoljan, pa će se u ovom radu postupak izrade karte promatrati i opisati s tehničkog aspekta odnosno na razini samog kôda softvera.

1.2 CILJEVI I ZADACI

Određeni su ciljevi i zadaci ovog rada:

C1 – Izraditi zidnu političku kartu svijeta za 2021. godinu iz podataka *OpenStreetMapa* u mjerilu 1:20 000 000.

C2 – Testirati funkcionalnost softvera i postupka te provesti ažuriranje i unaprijeđenje.

C3 – Analizirati kvalitetu podataka s osvrtom na podatke dobivene 2016. godine i dati nove preporuke za poboljšanje podataka.

1.3 HIPOTEZE

U skladu s određenim ciljevima i zadacima, a na temelju problematike ovog rada, postavljaju se polazne hipoteze:

H1 – Moguće je, u skladu s kartografskim načelima, visoko automatiziranim postupkom izraditi zidnu političku kartu svijeta za 2021. godinu iz podataka *OpenStreetMapa* u mjerilu 1:20 000 000.

H2 – Potrebno je provesti postupka ažuriranja i unaprijeđenja postojećeg softvera.

H3 – Moguće je pojednostavniti postupak zbog bolje kvaltete *OpenStreetMap* podataka.

1.4 ZNANSTVENO-STRUČNI DOPRINOS

Očekuje se da će ovaj rad preispitati funkcionalnost softvera postojećeg servisno orijentiranog postupka izrade cjelovite političke karte svijeta iz sirovih podataka *OpenStreetMapa*. Izvorni doprinos je provedba ažuriranja i unaprijeđenja u vidu izrade novih softverskih dijelova za nefunkcionalne djelove postupka izrade karte, kao i opsežno tumačenje postupka izrade karte.

2. TEORIJSKA OSNOVICA

2.1 POLITIČKA KARTA SVIJETA

2.1.1 Definicije, sadržaj i primjeri oblikovanja političkih karata svijeta

Političke karte spadaju pod tematske karte iz područja ljudske djelatnosti. Prema Međunarodnom kartografskom društvu (ICA) službena definicija tematskih karata iz 1995. godine glasi: „Tematske karte su kartografski prikazi najrazličitijih tema iz prirodnog i društvenog (gospodarskog, socijalnog i kulturnog) područja, koje su neposredno vezane za prostor“. Na toj vrsti karata jedan ili više topografskih objekata, poput naselja, prometnica, reljefa, vode, vegetacije i područja, posebno su kartografikom istaknuti i prikazani s posebnom važnošću (Frangeš, 2019).

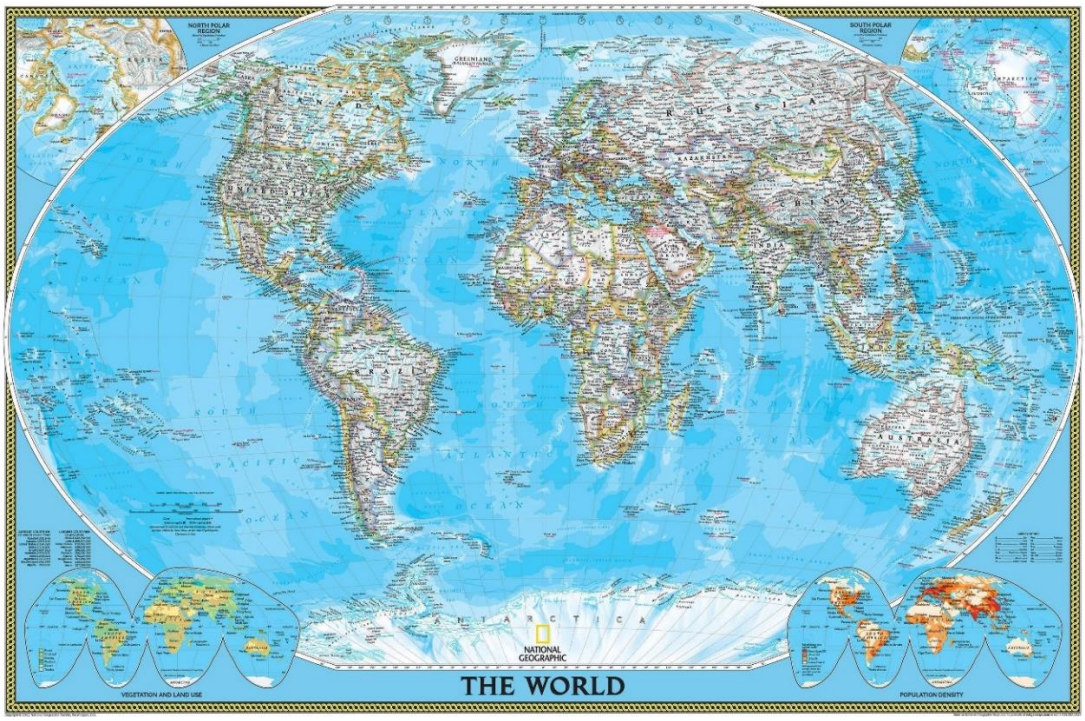
Kad tematski kartografski prikaz sadrži informacije o položaju i rasprostranjenju administrativnih područja riječ je o političkoj karti. Postoji nekoliko definicija političke karte.

Prema terminološkoj bazi hrvatskog strukovnog nazivlja (Struna) politička karta je tematska karta na kojoj je prikazana državna ili administrativna podjela i na kojoj su pojedine administrativne jedinice različito obojene (URL 4).

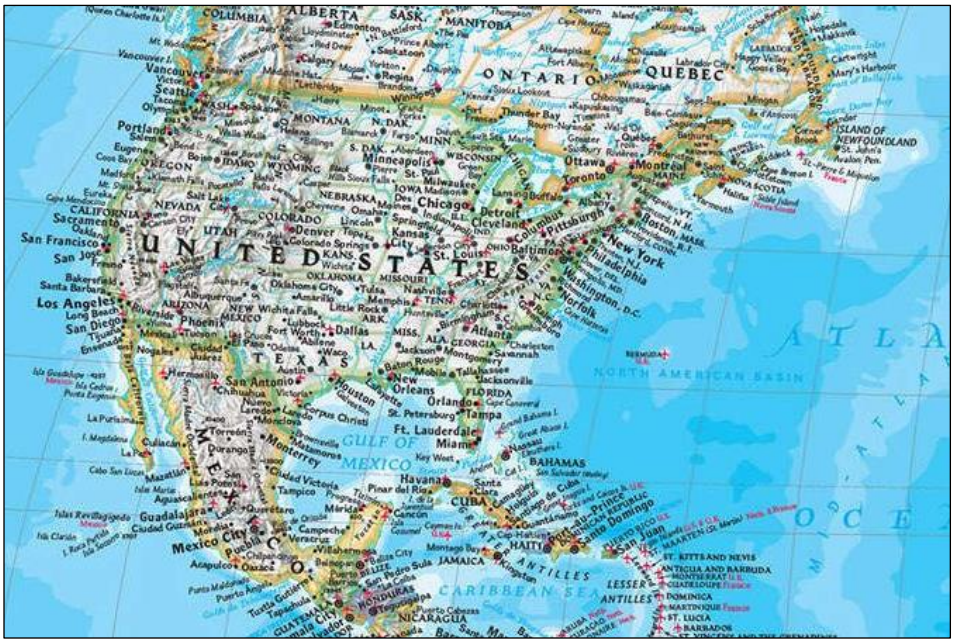
Još jednu jednostavnu definiciju političke karte dao je Neumannu (1997) u svom enciklopedijskom rječniku kartografije. Ona kaže da politička karta predstavlja političke ili administrativne jedinice, obično korištenjem različitog bojenja ili sjenčanja njihovih površina.

Eckert (1925) političkim kartama naziva one karte koje prikazuju razmještaj država. Mogu se smatrati i povijesnim kartama sadašnjosti, a trebale bi odražavati stvarno stanje. Nadalje, Eckert je mišljenja kako, što se tiče sadržaja, nema smisla da se na kartama razdvajaju fizički elementi od političkih jer nalikuju jedni drugima i međusobno se nadopunjuju. Međutim, moć percepcije učenika se smanjuje ako se politički sadržaj preklopi s fizičkim. Zbog didaktičkih razloga i zbog potrebe za rasterećenjem prikaza na karti razdvajanje fizičkog i političkog sadržaja može se pronaći na mnogim školskim kartama. Kod takvih karata Eckert zagovara plošno bojanje entiteta umjesto vinjetiranja granica jer smatra da će učenici tako bolje zapamtiti njihov položaj i oblik.

Zbog toga što u geoznanstvenim istraživanjima imaju svrhu preglednog prikaza područja, Raisz (1948) povezuje političke i opće karte. Na političkim kartama prikazani su nazivi administrativno-teritorijalnih jedinica, njihove granice i koordinatna mreža, ali kada se na to dodaju elementi poput gradova, rijeka, cesta i željezničkih pruga onda je to opća karta. Takva praksa dodavanja elemenata opće karte na političke je prisutna i na nekim renomiranim kartama svijeta poput one „World Classic Map“ časopisa *National Geographic* (URL 5). Na njoj je, uz sadržaj uobičajen za političke karte, dodan uistinu velik broj drugih fizičko-geografskih elemenata: rijeke, jezera, gradovi, napomene uz sporne granice, magnetni polovi zemlje, sjenčani reljef, oznake za međunarodne zračne luke, debljina leda na Grenlandu, vremenske zone, morske struje, itd. Zanimljiva je informacija da je sjenčani reljef više od 75 godina istaknut na zidnim kartama *National Geographica*. Ukupni dojam ponajviše kvari količina toponima kojima je cijela karta ugusto prekrivena, pa preglednost ove karte nije idealna. S aspekta političkih karata učinjeno je dosta kompromisa i zbog svega nabrojanog s pravom se može postaviti pitanje je li ipak riječ o općoj karti svijeta.



Slika 2.1. Karta „World Classic Map“ magazina National Geographic (URL 6)



Slika 2.2. Detalj karte „World Classic Map“ magazina National Geographic (URL 7)

U kartografskoj zajednici cijenjena je karta „World Political Map“ autora Toma Pattersona koja je slobodna za preuzimanje i korištenje, uključujući i izmjenu sadržaja. Dostupna je na *web* stranici *Shaded Relief* (URL 8) i to u tri projekcije.

Slika 2.3. Karta „World Political Map“ u Natural Earth 2 projekciji (URL 9)

Slika 2.4. Detalj karte „World Political Map“ (URL 10)

Kao što je vidljivo na priloženim slikama, i ova karta spada među političke karte koje imaju dodane elemente karakteristične za opće karte. Za razliku od gore spomenute „World Classic Map“ karte časopisa *National Geographic*, ovdje je to odrađeno poštujući kartografska načela, pa je broj dodatnih fizičko-geografskih elemenata i toponima primjeren tipu i namjeni karte. Osim glavnih gradova prikazani su samo oni s najvećim brojem stanovnika. Tako je kod manjih država prikazan samo glavni grad. Sjenčani reljef nije dominantan i ne opterećuje ukupni izgled karte. Zadržane su samo najveće rijeke sa pripadajućim imenima. Manja jezera su prikazana, ali bez nazivlja.

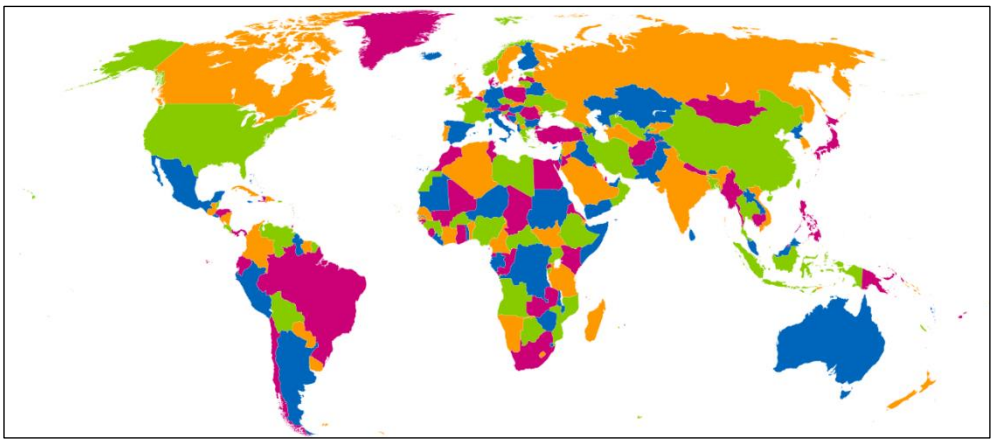
Među političkim kartama koje su bliže Neumannovoj definiciji najpoznatija je ona američke obavještajne agencije (engl. *Central Intelligence Agency* – CIA) naziva „Political Map of the World“. Na njoj su prikazani političko-teritorijalni entiteti, glavni gradovi država i saveznih država, gradovi s najvećim brojem stanovnika i najveća jezera. Osim nabrojanim elemenatima, dodijeljeni su nazivi i oceanima, većim morima, najvećim zaljevima, otocima odn. otočjima. Nisu prikazane ni rijeke ni reljef, pa se posebno dobro ističu političko-teritorijalni entiteti što i je osnovna svrha političkih karti.

Slika 2.5. Karta „Political Map of the World“ obaviještajne agencije CIA (URL 11)

Slika 2.6. Detalj CIA-ine karte „Political Map of the World“ (URL 12)

Prepoznatljivo obilježje političkih karti svakako su obojani političko-teritorijalni entiteti, pa je pri izradi takve karte potrebno posebnu pozornost posvetiti bojama i metodi bojanja. Raisz (1962) smatra da bi se na političkim kartama različite države trebale bojati u svim mogućim tonovima, a trebaju biti birane tako da susjedne države imaju različitu boju. Iako nije nužno korištenje različitih boja za sve države, jasno je da se varijacijama može postići mali broj ponavljanja boja ako im se mijenja ton, zasićenost i intenzitet. Kad je riječ o bojanju političkih karata ne možemo ne spomenuti tzv. teorem o četiri boje (engl. *Four Color*

Theorem) koji je upravo zbog primjene u kartografiji poznat i pod engleskim nazivom *Four Color Map Theorem*. Teorem u svom najjednostavnijem obliku kaže da nije potrebno više od četiri boje za bojenje regija bilo koje karte, a da pritom dva susjedna područja ne budu iste boje (URL 13). Teorem je i eksperimentalno potvrđen, a na slici Slika 2.7. je jednostavan primjer karte svijeta obojene s četiri boje bez susjednih zemalja iste boje.



Slika 2.7. Ilustracija teorema o četiri boje na političkoj karti svijeta (URL 14)

Ranije u tekstu je spomenuto, a na slikama se moglo uočiti, da postoje dva načina bojanja političko-teritorijalnih entiteta kod političkih karata. Jedan način je plošno bojanje čitave površine svakog pojedinog entiteta u jednu boju, a drugi način je vinjetiranje granice entiteta jednom bojom. Političke karte kod kojih su političko-teritorijalni entiteti obojani plošno i u intezivnim bojama (kao npr. na slici Slika 2.7.) izgledaju privlačno, ali nisu uvijek najbolje rješenje. Bolja metoda je bojenje uskog pojasa uz granicu tj. vinjetiranje (Raisz 1962).

Primjer plošnog bojanja je CIA-ina karta „Political Map of the World“ gdje su entiteti u zagasitim bojama, a granice između njih bijele. Vinjetiranje granica je korišteno u karti „World Classic Map“ časopisa National Geographic, dok je na karti „World Political Map“ Toma Pattersona korišteno vinjetiranje granica, ali i bojanje cijelog entiteta u istu boju sa smanjenim intezitetom. U sva tri slučaja je način bojanja izvrsno prilagođen ostalim sadržajima koji su prikazani na karti. Na CIA-inoj karti su prikazani samo gradovi i veća jezera, dok na druge dvije karte imamo dodatni sadržaj. Na karti časopisa National Geographic vinjetiranje granica je u vizualnom smislu ostavilo mjesta za rijeke i sjenčani reljef, a to što preostala površina pojedinog entiteta nije u boji je pojačalo kontrast i donekle pomoglo s čitljivošću prenagomilanih crnih imena gradova i država i plavih imena rijeka i

jezera. Karta Toma Pattersona ima mnogo manje takvog dodatnog sadržaja i manje izražen sjenčani reljef, pa iako su cijeli entiteti obojani, sve izgleda odmjereno, skladno i pregledno.

Nakon svega može se steći dojam da je u današnjoj kartografiji pomalo nejasna granica između političkih i općih karata. Iako sadržaj koji se prikazuje na političkim kartama nije strogo zadan, iz analiziranih primjera i uvažavajući definicije s početka ovog poglavlja možemo donijeti sljedeće zaključke:

- Prikaz političko-teritorijalnih entiteta različitim bojama je obavezan.
- Prikaz većih jezera je uobičajen.
- Prikaz glavnih gradova je uobičajen.
- Prikaz većih gradova je uobičajen.
- Prikaz većih rijeka je poželjan.
- Prikaz reljefa je neobavezan.

Najvažnije svojstvo karte, mogućnost neposrednog promatranja i studiranja geografskog objekta bez obzira kako taj objekt bio velik po pružanju ili površini, temelji se na dvije posebnosti kartografskog prikaza – na umanjenju objekata koje istražujemo i na njihovom prikazu u generaliziranom obliku (Frančula, 2003).

2.1.2 Mjerila korištena na kartama svijeta

Mjerilo je neizostavan element svake karte jer prikazuje odnos elemenata duljine luka u projekciji prema podudarnoj duljini na elipsoidu ili kugli. Danas se u oblikovanju tematskih karata umjesto brojčanog obvezno na kartu stavlja grafičko mjerilo jer ono pri povećavanju i smanjivanju karte zadržava svoju funkciju. Mjerna letvica treba biti jednostavna, jer tematske karte rjeđe imaju svrhu mjerenja dužina i površina i ne smije odvlačiti pozornost od osnovnog sadržaja tematske karte (Frangeš, 2019). Kod političkih karti svijeta broj i izbor jezera, većih gradova i rijeka izravno ovisi o mjerilu karte jer je mjerilo presudan faktor koji određuje količinu tematskih informacija (Frangeš, 2019). Zbog toga što je potrebno prikazati područje čitave Zemlje na relativno maloj površini zidne karte, mjerila zidnih karti svijeta su poprilično sitna. Tako je karta časopisa National Geographic izrađena u mjerilu 1:31 900 000, dok su karte Toma Pattersona i CIA-a u mjerilu 1:35 000 000.

2.1.3 Generalizacija na kartama svijeta

Druga posebnosti kartografskog prikaza je generalizacija. Sama riječ dolazi od latinske riječi *generalis* (hrv. opći) i znači uopćavanje. Prema tome kartografska generalizacija je uopćavanje sadržaja karte prilagođeno mjerilu i (ili) svrsi karte. Bit generalizacije je izbor i svrsishodno uopćavanje onog najvažnijeg, onog bitnog. Generalizacija nam pomaže da, s obzirom na namjenu karte, tematiku i mjerilo, na karti prikažemo stvarnost u njenim najvažnijim tipičnim osobinama. Postupak generalizacije izvodi kartograf u procesu izrade karte. Kartografska generalizacija obuhvaća sljedeće postupke: izbor, pojednostavljivanje, sažimanje, povećavanje (naglašavanje), pomicanje, pretvorbu metode prikaza. Izbor je najvažniji postupak generalizacije jer se u njemu odlučuje hoće li neki objekt biti prikazan na karti. Izbor se može provoditi prema minimalnim veličinama, na osnovi broja objekata i prema važnosti objekata. Kada sadržaj karte imamo u digitalnom obliku, tada je izbor prema minimalnim veličinama lako provesti (Frančula, 2003). Osim postupka izbora, kod političkih karata cijelog svijeta bitni su i ostali postupci. Tako je potrebno provesti pojednostavljivanje složenih linijskih objekata poput obalnih crta i tokova rijeka. Povećavanje (naglašavanje) će se koristiti kod manjih otočnih država koje bi u kombinaciji sitnog mjerila i izbora prema minimalnim veličinama bile izostavljene s karte.

Oni kartografi koji su dio svog radnog vijeka proveli u dobu ručnih (manualnih) postupaka, a drugi u digitalnoj eri, primijetili su da današnje karte nisu generalizirane kao što su nekad bile. Automatizirani kartografi zamijenili su neprirodno glatke linije ručne kartografije linijama koje su preopterećene detaljima. Izrada karti sitnog mjerila iz podataka krupnog mjerila je korijen ovog problema. Objašnjenje nedostataka prošlih i sadašnjih postupaka generalizacije zavise o dodatnom radu koji je potrebno uložiti. Dodavanje pojedinosti ručnoj karti zahtijeva mukotrpno praćenje na bliskoj vizualnoj udaljenosti, zbog čega su često pretjerano generalizirane karte iz manualnog vremena. Nasuprot tome, uklanjanje informacija dobivenih iz baza podataka koje koristimo pri izradi karte nije toliko težak posao, ali zahtijeva dodatni napor što objašnjava pretrpane karte s viškom detalja koje viđamo danas. Još jedan razlog neprovođenja generalizacije kod današnjih karata je oklijevanje u mijenjanju tuđih podataka, osobito ako su te podatke stvorili znanstvenici. Loše geografsko poznavanje područja koje se kartira može dodatno pogoršati ovu tendenciju (Patterson, 2010).

Od tri karte koje smo analizirali kroz ovo poglavlje, samo je karta Toma Pattersona imala zadovoljavajuće provedenu generalizaciju (Slika 2.8.). *CIA*-ina karta nema dobro provedenu generalizaciju obalne crte, dok karta časopisa *National Geographic* ima solidno provedenu generalizaciju obalne crte i rijeka pojednostavljivanjem, ali je izbor gradova proveden ne poštivajući načela kartografskog prikaza. Kartografski prikaz bilo kojih objekata, pojava i stanja mora biti dovoljno točan, cjelovit, ispravan, čitljiv i zoran. Stoga nema smisla prikazivati objekte koji korisniku karte neće biti čitljivi, samo za volju točnosti ili potpunosti. Kvaliteta i vrijednost karte će se povećati ako se generalizacijom najvažniji pojedinačni objekti učine čitljivim (Frančula, 2003).

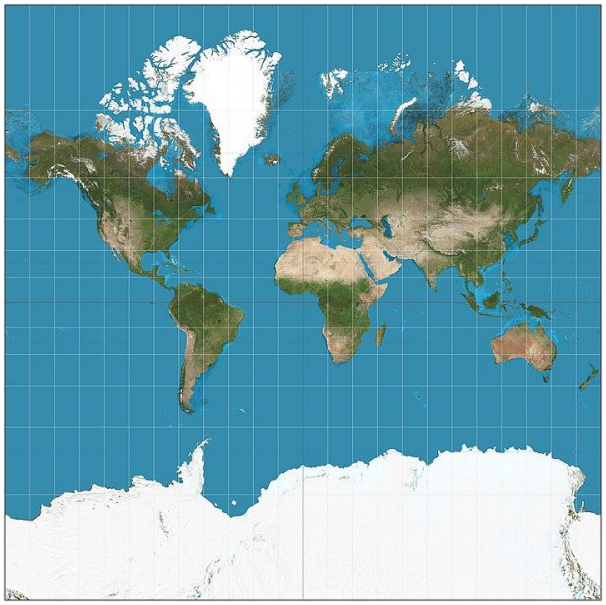
Slika 2.8. Usporedba generalizacije obalne crte na analiziranim kartama. Slijeva: World Classic Map (National Geographic), World Political Map (T. Patterson), Political Map of the World (CIA)

2.1.4 Kartografske projekcije prikladne za karte svijeta

U digitalnoj eri sveprisutne su karte svijeta u nekoj varijanti Mercatorove projekcije koje se lijepo prikazuju na četvrtastim ekranima elektroničkih uređaja poput računala i pametnih telefona. Koriste ju gotovo svi glavni davatelji internetskih karata, uključujući *Google Maps, Mapbox, Bing Maps, OpenStreetMap, Mapquest, Esri*, i mnogi drugi. To je *de facto* standard u *web* kartiranju.

Mercatorova projekcija spada u skupinu uspravnih konformnih cilindričnih projekcija u kojoj se glavni pravci podudaraju s meridijanima i paralelama, a ime je dobila po njenom pronalazaču i slavnom kartografu, Gerhardu Kremeru Mercatoru. Iz slike karte svijeta u Mercatorovoj projekciji (Slika 2.9.) vidljivo je da deformacija nema na ekvatoru, no udaljavanjem od ekvatora deformacije dužina i površina uočljivo rastu. Već na paraleli sa širinom $\varphi = \pm 60°$ deformacije površina iznose 300%. Na kartama svijeta sastavljenim u cilindričnim projekcijama dobro se prikazuju vremenske zone, a zbog svoje konformnosti ova projekcija je pogodna za prikaz vjetrova i morskih struja. U ostalim slučajevima

uspravne cilindrične projekcije nemaju čestu primjenu za kartiranje cijelog svijeta, prije svega jer se polarna područja kod tih projekcija prikazuju vrlo deformirano.



Slika 2.9. Mercatorova projekcija, područje preslikavanja ograničeno je paralelama sa širinom φ=±85° (URL 15)

Zbog velikih i lako uočljivih deformacija površina Mercatorova projekcija nije pogodna za izradu općegeografskih i političkih karata svijeta. Stoga se za karte svijeta koriste projekcije u kojima se pol preslikava kao linija. Od pseudocilindričnih ekvivalentnih projekcija s polom linijom preporučljivo je koristiti dvije Eckertove projekcije - eliptičnu (Eckertova IV) i sinusoidalnu (Eckertova VI), i sinusoidalnu projekciju prof. Kavrajskog. Među pseudocilindričnim uvjetnim projekcijama preporučuje se koristiti eliptičnu projekciju prof. Kavrajskog i Eckertovu V (sinusoidalnu) projekciju. Ekvivalentna Hammer-Aitovljeva i uvjetna Aitovljeva projekcija spadaju među one projekcije s polnom točkom i krivolinijskim paralelama koje se koriste koriste za karte svijeta.

Ipak, najveću primjenu kod izrade karata svijeta imaju projekcije s krivolinijskim meridijanima i paralelama i s polnom linijom. U tim projekcijama srednje kvadratne deformacije na čitavom području preslikavanja manje su nego u cilindričnim i pseudocilindričnim projekcijama te projekcijama s polnom točkom i krivolinijskim paralelama. Duljina linije pola prema duljini ekvatora treba se odnositi u omjeru od 1:2 do 1:3,5. Provedena ispitivanja su pokazala kako su srednje kvadratne deformacije manje u uvjetnim projekcijama nego u ekvivalentnim i konformnim projekcijama.

Iz te skupine projekcija najviše se za izradu karata svijeta u zapadnim zemljama upotrebljava Winkelova (trostruka) projekcija. Karakteristike te projekcije su blago zakrivljene linije paralela i vrlo povoljan raspored deformacija površina i kutova između ekvatora i polarnih područja zbog čega je Winkelova projekcija s pravom našla široku primjenu u izradi karata svijeta.

Oswald Winkel predložio je ovu projekciju za izradu karata svijeta 1913. godine, a dobije se kao aritmetička sredina između Aitovljeve i uspravne ekvidistantne cilindrične projekcije. Iako je u svom originalnom dizajnu Winkel koristio $\varphi_0=50°28'$ za standardnu paralelu preslikavanja, sa $\varphi_0=40°$ se dobije povoljan raspored deformacija površina i kutova, gdje je φ_0 širina paralele koja se preslikava bez deformacija. Upravo ova projekcija se često koristi za karte svijeta. Aitovljeva projekcija je dobivena modifikacijom poprečne azimutalne ekvidistantne projekcije, pa proizlazi da je Winkelova projekcija izvedena iz tri projekcije zbog čega i nosi naziv trostruka i spada u grupu uvjetnih projekcija (Frančula, 2004).

Od tri karte koje su analizirane kroz ovo poglavlje, karta „World Classic Map“ u izdanju *National Geographica* izrađena je u Winkelovoj trostrukoj projekciji s nultim meridijanom u sredini. Ovu projekciju za svoje karte *National Geographic Society* koristi od 1998. godine kad se odustalo od dotad korištene Robinsonove projekcije iz razloga što Winkelova projekcija smanjuje izobličenje kopnenih masa koje su bliže polovima. Karta „World Political Map“ Toma Pattersona koristi vlastitu projekciju Natural Earth II. Riječ je o novoj pseudocilindričnoj projekciji izrazito zaobljenog oblika. Na većim geografskim širinama meridijani se savijaju strmo prema kratkoj polnoj liniji. U usporedbi s drugim zaobljenim projekcijama, poput Aitovljevei. Natural Earth II manje se ispupčuje na strane, pa tako veću površinu papira ispunjava kartom. Kontinenti izgledaju poznato bez protezanja u smjeru sjever-jug kao npr. kod eliptične Eckertove IV projekcije. Natural Earth II ima slične vrijednosti deformacije površina kao kod Robinsonove i Winkelove projekcije (URL 16). Karta „Political Map of the World“ američke obavještajne agencije *CIA* koristi Robinsonovu projekciju. Riječ je pseudocilindričnoj projekciji kod koje se meridijani preslikavaju kao krivulje simetrične prema izabranom srednjem meridijanu koji se preslikava kao pravac, a paralele se preslikavaju kao međusobno paralelni pravci okomiti na srednji meridijan. Na ovoj karti nije poznata informacija o srednjem meridijanu koji se preslikava kao pravac, ali

je vidljivo da to nije meridijan koji prolazi kroz Greenwich tzv. *nulti meridijan.* Pretpostavka na temelju slobodne procjene je da je riječ o meridijanu na 10° istočne geografske duljine. To je napravljeno s dobrim razlogom. Budući da se ovdje radi o političkoj karti svijeta, a Rusija površinom prelazi 180. meridijan i proteže se do približno 170° zapadne geografske duljine, sačuvana je cjelovitost u prikazu političko-teritorijalnih entiteta koji su na političkoj karti najvažniji element (Slika 2.5.).

2.2 SERVISNO ORIJENTIRANA KARTOGRAFIJA

Osim dobrog dizajna i upotrebljivosti, za kvalitetnu kartu su potrebne pomno odabrane i pripremljene informacije. Glavni izvor podataka u servisno orijentiranoj kartografiji je infrastruktura prostornih podataka koju karakterizira pružanje velike količine izvora podataka različitih kvaliteta, struktura i pokrivenosti.

Servisno orijentirana kartografija se može opisati kao proizvodnja karata s distribuiranim izvorima, podacima prikupljenim u stvarnom vremenu i vrlo visokim stupnjem automatizacije. Takav način kartiranja nudi stvaranje karata i složene analize geoinformacija korištenjem paradigme servisno orijentirane arhitekture (engl. *Service-Oriented Architecture*, SOA), koja pokriva specifične karakteristike informacijske tehnologije u decentraliziranoj mreži. Decentralizirane mreže, koje također mogu sadržavati centralizirane komponente, omogućuju pristupu raznovrsnijim izvorima podataka, podržavaju fleksibilne tokove izrade i povećavaju potrebe pohrane, kao i procesnu snagu ako se uspostave odgovarajući standardi, sučelja i veze.

Zbog sve većeg porasta zbirki podataka i raznolikosti izvora podataka, potrebno je korištenje automatizacije u obradi geoinformacija. Ona slijedi aspekte integracije podataka, interpretacije podataka, „semantifikacije“ podataka (poput dodavanja višejezičnog i kontekstualnog značenja i važnosti podacima) i pripreme prijenosa podataka u smislu učinkovite komunikacije za onoga tko će te informacije obrađivati. Servisno oriejntirana arhitektura je pritom glavna paradigma za povezivanje i istraživanje svega.

Koristeći prethodno spomenuto inženjerstvo prostornih podataka, prostornu analitiku i integraciju podataka, servisno orijentirana kartografija ovisi o decentraliziranim i slabo povezanim izvorima podataka i usluga. Ova karakteristika rezultira različitim nadležnostima

podataka od strane davatelja podataka. Te su nadležnosti spojene u tehnološku tematsku mrežu, mrežu infrastrukture prostornih podataka. Nadležnosti podataka opisuju relevantnost proizvođača podataka koji je odgovoran za skup podataka i njegovu kvalitetu. Slično tome, nadležnost servisa opisuje značaj funkcionalnih komponenti pružateljskog mjesta koji je odgovoran da je funkcionalnost softverski definirane infrastrukture (SDI) dostupna kad god je to potrebno. Funkcionalnost uključuje pristup podacima, transformacije, procesne korake i tako dalje. Sve nadležnosti su autonomne unutar konceptualne mreže, što znači da se mogu izmijeniti, proširivati ili razmjenjivati bez destabilizacije mreže.

Infrastruktura prostornih podataka slijedi pristup s razinama, što znači da se podaci, uključujući metapodatke i registre, nalaze na razini podataka. Servisi i općenite funkcionalnosti su smještene na razini logike. Aplikacije kao što su klijenti karata (engl. *map clients*) smještene su na razini prezentacije. Ova arhitektura s više razina stvara neovisnost između podataka, funkcionalnosti i prezentacije, što znači da se promjene na jednoj razini mogu izvršiti bez utjecaja na druge razine. Sve tri razine uključene su u izradu karata. Zbog toga je ovaj koncept IT arhitekture najrelevantniji za dizajn produkcije karata i upravljanje geoinformacijama (Jobst i Gartner, 2019).

Servisno orijentirana kartografija jedna je od glavnih paradigmi za ugrađivanje velikih setova podataka i distribuiranih izvora u modernu izradu karata, bez posjedovanja izvora tih podataka. Kako bi bila stabilna i pouzdana, ova arhitektura zahtijeva posebne okvire (engl. *frameworks*), alate i postupke. Osim tehnoloških struktura, organizacijski aspekti, kao i aspekti lanca opskrbe i sposobnosti GIS-a, pružaju moćne alate za uspješno upravljanje suvremenim geoinformacijama. Tehnološka perspektiva moderne izrade karata uključuje IT arhitekte, programere, dizajnere, stručnjake za geoinformacije i kartografe. Struke i njihovo znanje akumuliraju se za potrebe prikupljanja prostornih podataka, rudarenja prostornih podataka, stvaranja i obrade prostornih informacija, kao i za geovizualizaciju (URL 17).

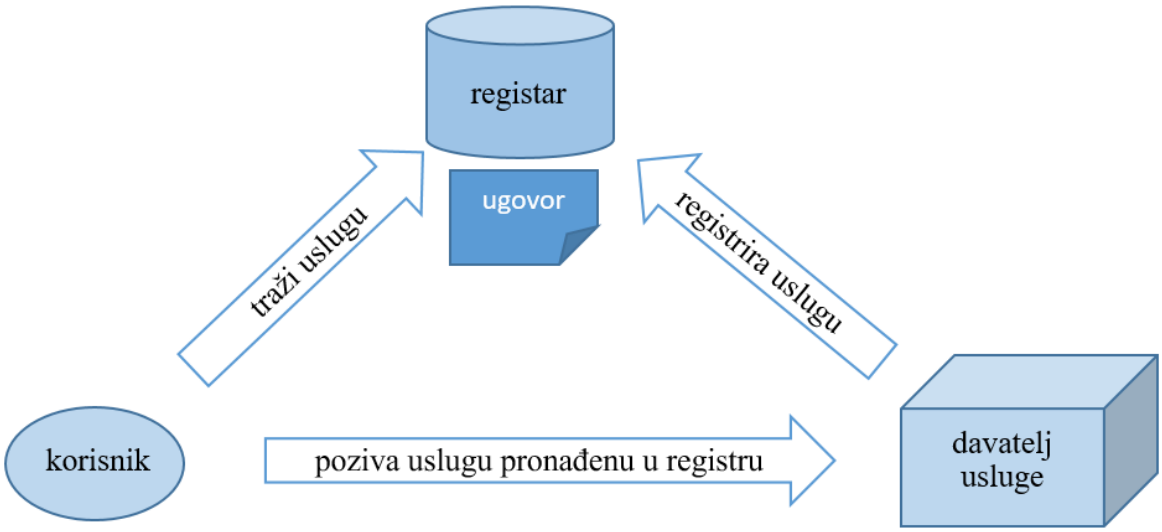
U implementaciji izrade servisno orijentiranih karata, obvezno je poštivati zakone o autorskim pravima i ne činiti prekršaje. Vizualizacijski aspekti se bave grafičkim kartografskim prikazom i dizajnom web prezentacije. Korisnički aspekti usmjereni su na prilagođavanje korisniku. Izrada servisno orijentiranih karata omogućuje rezultate na temelju zahtjeva korisnika koji se definiraju prema namjeni za koju se karta treba koristiti.

To je povezano s psihološkim aspektima, koji imaju važnu ulogu u izboru boja, specifičnom načinu prezentacije i prilagodbi lokalnim običajima (URL 18).

2.3 SERVISNO ORIJENTIRANA ARHITEKTURA (SOA)

Pojam „servisno orijentirana arhitektura“ prva je počela koristiti američka tvrtka Gartner (URL 19). Riječ je o arhitekturi koja se zasniva na ideji da davatelji usluga registriraju svoju uslugu u javnom registru, a ovaj registar potrošači koriste za pronalaženje usluga koje odgovaraju određenim kriterijima. Ukoliko registar ima takvu uslugu, potrošaču će pružiti ugovor i adresu krajnje točke za tu uslugu (URL 20). Drugim riječima, jedan sustav izloži određen skup vlastitih funkcionalnosti, a drugi sustavi koriste te funkcionalnosti. Osnovna namjena takve arhitekture je razmjena informacija između dva i više računala ili između računala i korisnika. Softver, odnosno softverski model u kojem se aplikacija sastoji od softverskih servisa i korisnika tih servisa (softverskih klijenata) dizajniran je prema „klijent – poslužitelj“ modelu. Razlika od standardnog „klijent – poslužitelj“ modela je u nastojanju da razdvoji softverske komponente te u korištenju različitih načina pristupa do tih servisa po prinicipu modularnosti i niske međuovisnosti servisa (engl. *low coupling*) što za rezultat ima visoku autonomiju samih servisa (engl. *high cohesion*). Kao što je već rečeno, aplikacije su reorganizirane u grupe funkcionalnosti nazvane servisi, a servisi su zapravo aplikacije izložene preko standardiziranog sučelja i kao takve dostupne su i razumljive ostalim sustavima u okruženju.

Prednosti servisno orijentirane arhitekture su pojednostavljenje ponovne iskoristivosti elemenata informatičkog sustava, povećana brzina i produktivnost razvoja jer su elementi jednom izloženi preko servisa dostupni drugim aplikacijama na korištenje, jednostavnije proširivanje sustava jer se uglavnom sastoji u dodavanju novih servisa, velika modularnost koja omogućuje jednostavniju promjenu funkcionalnosti i lakše održavanje sustava (URL 21).

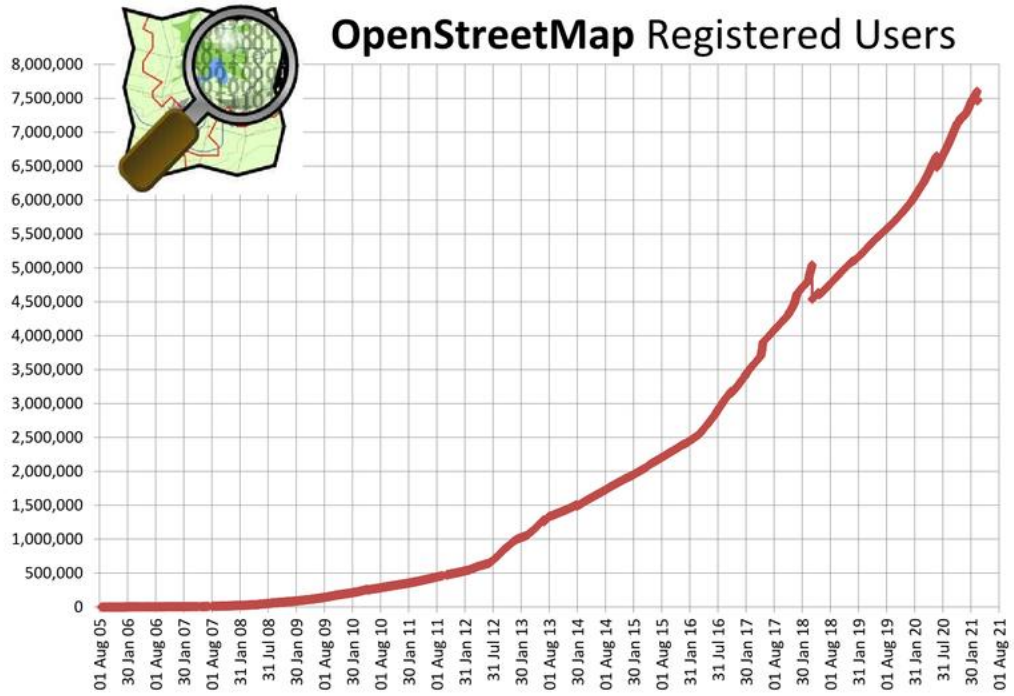


Slika 2.10. Osnovni koncept servisno orijentirane arhitekture

2.4 OPENSTREETMAP

OpenStreetMap (OSM) je projekt u kojem sudjeluju milijuni ljudi diljem svijeta, uglavnom laika bez stručnih znanja s područja kartografije i geoinformatike. Glavna značajka projekta je besplatna karta čitavog svijeta s mogućnošću uređivanja koju volonteri grade uglavnom od nule i objavljuju je s licencom otvorenog sadržaja (engl. *open-content license*). OpenStreetMap licenca dopušta besplatan (ili gotovo besplatan) pristup samoj karti, ali i svim temeljnim podacima te karte (URL 22).

Prema službenoj statistici, u trenutku pisanja ovog rada bilo je 8 037 603 registriranih korisnika koji su učitali 8 856 296 252 GPS točaka, 7 181 763 302 *čvorova*, 798 556 633 *puteva* i 9 232 745 *relacija* (URL 23). Takva statistika čini *OpenStreetMap* najizrazitijim primjerom volonterskih geoinformacija (engl. *Volunteered Geographic Information* – VGI).



Slika 2.11. Prikaz porasta broja registriranih OpenStreetMap korisnika (URL 24)

2.4.1 Razvoj OSM-a

Geografski podaci (prostorni podaci) nisu besplatni u mnogim dijelovima svijeta gdje je zadatak kartiranja i izdavanja karata dan raznim državnim agencijama koje zauzvrat mogu zaraditi prodajom tih podataka, što znači da se iz poreza plaćaju kartiranja i izmjere, a zatim je potrebno ponovno platiti da bi se dobila kopija neke karte umjesto da ih se slobodno i široko distribuira (URL 25).

Velika većina dostupnih *web* karata poput *Bing Maps* (URL 26), *Google Maps* (URL 27), *Waze* (URL 28), *MapQuest* (URL 29) i dr. su na prvi pogled besplatne, a ustvari podliježu raznim autorskim pravima i ograničenjima pri korištenju. Same karte jesu besplatne za pregledavanje, ali podaci koji su osnova za njihovo stvaranje i daljnju obradu nisu (Stilinović, 2013).

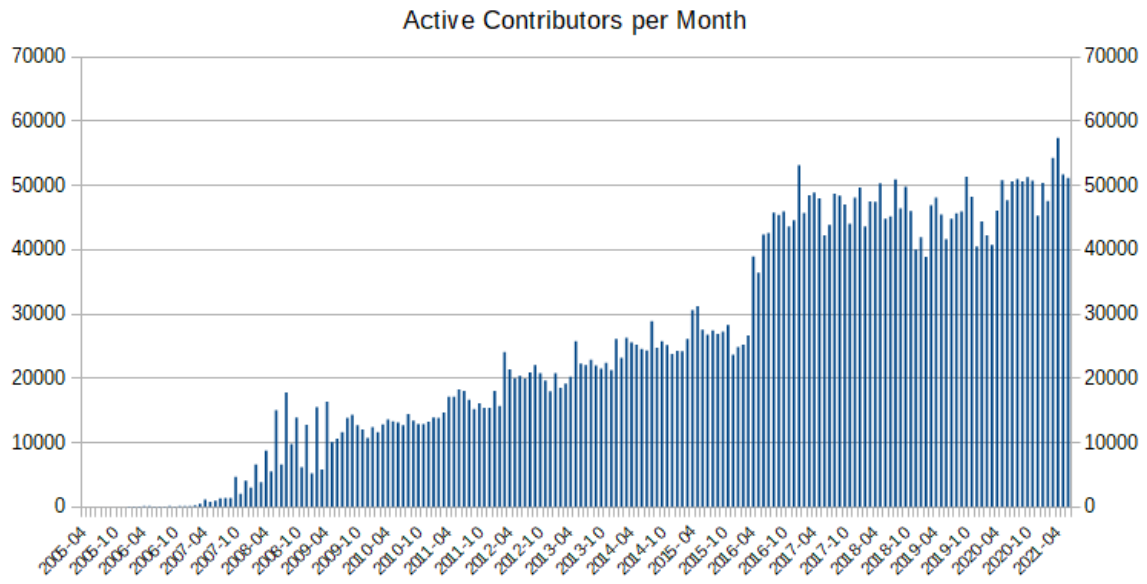
Takva ograničenost po pitanju dostupnosti geografskih podataka i njihove daljnje upotrebe potaknula je pokretanje *OpenStreetMapa*. Steve Coast pokrenuo je projekt 2004. godine, usmjerivši se prvotno na kartiranje Ujedinjenog Kraljevstva. Potom je 2006. godine

osnovana i zaklada *OpenStreetMap Foundation* s ciljem poticanja rasta, razvoja i distribucije slobodnih prostornih podataka koje bi mogli koristiti svi (URL 30).

Napredak tehnologije, poput jeftinih i prijenosnih GPS uređaja, značio je da je sada moguće stvarati vlastite karte u suradnji s drugima korisnicima bez gore navedenih ograničenja. Jedan od većih zamaha u razvoju *OSM*-a bilo je prešutno odobrenje za korištenje tada vrlo raširenih *Yahoo!* zračnih snimki kao podloga za kartiranje. Te zračne snimke korištene su od 2007. do 2011. godine i pomogle su u stvaranju preciznijih karata, posebice u nekim dijelovima svijeta. Utjecaj je bio golem jer se kartiranje obavljalo brže i točnije tijekom ključnih prvih godina razvoja *OSM*-a (URL 31).

Projekt OpenStreetMap očekivano je privukao pozornost akademske zajednice. Takav volonterski pristup prikupljanju podataka u početku je nazvan volonterske geoinformacije, a s vremenom je i sam OSM postao sinonim za VGI. Kako se uvelike povećavao broj doprinositelja (engl. *contributors*), na pojam volonterske geoinformacije se nadovezao i pojam masovnog prikupljanja informacija (engl. *crowdsourcing*).

Najrealniji pokazatelj konstantnog razvoja OSM-a je rast broja aktivnih doprinositelja prikazan na slici ispod.



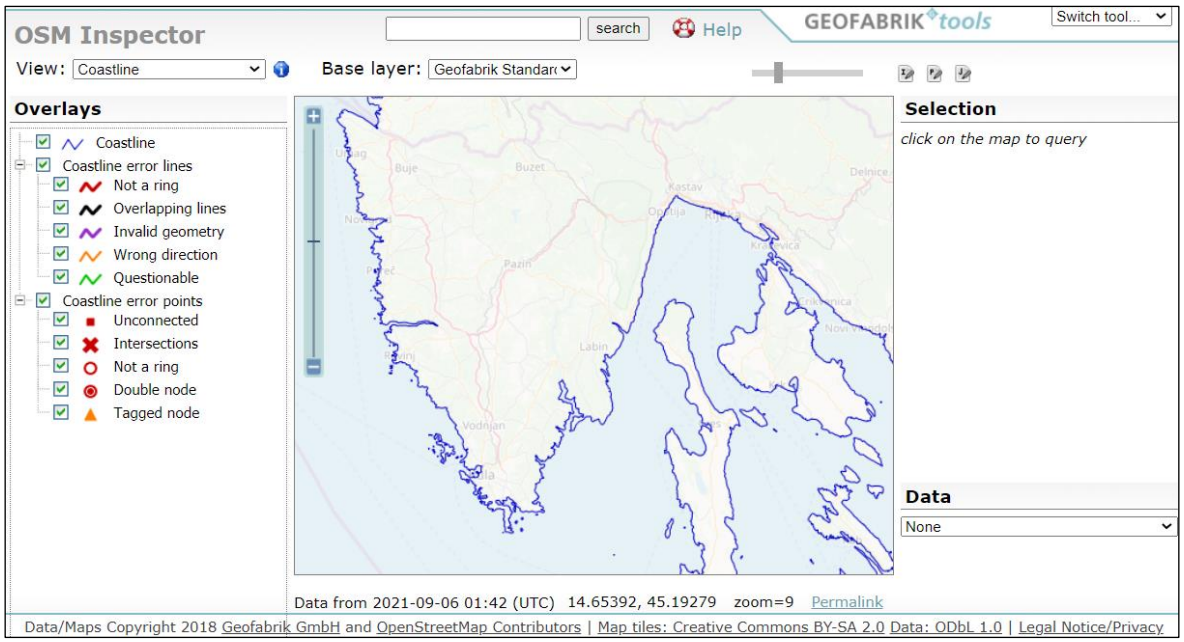
Slika 2.12. Prikaz porasta broja aktivnih mjesečnih doprinositelja u OSM-u (URL 32)

2.4.2 Kvaliteta OSM podataka

OpenStreetMap ima veliki potencijal u slobodnoj ponudi prostornih podataka, no zbog načina na koji su ti podaci nastali mora se postaviti pitanje kvalitete i njihove konzistentnosti. Podatke prikupljaju i dijele korisnici s različitim predznanjima i različitim potrebama.

Za svrhu nadzora kvalitete OSM je uspostavio *online* korisničku hijerarhiju s administratorima i moderatorima, međutim ta hijerarhija ne daje željene rezultate u praksi. Administratori, kojih ima jako malo, mogu mijenjati uloge drugih korisnika. Moderatori mogu privremeno blokirati druge korisnike, koji onda ne mogu koristiti aplikacijsko programsko sučelje tj. funkcije povezane s čitanjem podataka iz baze i zapisivanjem u nju. Blokada ima javno vidljivo tekstualno objašnjenje, što omogućava drugim članovima zajednice javno raspravljanje problema i blokade. Ukoliko se ispostavi da je blokada neutemeljena moderatori ju mogu opozvati (URL 33).

Postoji i čitav niz alata koji brinu o kvaliteta podataka unutar OSM-a (URL 34). Oni automatski detektiraju pogreške ili greške prijavljuju korisnici. Ideja je prikazati na karti anomalije identificirane u *OpenStreetMap* podacima. Svakako najpoznatiji među tim alatima su *OSM Inspector* (URL 35) i *Osmose* (URL 36) od kojih potonji ima ugrađene opcije prijedloga za ispravke i vodiča za ispravljanje. S obzirom da su obalne crte možda i najkompleksniji objekti u čitavom OSM-u (zamislimo samo čitavu Euroaziju iscrtanu jednom neprekinutom linijom) svojevremeno je bio vrlo korišten alat *Coastline error checker* koji je prije stvaranja obala izvršavao brojne automatizirane provjere, a također i neke automatizirane popravke podataka OSM-a kako bi pokušao dati potpuno spojenu obalu (URL 37). Danas se za tu namjenu koristi već spomenuti *OSM Inspector* s uključenom opcijom provjere obalne crte za koju se ažuriranje obavlja dva puta dnevno (URL 38).



Slika 2.13. Provjera obalne crte u sučelju alata OSM Inspector

Većina pogrešaka su nenamjerne, nastale zbog neiskustva novih članova. Ipak, određeni alati nadziru potencijalni vandalizam kombiniranjem slijedećih kriterija: novi korisnik, promjena korisničkog imena u odnosu na *ID*, broj uređivanja, površina uređivanja, puno specijalnih radnji poput preimenovanja i reklasifikacija, puno brisanja, pomicanje podataka na velike udaljenosti, smanjenje broja čvorova na putevima itd. (URL 39). U slučaju loših promjena, one se uvijek mogu poništiti vraćanjem na prijašnju verziju jer je u svakom trenutku moguć pristup cijeloj povijesti uređivanja podataka OSM-a od njegova početka.

2.4.3 Model podataka OSM-a

Da bi se uspješno koristili podatke *OpenStreetMapa* potrebno je dobro razumijeti njegov model podataka. Elementi su osnovne sastavnice konceptualnog modela podataka fizičkog svijeta u *OpenStreetMapu*. Postoje tri vrste osnovnih elemenata:

- čvorovi (engl. *nodes*) koji definiraju točke u prostoru,
- putevi (engl. *ways*) koji definiraju linijska obilježja i granice područja,
- relacije (engl. *relations*) koje se ponekad koriste za objašnjenje kako drugi elementi djeluju zajedno.

Sve tri vrste elemenata mogu imati jednu ili više pridruženih oznaka (engl. *tags*) koje opisuju značenje svakog pojedinog elementa (URL 40).

Čvor se sastoji od jedne točke u prostoru definirane zemljopisnom širinom, dužinom i ID - om čvora. Treća, izborna dimenzija (nadmorska visina) također se može uključiti korištenjem ključa „ele“ (skraćeno od engl. *elevation*). Čvorovi se mogu koristiti za definiranje samostalnih točkastih značajki, ali se češće koriste za definiranje oblika ili "putanje" puta.

Kada se koriste za definiranje samostalnih točkastih značajki, čvor će obično imati barem jednu oznaku za definiranje svoje namjene. Čvorovi mogu imati više oznaka i/ili biti dio relacije.

Više čvorova čine dio jednog ili više puteva, definirajući oblik ili "putanju" puta. Tada čvorovi obično nemaju oznake. Tamo gdje se putevi križaju na istoj nadmorskoj visini, dva puta moraju dijeliti čvor (na primjer, čvor na cesti). Ako se autoceste ili željeznice sijeku na različitim visinama bez povezivanja, ne bi trebale dijeliti čvor (URL 41).

Put bi u svakodnevnom govoru bio linija. Obično predstavlja linearno obilježje na tlu poput ceste, zida ili rijeke. Tehnički gledano put je uređeni popis čvorova koji ima barem jednu oznaku ili je uključen u relaciju. Može imati između dva i 2000 čvorova, iako je moguće da postoje neispravni put s nula ili jednim čvorom. Put može biti otvoren ili zatvoren.

Kod otvorenog puta prvi i posljednji čvor nisu identični. Uobičajeni primjeri uključuju većinu cesta, potoka i željezničkih pruga, jer one počinju na jednom mjestu, a završavaju na drugom. U bazi podataka put uvijek ima smjer čak i ako je značajka koju predstavlja „dvosmjerna“ (npr. većina cesta s dvosmjernim prometom) gdje promet prolazi u oba smjera) ili ako nema smjer (npr. zid).

Kod zatvorenog puta posljednji čvor puta identičan je prvom čvoru. Zatvoreni put može se tumačiti ili kao zatvorena polilinija, ili kao područje, ili oboje, ovisno o oznakama.

Zatvoreni put s oznakom „area=yes“ uvijek treba tumačiti kao područje, iako oznaka u većini slučajeva nije potrebna.

Moguće je da zatvoreni put bude označen na način da ga treba tumačiti i kao zatvorene polilinije i kao područje. Primjer je travnata površina unutar kružnog toka. U tom slučaju to

se područje tretira i kao zatvorena polilinija koja definira kružni tok i kao područje s upotrebom zemljišta trava (URL 42).

Relacija je grupa elemenata. Sastoji se od jedne ili više oznaka i poredanog popisa jednog ili više čvorova, puteva i/ili relacija kao članova. Koristi se za definiranje logičkih ili geografskih odnosa između drugih elemenata. Primjeri relacja su:

- relaciju rute, koja popisuje puteve koji tvore glavnu (označenu) autocestu, biciklističku ili autobusnu rutu;
- ograničenje skretanja koje kaže da se ne može skrenuti s jednog puta na drugi;
- multipoligon koji opisuje područje, gdje je granica definirana „vanjskim putom“ a rupe su definirane „unutarnjim putevima“.

Stoga, relacije mogu imati različita značenja koja su definirana oznakama. Relacija obično ima oznaku „tip“ (engl. *type*), dok se druge oznake relacije tumače u skladu s oznakom tipa. Svaki element (član relacije) može imati ulogu (engl. *role*) unutar relacije. Pojedini element, kao pojedinačni put, može se pojaviti više puta u relaciji (URL 43).

Ranije smo spomenuli da zatvoreni put može definirati područje (ispunjeni poligon). Većina zatvorenih puteva smatra se područjima čak i bez oznake area = yes. Za oznake koje se mogu koristiti za definiranje zatvorenih polilinija potrebno je dodati i oznaku area = yes ako želimo definirati područje. Naprimjer, za definiranje pješačke zone ili trga. Područja se također mogu opisati pomoću jednog ili više načina koji su povezani s multipoligonskom relacijom.

Složenija područja se definiraju s multipoligon relacijom. To bi bio skup petova koji definiraju jednu ili više vanjskih granica, i po potrebi nula ili više unutarnjih granica tj. „rupa“ (URL 44).

Svi tipovi elemenata podataka mogu imati oznake koje opisuju značenje pojedinog elementa uz koji su pridružene. Svaka oznaka se sastoji od dva tekstualna polja - ključa (engl. *key*) i vrijednosti (engl. *value*). Obje vrijednosti su Unicode znakovni niz do 255 znakova. Naprimjer, oznaka highway=residential definira put kao cestu čija glavna funkcija je osigurati ljudima pristup domovima. Nema striktno utvrđenog rječnika s oznakama, ali postoje mnoge konvencije dokumentirane na *wiki* stranicama OSM-a.

2.4.4 OSM formati datoteka

Od postojećih formata za razmjenu i prijenos OSM podataka među najznačajnijima su OSM XML, PBF i o5m.

Ideja OSM XML formata bila je stvoriti jezik koji će u isto vrijeme biti čitljiv i računalnom programi i ljudima. Struktura zapisivanje podataka je granajuća što je čini vrlo je sličnoj HTML-u. Glavne prednosti XML-a su jednostavnost čitanja zbog jasne strukture, neovisnost o hardveru, mogućnost prilagodbe raznim konkretnim formatima te dobar omjer kompresije. Većina OSM alata koristi upravo XML format. Strukturalno gledajući, XML je u lista svih elemenata osnovnih tipova podataka (čvorova, puteva i relacija) koji čine arhitekturu OSM podataka. OSM XML datoteka najčešće ima .osm ekstenziju. Glavni nedostatak XML-a je veličina datoteke što rezultira sporom obradom i zapisivanjem podataka. Stoga je nekad potrebno dekompresirati datoteku prije upotrebe (Stilinović, 2013).

Upravo zbog navedenog nedostatka XML datoteka, kao njegova alternativa nastao je PBF format. Datoteka PBF formata otprilike je duplo manja, ispisuje se pet puta brže, a čita i do šest puta brže. Glavni nedostaci PBF formata su što nije čitljiv ljudima, teško se uređuje, a za funkcije kompresije aplikacije trebaju *zlib* biblioteku (URL 45).

Format o5m nastao je kao kompromis između OSM XML i PBF formata kombinirajući prednosti oba formata. Strukturom je sličan OSM XML format, pa se procedure unosa i ispisa postojećih aplikacija za obradu OSM podataka mogu prilagoditi ovom formatu uz male preinake. Kodiranje podataka vrlo je slično PBF formatu, pa gotovo da i nema razlike u veličini datoteke. Prednosti ovog formata su male veličine datoteka, spajanje dvije ili više datoteka i do pet puta brža obrada podataka u odnosu na OSM XML format zbog čega je i dobio naziv o5m. Nedostaci su što nije čitljiv ljudima i nemogućnosti uređivanja uobičajenim programima za uređivanje teksta (engl. *text editors*) (URL 46).

3. KORIŠTENE TEHNOLOGIJE

U ovom poglavlju bit će nabrojani i objašnjeni programi, moduli, aplikacije i ostali alati korišteni u procesu izrade karte.

3.1 OSMFILTER

Program *osmfilter* je alat naredbenog retka (engl. *command line tool*) koji se koristi za filtriranje OSM datoteka pomoću oznaka (engl. *tags*) specifičnih za objekte koje želimo dohvatiti. Podržani ulazni i izlazni formati su OSM XML format (*.osm) i o5m format (*.o5m). Kako bi se ubrzao proces obrade podataka preporučuje se korištenje o5m formata barem za unos (URL 47).

Sintaksa programa *osmfilter* je jednostavna i prikazana je u redu ispod.

```
./osmfilter ulazna_datoteka.o5m izrazi_s_filterima -o=izlazna_datoteka.osm
```

Najčešći slučajevi uoptrebe ovog programa su: filtriranje objekata (engl. *object filter*), filtriranje oznaka (engl. *tags filter*), izmjena oznaka (engl. *tags modification*), dohvaćanje statistike oznaka (engl. *getting tag statistics*) i korištenje datoteke parametara (engl. *parameter file*).

Pri izradi karte ćemo koristiti samo filtriranje objekata i oznaka, pa će ta dva načina upotrebe biti objašnjena u sljedećim poglavljima.

3.1.1 Filtriranje objekata

Filteri ove kategorije uvijek će utjecati na cijeli objekt - čvorove, puteve ili relacije (engl. *nodes, ways, relations*). To znači da ovi filteri odlučuju za svaki objekt hoće li se zadržati u datoteci ili ne. Tako je moguće zadržiti određene objekte i njima ovisne objekte. Primjer za to su sljedeći izrazi:

```
--keep= --keep-relations="route=bus"
--keep="highway=cycleway and lit=yes"
--keep="admin_level=6 and name=Nürnberger\ Land"
```

gdje se vrijednosti nalaze među navodnim znacima i uz napomenu da svakom pojedinom znaku razmaka unutar vrijednosti mora prethoditi kosa kosa crta. Zatim je moguće zadržati određenu vrstu objekata. Da bi to postigli navedimo filter koji odgovara samo toj vrsti

objekata, a ostale vrste moramo ili ispustiti (engl. *drop*) ili navođenjem praznog filtera zaobići podudaranje. Kad bismo htjeli dobiti samo čvorove s oznakom `fixme =*` upotrijebili bi jedan od ekvivalentnih izraza prikazanih ispod.

```
--keep= --keep-nodes="fixme="
--keep-nodes="fixme=" --keep-ways= --keep-relations=
```

Kao što je navedeno iznad, moguće je ispustiti određene objekte, pa čak i kombinirati zadržavanje jednih, a ispuštanje drugih objekata.

```
--keep="highway=" --drop="access=no"
```

Moguće je izbrisati sve objekte određene vrste primjenom jedne od ovih izraza, a dopuštene su i kombinacije.

```
--drop-nodes
--drop-ways
--drop-relations
```

Osim navedenih, postoje i napredniji načini filtriranja. Jedan od naprednih načina filtriranja je zanemarivanje međuobjektne ovisnosti. Obično će i svi objekti koje koristi uključeni objekt također biti sačuvani u podacima. Na primjer, filter `--keep="highway="` neće obuhvatiti samo sve ceste već i njihove čvorove. Isto se odnosi i na relacije i njihove članove. Ako ne želimo da program uzima u obzir ove ovisnosti, koristit ćemo sljedeći izraz:

```
--ignore-dependencies
```

Za definiranje složenih filtera mogu se koristiti binarni logički operatori „AND“ i „OR“. Prvi ima veći prioritet od potonjeg, pa će se prvo izračunati izrazi s operatoron "AND". Ovaj zadani redoslijed može se zamijeniti korištenjem zagrada koje pritom moraju biti odvojene razmacima od svih drugih zagrada, operanada ili operatora. Primjer jednog takvog izraza s logičkim operatorima je:

```
--keep="place=city or ( place=town and population>=10000 )"
```

Prilikom filtriranja možemo upotrijebiti šest različitih vrsta usporedbi koristeći simbole:
`= != < > <= >=`

Program će prema zadanim postavkama uspoređivati vrijednosti ASCII-abecedno. Za brojčane vrijednosti, usporedba će se izračunati numerički. Naprimjer, brojčana vrijednost „2“ dolazi ispred brojčane vrijednost „10“, iako bi po abecednom redu „10“ bilo prije. Kada

se koristi operator usporedbe „=“, zamjenski znakovi (engl. *wildcards*) su dopušteni na početku i na kraju vrijednosti. Zamjenski znak je simbol koji se koristi za zamjenu ili predstavljanje jednog ili više znakova. Primjer korištenja zamjenskog znaka:

```
--keep="name=main* and highway=*ary and source=*aerial*"
```

Posebne oznake koristimo ako je potrebno usporediti metapodatke. Sljedeći izrazi mogu se koristiti kao ključne vrijednosti filtra:

```
@uid @id @user
```

3.1.2 Filtriranje oznaka

Filter oznaka (engl. *tags*) uvijek se odnosi samo na neke određene oznake, a nikada ne odlučuju o postojanju cijelih objekata. Pomoću takvog filtera možemo definirati oznake koje želimo zadržati u datoteci ili isključiti. Primjeri zadržavanja isključivo određene oznake:

```
--keep-tags="all amenity=restaurant =fast_food"
--keep-tags="all highway= waterway= name="
```

Ovu opciju treba koristiti oprezno jer može filtrirati esencijalne oznake. Na primjer, nezadržavanje `type=` oznake prekinulo bi daljnju obradu relacija. Primjer odbacivanja određene oznake je:

```
--drop-tags="oneway= name="
```

Za većinu primjena tzv. autorske oznake nisu potrebne. Ako želimo izuzeti korisničko ime (engl. *user name*), korisnički ID (engl. *user id*), skup promjena (engl. *changeset*) i vremensku oznaku (engl. *timestamp*), dodajemo sljedeći argument:

```
--drop-author
```

Ako želimo ispustiti ne samo podatke o autoru, već i brojeve verzija (engl. *version numbers*), upotrijebit ćemo opciju:

```
--drop-version
```

Ipak, treba znati da će većina programa odbiti rad s našim podacima nakon što smo izbrisali brojeve verzija.

3.2 OSMCONVERT

Program *osmconvert* čita OSM datoteke različitih formata i vrši konverziju u odabrani format izlazne datoteke. Najčešći slučajevi uoptrebe ovog programa su: konverzija datoteka, primjena geografskih granica, izuzmanje informacija ili sadržaja iz izlazne datoteke, izmjena oznaka, ažuriranje osm datoteka, dohvaćanje razlika između dviju osm datoteka, postavljanje vremenske oznake datoteke, dohvaćanje statističkih podataka, ispis *.csv datoteka, kombiniranje funkcija, paralelna obrada i korištenje datoteke parametara.

Pri izradi karte ćemo koristiti samo konverziju datoteka i izuzmanje informacija ili sadržaja iz izlazne datoteke, pa će ta dva načina upotrebe biti objašnjena u sljedećim poglavljima.

3.2.1 Konverzija datoteka

Formati koje program može čitati su: *.osm, *.osc, *.osc.gz, *.osh, *.o5m, *.o5c i *.pbf. Formati u koje program može konvertirati učitane datoteke su: *.osm, *.osc, *.osh, *.o5m, *.o5c i *.pbf. Program je dostupan u interaktivnom načinu, ali u njemu nisu dostupne sve mogućnosti. Stoga je preporuka *osmconvert* pokrenuti i koristiti u naredbenom retku (engl. *command line*). Ako želimo usmjeriti izlaz programa na standardni izlaz, potrebno je jednom od sljedećih opcija programu reći koji format podataka će se koristiti: --out-osm (zadana postavka), --out-osc, --out-osh, --out-o5m, --out-o5c, --out-pbf. Ako ispred naziva izlazne datoteke koristimo opciju `-o=` program će odrediti format podataka na temelju ekstenzije iza naziva datoteke. Primjeri za oba načina određivanja formata izlazne datoteke:

```
osmconvert croatia.pbf >croatia.osm
osmconvert croatia.pbf -o= croatia.o5m
osmconvert croatia.pbf --out-o5m > croatia.o5m
```

Ako podatke unosimo putem standardnog ulaza, tada možemo koristiti i komprimirane ulazne datoteke. Slijede primjeri.

```
bzcat croatia.osm.bz2 | osmconvert - -o=croatia.o5m
osmconvert croatia.pbf | gzip -1 >croatia.osm.gz
```

Opcija „-“ obavještava program da očekuje ulazne podatke putem standardnog unosa. Ugrađeni algoritam za dekompresiju manje je moćan od specijaliziranih programa za dekompresiju jer je moguće dekomprimirati samo *.*gz* datoteke. Sukladno tome program

sam prepoznaje gzip kompresiju pa nije važno koja je ulazna datoteka gzip-komprimirana, a koja nije.

3.2.2 Izuzmanje informacija ili sadržaja iz izlazne datoteke

Kao i kod programa *osmfilter* moguće je izuzeti korisničko ime, korisnički ID, skup promjena i vremensku oznaku objekta opcijom naredbenog retka `--drop-author`. Na primjer:

```
osmconvert --drop-author a.pbf -o=a.osm
```

U pravilu neće biti problema pri brisanju podataka o autoru iz *.osm ili *.o5m datoteka, međutim, ne preporučuje se raditi to s *.pbf datotekama jer se većina programa neće nositi s ovom promjenom formata. Ukoliko kasnije trebate ponovno dodati podatke o autoru, jer npr. sljedeći program ovisi o ovom formatu, možete ih iznova generirati s opcijom `--fake-author`. Naravno, nove informacije o autoru bit će samo zamjenske vrijednosti koje se pridržavaju opisa formata.

3.3 OGR2OGR

Modul *ogr2ogr* služi za konverziju i reprojiciranje prostornih podataka u vektorskim formatima, a dolazi s OGR bibliotekom (engl. *library*). OGR je nekad bio zasebna vektorska IO biblioteka koja je bila odvojena od GDAL biblioteke do izdanja GDAL 2.0 kada su integrirane u GDAL/OGR. Sama GDAL/OGR biblioteka ima OGR OSM upravljački program (engl. *driver*) (URL 48). Ovaj upravljački program čita datoteke OSM-a u *.osm (baziran na XML-u) i u *.pbf (optimizirani binarni) formatu, a kategorizirati će značajke (engl. *features*) u pet slojeva (engl. *layers*):

- points: "čvorovi" (engl. *node*), točkaste značajke koje imaju priložene oznake (engl. *tags*)
- lines: "putevi" (engl. *ways*), značajke koje nisu površina
- multilinestrings: "relacije" (engl. *relation*) koje tvore niz linija (engl.*lines*), a mogu imati oznake type = ‘multilinestring’ ili type = ‘route’
- multipolygons: "relacije" (engl. *relation*) koje tvore multipoligone (mogu imati oznaku type = ‘multipolygon’ ili type = ‘boundary’) i "putevi" (engl. *ways*) koji su prepoznati kao površina (engl. *area*)

- other_relations: "relacije" koje ne pripadaju u prošla dva navedena sloja (URL 49).

Tijekom procesa konverzije *ogr2ogr* može izvesti različite operacije kao što su prostorni odabir, odabir atributa, smanjenje skupa atributa, postavljanje izlaznog koordinatnog sustava ili čak reprojektiranje značajki (engl. *features*). Sam modul ima veliki broj opcija koje je moguće prilagoditi svojim zahtjevima. Ovdje ćemo navesti i objasniti one koje će biti korištene tijekom procesa izrade karte.

`--config` poziva na korištenje konfiguracijske datoteke (vidi poglavlje 3.4).

`-f "format_name"` opcija za definiranje naziva formata izlazne datoteke, npr. „ESRI Shapefile“, „MapInfo File“, „PostgreSQL“. Ukoliko nije naveden, format se pogađa iz ekstenzije, iako je do GDAL verzije 2.3 *ESRI Shapefile* format bio zadan.

`-sql "sql_statement"` opcija za izvršavanje SQL izraza. Rezultirajuća tablica/sloj će bit spremljena u izlaznu datoteku. Počevši od GDAL 2.1, sintaksa `@filename` može se upotrijebiti kako bi se ukazalo da je sadržaj u toj označenoj datoteci.

`-overwrite` opcija koja briše izlazni sloj (ukoliko već postoji) i ponovno ga kreira praznog.

`-skipfailures` nastavlja proces i nakon neuspjeha, preskačući neuspješnu značajku (engl. *features*).

`-nlt [TYPE]` definira vrstu geometrije za kreirani sloj. Moguće su sljedeće vrste geometrije (engl. *type*): NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON, MULTILINESTRING, CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE i MULTISURFACE. Moguće je dodati Z, M, ili ZM imenu tipa geometrije kako bi odredili koordinate s visinom i mjerilom. PROMOTE_TO_MULTI se može koristiti za automatsko promicanje slojeva koji miješaju poligone ili multipoligone u multipoligone i slojeva koji miješaju nizove linija (engl. *linestrings*) ili višestruke linije (engl. *multilinestrings*) u višestruke linije. Može biti korisno pri pretvaranju *shapefile* datoteka u PostGIS i druge upravljačke programe (engl. *drivers*) koji provode strogu provjeru vrste geometrije. CONVERT_TO_LINEAR se može koristiti za konvertiranje nelinearnih tipova geometrije u linearne tipove geometrije njihovim aproksimiranjem, a CONVERT_TO_CURVE za promicanje nelinearnog tipa u njegovu generaliziranu vrstu krivulje (POLYGON u

CURVEPOLYGON, MULTIPOLYGON u MULTISURFACE, LINESTRING u COMPOUNDCURVE, MULTILINESTRING u MULTICURVE). Neke prisilne konverzije geometrije mogu rezultirati nevažećim (engl. *invalid*) geometrijama. Počevši od GDAL 3.0.5, CONVERT_TO_LINEAR i PROMOTE_TO_MULTI mogu se koristiti istovremeno.

`-lco [NAME=VALUE]` opcija stvaranja sloja (engl. *layer creation option*).

`-append` dodaje podatke u postojeći sloj umjesto stvaranja novog.

`-update` otvara postojeću izlaznu datoteku u načinu ažuriranja (engl. *update mode*) umjesto pokušaja stvaranja nove.

`-simplify [tolerance]` definirat će toleranciju udaljenosti za proces pojednostavljivanja. Upotrijebljeni algoritam čuva topologiju po značajci, osobito za geometrije poligona, ali ne i za cijeli sloj.

Popis ostalih opcija koje je moguće koristiti u modulu *ogr2ogr* nalazi se na *web* stranici GDAL biblioteke (URL 50).

3.4 OSMCONF.INI DATOTEKA

Konfiguracijska datoteka osmconf.ini koristi se za uvoz OSM datoteka. U mapi podataka GDAL distribucije nalazi se datoteka osmconf.ini (URL 51) koja se može prilagoditi vlastitim potrebama. Također je moguće definirati alternativnu putanju do osmconf.ini datoteke s konfiguracijskom opcijom OSM_CONFIG_FILE. Prilagođavanjem same datoteke definiramo koji će se OSM atributi potencijalno sačuvati prilikom konverzije iz OSM datoteke (URL 52). OSM sam po sebi ima ogroman broj mogućih atributa, pa se ovdje na neki način radi o filtraciji atributa što za poslijedicu ima bržu *ogr2ogr* konverziju i manju i „čišću“ izlaznu datoteku, a sukladno tome i atributnu tablicu. Sama osmconf.ini datoteka je jednostavne strukture. Uobičajeno se na početku definiraju atributi i njihove vrijednosti za puteve (engl. *ways*) koji će se smatrati poligonima ukoliko su zatvoreni. U našem slučaju to je:

```
closed_ways_are_polygons=aeroway,amenity,boundary,building,craft,geological,his
toric,landuse,leisure,military,natural,office,place,shop,sport,tourism
```

Idućim retkom definirali da u imenima atributa želimo sve znakove „:“ pretvoriti u zanak „_“.

```
attribute_name_laundering=yes
```

Nadalje, za svaki od pet slojeva OSM datoteke (vidi poglavlje 3.3) je definiran „blok“ s popisom atributa kojima dodajemo željenu vrijednost. Na tzv. sistemske (uobičajene) atribute koje bi trebao imati svaki OSM objekt stavljamo vrijednost „yes“ ili „no“, iza „atributes=“ navodimo atribute koje želimo zadržati, iza „ignore=“ navodimo atribute koje odbacujemo, a pod „other_tags=“ stavljamo vrijednost „no“ ako želimo spriječiti kreiranje polja s ostalim atributima koji nisu obuhvaćeni ovim filtriranjem. U našoj osmconf.ini datoteci će svih pet „blokova“ imati iste atribute i vrijednosti koji nam trebaju pri izradi karte, a to su:

```
osm_id=yes
osm_version=no
osm_timestamp=no
osm_uid=no
osm_user=no
osm_changeset=no
attributes=name,name:$OSM_LANG,population,type,place,ISO3166-1:alpha2,ISO3166-
1,admin_level,boundary,capital,intermittent
unsignificant=created_by,converted_by,source,time,ele
ignore=created_by,converted_by,source,time,ele,note,openGeoDB:,fixme,FIXME
other_tags=no
```

Atribut *name* predstavlja „lokalno“ ime (izvorno), *name:$OSM_LANG* je ime na jeziku karte kojeg sami definiramo, *population* je broj stanovnika što nam treba kod prikaza gradova, *type* nam govori o kojem tipu objekta se radi, *place* nam govori o kojoj vrsti naselja se radi, *capital* nam govori radi li se o glavnom gradu države, a *intermittent* je li neka vodena površina, u našem slučaju jezero, (privremeno) bez vode tj. presušila. Radi jednostavnosti kreirat će se samo jedna osmconf.ini datoteka koja će se pozivati po potrebi.

3.5 UBUNTU

Ubuntu je potpuni *Linux* operativni sustav, slobodno dostupan uz profesionalnu podršku i podršku zajednice. Ubuntu zajednica izgrađena je na idejama sadržanim u Ubuntu Manifestu koji kaže da bi softver trebao biti dostupan besplatno, da bi softverske alate ljudi trebali koristiti na njihovom lokalnom jeziku unatoč bilo kakvim teškoćama te da bi ljudi

trebali imati slobodu prilagođavanja i izmjene svog softvera na bilo koji način koji smatraju prikladnim. Ubuntu uključuje tisuće softvera, pokrivajući sve standardne aplikacije za stolna računala od aplikacija za obradu teksta i proračunskih tablica do aplikacija za pristup internetu, softvera web poslužitelja, softvera za e-poštu, programskih jezika i alata (URL 53).

3.6 TERMINAL

Terminal je sučelje naredbenog retka (engl. *Command Line Interface*, CLI) za interakciju s programima na računalu u kojem je moguće izravno upisivati i izvršavati tekstualne naredbe. Poznat je i pod nazivima *Console*, *Shell*, *Command line*, *Command prompt*.

3.7 SHELL SCRIPT

Shell skripta (engl. *shell scrip*) je računalni program dizajniran za pokretanje iz *Unix/Linux* „ljuske“ (engl. *shell*). U računarstvu je *ljuska* naziv za sučelje između čovjeka i jezgre (kernela) operacijskog sustava. Korisnici obično komuniciraju s Unix „ljuskom“ pomoću terminalskog emulatora kao što je *Terminal*. *Shell* skripte imaju nekoliko potrebnih elemenata koji govore okruženju jezgre što i kada treba učiniti.

3.8 PYTHON

Python (URL 54) je programski jezik visoke razine koji je dinamičan, objektno-orijentiran, interpreterski i interaktivan. Podržava više stilova programiranja, a dostupan je pod licencom koja omogućuje slobodno korištenje i distribuciju, čak i u komercijalne svrhe. Sintaksa mu je jasna i čitka što omogućuje brzo pisanje kôda. Dostupan je na većini poznatih platformi zbog čega je široko rasprostranjen i ima veliku zajednicu korisnika. *Python* ima opsežnu standardnu biblioteku s ugrađenim modulima, a funkcionalnosti je moguće proširiti dodatnim modulima i paketima (Stilinović, 2013).

3.9 QGIS

QGIS (URL 55) je profesionalna GIS aplikacija i razvojna platforma koju razvija tim predanih volontera i organizacija. *QGIS* je izgrađen kao besplatni softver otvorenog kôda.

Radi na operativnim sustavima *Linux*, *Unix*, *Mac OS* i *Window* te podržava brojne vektorske, rasterske i formate baza podataka i funkcionalnosti. *QGIS* pruža kontinuirano rastući broj mogućnosti koje pružaju osnovne funkcije i dodaci. Možete vizualizirati, upravljati, uređivati, analizirati podatke i dizajnirati karte. Korisnik ima slobodu prilagođavanja aplikacije vlastitim potrebama, od prilagođenih obrazaca za unos podataka do prilagođenog korisničkog sučelja i tijeka rada. Korisnici *QGIS*-a su iz raznih područja, uključujući: vlade, obrazovanje, planiranje, inženjering, nevladine organizacije, vojsku i itd. (URL 56).

3.10 GRASS GIS

GRASS GIS (URL 57), punog naziva Geographic Resources Analysis Support System, je geografski informacijski sustav otvorenog kôda koji pruža moćne mogućnosti rasterske, vektorske i geoprostorne obrade. Može se koristiti ili kao samostalna aplikacija ili kao pozadina za druge softverske pakete, kao što su *QGIS* i *R*, ili u *oblaku* (engl. *cloud*). Distribuira se slobodno pod uvjetima GNU Opće javne licence (GPL) (URL 58).

4. METODOLOŠKI PRISTUP

4.1 IZVORI PODATAKA

Budući da je predmet ovog rada provođenje testiranja i ažuriranja softvera i automatskog postupka izrade karte, naši primarni podaci su oni koje su autori javno objavili i učinili dostupnim za preuzimanje (URL 3). Taj paket podataka se sastoji od:

- niza izvršnih skripti i *Python* programa,
- preddefiniranog *QGIS* projekta *OSM_World_Political_Map.qgs* u kojem će se automatski učitavati izrađeni slojevi odn. cijela karta na kraju postupka,
- direktorija „PK“ s *Reference_OSMPoliticalMap.qgs* referentnom kartom i direktorijijima s podacima potrebnim za njeno generiranje.

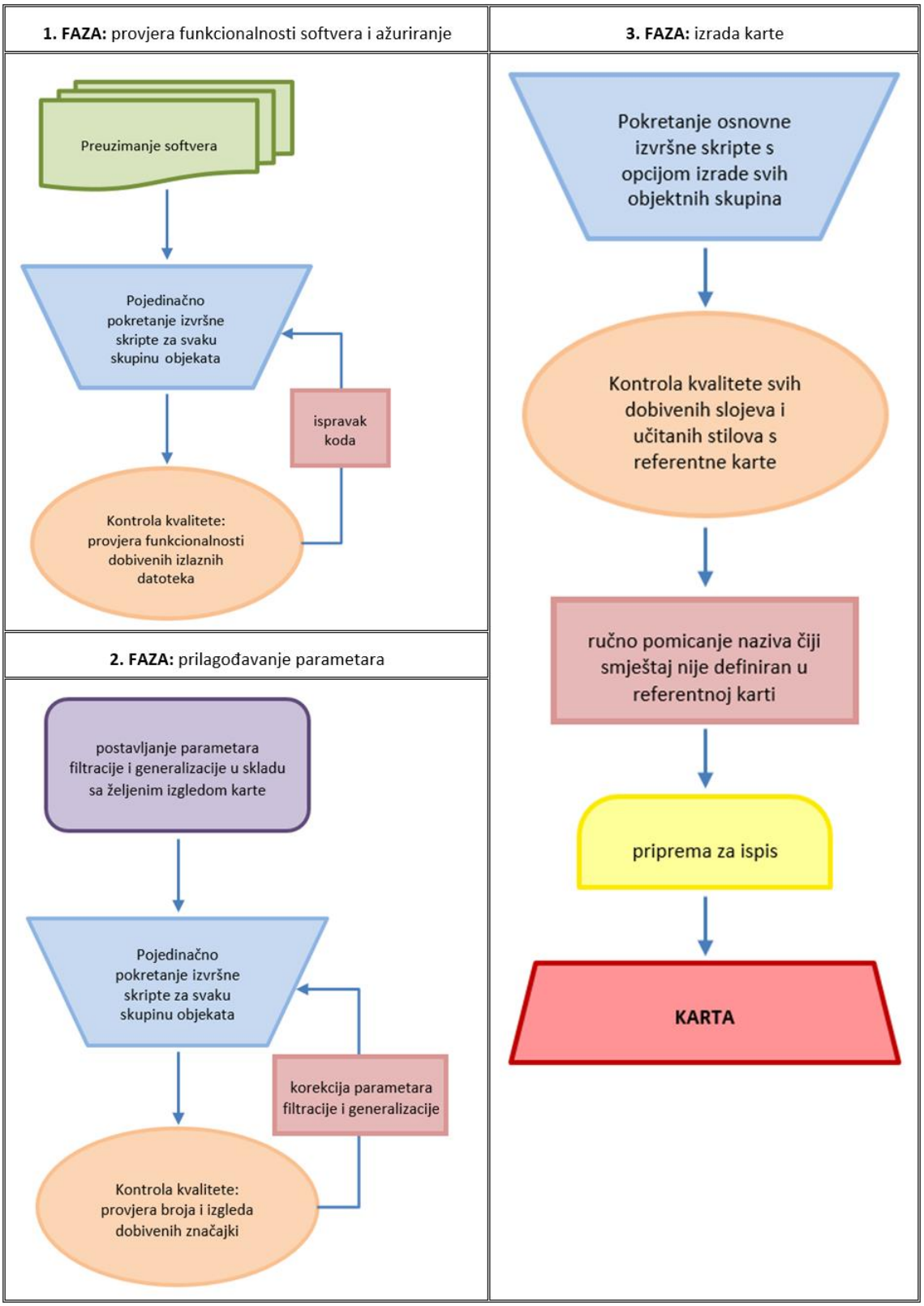
Prema dostupnom opisu postupka (URL 59) nakon pokretanja procesa svi podaci potrebni za izradu karte, koji se ne nalaze u setu primarnih podataka, će se preuzeti tijekom procesa, a važno je osigurati stabilnu i brzu podatkovnu mrežu za pristup internetu.

4.2 RAČUNALNA PODRŠKA

U automatiziranom postupku izrade karte kojeg planiramo provesti biti će korištene velike datoteke. Posebno se to odnosi na OSM datoteku za cijeli planet (tzv. *planet file*) koja je u trenutku pisanja ovog rada ima veličinu 58,8 GB u kompresiranom PBF formatu, i 93,1 GB u o5m formatu iz koje ćemo filtrirati određene podatke više puta puta tijekom procesa izrade karte. Za izradu ove karte korišteno je računalo koje specifikacijama, za današnje pojmove, ne spada u kategoriju snažnih računala, ali zadovoljava preporuke autora.

Na računalu je instaliran operativni sustav Ubuntu 20.04 LTS (Focal Fossa) 64-bit, procesor je Intel® Core™ i7-3520M CPU @ 2.90GHz × 2, količina radne memorije (RAM) iznosi 8,00 GB, a za pohranu je korišten vanjski tvrdi disk kapaciteta 2TB.

4.3 DIJAGRAM TOKA ISTRAŽIVANJA



Slika 4.1. Dijagram toka istraživanja

5. IZRADA POLITIČKE KARTE SVIJETA IZ PODATAKA OPENSTREETMAPA

5.1 PREUZIMANJE SLUŽBENOG SOFTVERA ZA IZRADU

Kao što je već rečeno u poglavlju 4.1, softver s kojim se može, u automatiziranom postupku, izraditi politička karta svijeta iz podataka *OpenStreetMapa* dostupan je za preuzimanje s digitalnog repozitorija Github. Sve preuzete datoteke je potrebno spremiti u isti direktorij proizvoljnog naziva. To je bitno zbog automatskog pozivanja i izvršavanja skripti i programa tokom procesa izrade.

5.2 AUTOMATIZIRANO KREIRANJE OBJEKTNIH CJELINA

Preduvijet za uspješnu automatiziranu izradu karte su pravilno instalirani potrebni softveri koji će se koristiti u procesu izrade. Ideja autora je bila da se u izradi koriste isključivo softveri otvorenog kôda (engl. *open source software*), a s obzirom da se u ovom radu radi ažuriranje softvera za izradu karte odlučeno je testirati ga i prilagoditi najnovijim stabilnim verzijama potrebnih softvera. Od geoinformacijskih softvera koristit će se zadnje stabilne verzije za *QGIS* (3.20.1 Odense) i *GRASS GIS* (7.8.5). Operativni sustav Ubuntu 20.04 LTS (Focal Fossa) dolazi s predinstaliranim programom *Python3* u verziji *Python 3*.8 (URL 60), pa ga nije bilo potrebno nakadno instalirati. Instaliran je i *pyproj Python* paket u verziji 3.1.0 sa PROJ verzijom 8.0.1. Za korištenje *ogr2ogr* modula obavezna je instalacija iste verzije GDAL/OGR biblioteke za Ubuntu i GDAL *Python* biblioteke, u ovom slučaju instalirana je verzija GDAL 3.2.1.

Zbog novih verzija softvera bile su nužne neke promjene tehničke prirode u samom kôdu softvera za izradu karte, ali one neće uvijek biti posebno naglašene i obrazložene u ovom radu. Neki od primjera takvih promjena su pozivanje *Python3* programa naredbom „python3“ umjesto „python“, novi način pozivanja modula u *Python3* programu, novi način pisanja sintakse „print“ naredbe u *Python3* programu gdje se ono što želimo ispisati (sadržaj iza ključne riječi „print“) stavlja u zagrade.

Sve autorske dijelove softvera za izradu karte koji su preuzeti za potrebe ovog rada (izvršne skripte, *Python* programi, *QGIS* projekti) ćemo u ostatku ovog rada oslovljavati kao „originalne“, a naše verzije koje će biti predložene nakon testiranja kao „ažurirane verzije“.

U originalnoj verziji izvršna skripta *_8_Labels.sh* je služila za pokretanje općeg *Python* programa *_8_copyfields.py* koji bi u atributne tablice slojeva s državama, rijekama, jezerima, gradovima i morima kopirao potrebne vrijednosti za smještaj i izgled naziva objekata. Prilikom testiranja originalnog softvera pokretali smo izradu jednog po jednog sloja, što softver i nudi uz opciju izrade cijele karte, i tada se pojavio jedan tehnički problem. Naime, kada bi se nakon pojedinačno kreiranog sloja s objektnom skupinom (nekog od gore navedenih) pokrenula izvršna skripta *_8_Labels.sh* javljale bi se greške jer je izvršna skripta *_8_Labels.sh* zamišljena da radi promjene na svim navedenim objektnim skupinama objekata, iako nisu sve netom kreirane. Doduše, za netom kreirani sloj bi se postupak koji radi opći *Python* program *_8_copyfields.py* uspješno izvršio i sadržaj bi bio kopiran iz referentne karte u atributnu tablicu stvorenog sloja, ali dojava greški bi mogla djelovati zbunjujuće za mnoge korisnike. Stoga je u ažuriranoj verziji skripti za izradu objektnih cjelina dodan, u svaku skriptu zasebno, pripadajući dio kôda iz izvršnae skripte *_8_Labels.sh*, pa se taj dio procesa obavlja odmah nakon kreiranja pojedinog sloja. Ovakav koncept je jednostavniji i spriječit će moguće nejasnoće. Zbog svega navedenog izvršna skripta *_8_Labels.sh* je postala nepotrebna te nije dio ažurirane verzije softvera.

Također, sve ažurirane verzije izvršnih skripti i *Python* programa koje smo koristili za izradu karte, a koje će u predstojećim poglavljima biti detaljno opisane, zbog svog opsega neće biti prikazane u radu, već će biti pohranjene i priložene na digitalnom mediju uz ovaj rad.

5.2.1 Postavljanje mjerila, projekcije, koordinatne mreže i jezika karte

U *Terminalu* za radni direktorij postavljamo naš direktorij s preuzetim originalnim softverom, te naerdbom `./_0_PoliticalMap.sh` pozivamo osnovnu izvršnu skriptu. Na početku skripte se naredbama

```
export OSM_DIR=./NEW_MAP
#rm $OSM_DIR -rf
mkdir $OSM_DIR
```

u trenutnom direktoriju kreira novi s imenom *NEW_MAP* kojem pridružujemo varijablu *OSM_DIR* što znači da će taj direktorij biti radni direktorij tijekom cijelog postupka.

Potom se iz originalnog softvera kopiraju direktoriji koji sadrže datoteke s pomoćnim linijama za smještaj imena malih karipskih država, datoteke koje moraju biti na printanoj

verzije karte, datoteke s preddefiniranim stilovima za oblikovanje pojedinih objektnih skupina u *QGIS*-u sljedećim naredbama:

```
cp ./PK/08_additional $OSM_DIR -R
cp ./PK/MapDescription $OSM_DIR -R
cp ./PK/QGISLayerStyleFiles $OSM_DIR -R
```

Sljedeća naredba kopira referentnu kartu iz originalnog softvera u radni direktorij pod nazivom OSM_World_Political_Map.qgs.

```
cp ./PK/Reference_OSMPoliticalMap.qgs $OSM_DIR/OSM_World_Political_Map.qgs
```

U principu je to *QGIS* projekt u kojem će se učitavati svi podaci odn. slojevi koji će nastati u procesu izrade, a na kraju postupka bi tu trebala biti kreirana čitava karta. Potom slijedi definiranje mjerila naredbama:

```
read -p "Set map scale denominator [30000000]: " scale
if [ "$scale" = "" ]
then
    export OSM_SCALE=30000000
else
    export OSM_SCALE=$scale
```

Prvo se traži od korisnika da unese željenu vrijednost mjerila, a ta vrijednost se pridodaje varijabli *OSM_SCALE*. Ukoliko korisnik samo potvrdi, bit će prihvaćena predložena vrijednost „30 000 000“. Na isti način se definira i jezik karte naredbama:

```
read -p "Set map language with two letter code, e.g. hr for Croatian [en]: "
lang
if [ "$lang" = "" ]
then
    export OSM_LANG=en
else
    export OSM_LANG=$lang
```

Korisnik unosi oznaku željenog jezika, a ta vrijednost se pridodaje varijabli *OSM_LANG*. Ukoliko korisnik samo potvrdi, bit će prihvaćena predložena vrijednost „en“ za engleski jezik. Slijedi postavljanje projekcije sljedećim linijama kôda:

```
read -p "Set map projection using PROJ4 syntax [+proj=wintri +ellps=WGS84
+lat_1=50.4597762522]: " proj
```

```
if [ "$proj" = "" ]
then
    export OSM_PROJ="+proj=wintri +ellps=WGS84 +lat_1=50.4597762522"
else
    export OSM_PROJ="$proj"
```

Korisnik unosi parametre željene projekcije koristeći *PROJ* sintaksu i to se pridodaje varijabli *OSM_PROJ*. Ukoliko korisnik samo potvrdi, bit će prihvaćena predložena Winkelova trostruka projekcija s φ_0=50°28′ kao standardnom paralelom preslikavanja.

Naredbama koje slijede postavljamo gustoću koordinatne mreže tj. razmak između paralela i meridijana koji će biti označeni na karti:

```
read -p "Set graticule density [20]: " graticule
if [ "$graticule" = "" ]
then
    export OSM_GRATICULE=20
else
    export OSM_GRATICULE="$graticule"
```

Korisnik unosi vrijednost, a ta vrijednost se pridodaje varijabli *OSM_GRATICULE*. Ukoliko korisnik samo potvrdi, bit će prihvaćena predložena vrijednost „20“ što znači da će se na karti prikazati meridijani i paralele s vrijednostima ±20, ±40, ±60, itd.

Slijedi dio skripte koji nam govori koje procese možemo pokrenuti. Moguće je samo preuzeti i konvertirati OSM *planet* file, ili pokrenuti izradu samo jedne objektne skupine ili svih odjednom za izradu cijele karte. Sljedeća naredba traži od korisnika da unese broj koji se nalazi ispred željenog procesa: `read -p "Enter number for process [10]: " process`. Ukoliko korisnik samo potvrdi, bit će prihvaćen predloženi broj deset koji znači pokretanje izrade cijele karte. Potom će se ispisati popis svih prethodno definiranih parametara. Sljedeća naredba traži od korisnika da provjeri definirane parametre i unosom riječi „start“ pokrene izabrani proces:

```
read -p "Please check it again and confirm by typing 'start': " confirm
```

Tada kreće automatizirani proces izvršavanja skripti, ovisno o izabranom procesu.

Naredba `./osmconf.sh >osmconf.ini` pokreće vrlo jednostavnu opću izvršnu skriptu čiji sadržaj služi samo za kreiranje osmconf.ini konfiguracijske datoteke o kojoj je bilo riječi u

poglavlju 3.4. Ostatak osnovne izvršne skripte *_0_PoliticalMap.sh* zadužen je za preuzimanje i konverziju OSM *planet* datoteke, o čemu će biti govora u sljedećem poglavlju, i za pozivanje izvršne skripte za objektnu skupinu koju želimo kreirati odn. za pozivanje svih izvršnih skripti za objektne skupine ukoliko smo izabrali opciju izrade cijele karte. Proces izrade svake objektne skupine bit će opisan u predstojećim poglavljima.

5.2.2 Preuzimanje OSM datoteke za cijeli planet

Preuzimanje OSM datoteke za cijeli svijet je prvi korak u procesu izrade političke karte svijeta. Postoji više načina preuzimanja te datoteke, kao i više servisa koji je distribuiraju. Na nekim servisima se *Planet* datoteka ažurira na dnevnoj bazi, na nekim na tjednoj, a na nekim čak svake minute. Popis servisa i način distribucije *Planet* datoteke dostupan je na *wiki* stranicama OSM-a (URL 61). S obzirom da ćemo u ovom automatiziranom procesu preuzeti datoteku putem internetske veze, važno nam je da se dohvati PBF format zbog toga što se radi o kompresiranom formatu koji je duplo manji od OSM XML formata. U trenutku kad je preuzeta (23. 07. 2021.) ova datoteka je bila veličine 58,8 GB. Sljedećom naredbom iz osnovne izvršne skripte *_0_PoliticalMap.sh* dohvaća se zadnja dostupna verzija *Planet* datoteke s jednog od službenih servisa za distribuciju.

```
wget
http://ftp5.gwdg.de/pub/misc/openstreetmap/planet.openstreetmap.org/pbf/planet-latest.osm.pbf --output-document=planet.osm.pbf
```

U toj naredbi `wget` dohvaća željeni URL, a `--output-document=` definira ime pod kojim će se pohraniti preuzeta *Planet* datoteka. Potom slijedi konverzija preuzete *planet.osm.pbf* datoteke u o5m format sljedećom naredbom `osmconvert --drop-author planet.osm.pbf -o=planet.osm.o5m` gdje `--drop-author` odbacuje tzv. autorske oznake prikom konverzije, a `-o=planet.osm.o5m` definira naziv izlazne datoteke. Naredba `rm -f planet.osm.pbf` briše PBF datoteku koja nam više nije potrebna, dok opcija `-f` omogućuje da se datoteka izbriše bez potrebe za potvrdom i kad je „read-only“.

Dobivenu *planet.osm.o5m* datoteku veličine 93,1 GB ćemo koristiti u daljnjem postupku izrade karte.

5.2.3 Izrada koordinatne mreže i okvira karte

Izvršna skripta *_1_Graticule.sh* zadužena je za kreiranje okvira karte i koordinatne mreže, a tijekom procesa izvršavanja bit će pozvani i korišteni posebni *Python* program *_1_Graticule.py* i opći *Python* programi *reproject.py* i *set_fields.py*. Ažurirane verzije tih softvera dostupne su na digitalnom mediju priloženom uz ovaj rad.

Na početku svake izvršne skripte naredbom `cd $OSM_DIR` postavljamo naš zadani direktorij *NEW_MAP* kao radni direktorij. Prvom naredbom kreirat ćemo potrebne linije okvira, meridijane, paralele, obratnice i polarnice koristeći se posebnim *Python* programom *_1_Graticule.py* napisanim isključivo za tu svrhu. Bilo bi preopsežno da se u ovom radu detaljno i na razini pojedinih naredbi objašnjava svaki opći i posebni *Python* program, pa ćemo se zadržati na opisnom objašnjenju funkcioniranja tih programa. Tim više što je cijeli koncept izrade karte zamišeljen tako da korisnik parametre potrebne za posebne i opće *Python* programe zadaje odn. mijenja isključivo u izvršnoj skripti pojedine objektne skupine u vidu argumenata naredbenog retka. Dakle, prva naredba je:

```
python3 ../_1_Graticule.py $OSM_GRATICULE 1 $OSM_LANG
```

Ovdje koristimo metodu prenošenja argumenata izravno iz naredbenog retka u *Python* program. Broj argumenata je proizvoljan. U ovom slučaju mi smo u posebni *Python* program *_1_Graticule.py* prenijeli tri argumenta: „$OSM_GRATICULE“, „1“ i „$OSM_LANG“. S obzirom da se unutar *Python* programa *_1_Graticule.py* žele dohvatiti parametri poslani iz naredbenog retka, u *_1_Graticule.py* programu potrebno je na početku programa uključiti paket imena *sys* naredbom `import sys`. Parametri koji su *Python* programu predani preko naredbenoga retka nalaze se u listi *sys.argv*. Treba imati na umu da se na poziciji indeksa 0 nalazi ime *Python* programa koji se pokreće, a na ostalim pozicijama su predani argumenti. Sintaksa dohvaćanja argumenata unutar *Python* programa je: `sys.argv[indeks]`, gdje indeks predstavlja redni broj argumenta koji se želi dohvatiti (URL 62).

Slijedi objašnjenje gore navedene naredbe.

`python3` je naredba koja omogućuje pokretanje *Python* programa direktno iz *Terminala* u interaktivnom načinu rada. `_1_Graticule.py` je naziv posebnog *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `$OSM_GRATICULE` je varijabla sa zadanom

gustoćom koordinatne mreže koju smo definirali na početku izrade karte, a sintaksa dohvaćanja u *_1_Graticule.py Python* programu je sys.argv[1]

`1` je cijelobrojna udaljenost između meridijana i paralela (korak) u stupnjevima (°), a sintaksa dohvaćanja u *_1_Graticule.py Python* programu je sys.argv[2]

`$OSM_LANG` je varijabla sa zadanim jezikom karte koji smo definirali na početku izrade karte, a sintaksa dohvaćanja u *_1_Graticule.py Python* programu je sys.argv[3]

Tako su kreirane vektorske datoteke s okvirom karte (*frame_wgs84.shp*), meridijanima (*meridians_wgs84.shp*), paralelama (*parallels_wgs84.shp*), obratnicama i polarnicama (*tropics_wgs84.shp*) koji su pohranjeni u direktorij *01_graticule*. Navedeni slojevi su u lat/lon projekciji na elipsoidu WGS84 i njih je potrebno reprojicirati u Winkelovu trostruku projekciju što je učinjeno nizom sličnih naredbi, a objasnit ćemo ih na primjeru prve naredbe koja se odnosi na reprojiciranje sloja s meridijanima.

```
python3 ../reproject.py 01_graticule/meridians_wgs84.shp
01_graticule/meridians_final.shp
```

Slijedi objašnjenje naredbe za reprojiciranje meridijana.

`../reproject.py` je naziv općeg *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `01_graticule/meridians_wgs84.shp` je ulazna datoteka koju ćemo reprojicirati, a sintaksa dohvaćanja u *reproject.py Python* programu je sys.argv[1]. `01_graticule/meridians_final.shp` je naziv i lokacija za izlaznu datoteku nakon procesa reprojiciranja, a sintaksa dohvaćanja u *reproject.py Python* programu je sys.argv[2].

Tako je dobiven vektorski sloj s meridijanima u Winkelovoj trostrukoj projekciji naziva *meridians_final.shp*. Sukladno tome dobit ćemo i datoteke *parallels_final.shp*, *tropics_final.shp* i *frame_final.shp*.

Navedeni opći *Python* program *reproject.py* povlači podatak o zadanoj projekciji preko već definirane varijable *OSM_PROJ* i kroz proces konverzije provlači sve objekte pojedine vrste geometrije – MULTIPOLYGON, POLYGON, MULTILINESTRING, LINESTRING i POINT.

Sliijedi brisanje nepotrebnih datoteka naredbom `rm` u kojoj opcija `*` omogućuje brisanje svih datoteka tog naziva bez obzira na ekstenziju.

Naredba za postavljanje potrebnih stupaca u atributnoj tablici za sloj s meridijanima je:

```
python3 ../set_fields.py longitude labels=yes 01_graticule/meridians_final.shp
```

U navedenoj naredbi `../set_fields.py` je naziv općeg *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `longitude` je ime stupaca kojeg želimo zadržati u atributnoj tablici. Sintaksa dohvaćanja željenih imena stupaca u *set_fields.py Python* programu je sys.argv[1]. `labels=yes` govori da želimo dodatne stupce za ručno pomicanje i rotaciju naziva. Sintaksa dohvaćanja u *set_fields.py Python* programu je sys.argv[2]. `01_graticule/meridians_final.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, za koju ćemo napraviti promjene u atributnoj tablici. Sintaksa dohvaćanja u *set_fields.py Python* programu je sys.argv[3].

Na jednaki način se postavlja stupac „latitude“ u atributnoj tablici datoteke *parallels_final.shp* i stupac s nazivima polarnica, obratnica i ekvatora na jeziku karte (zadan varijablom *$OSM_LANG*) u datoteci *tropics_final.shp*.

Navedeni opći *Python* program *set_fields.py* u atributnoj tablici željene datoteke ostavlja samo one stupce koje navedemo na poziciji prvog argumenta i dodaje stupce za ručno pomicanje i rotaciju naziva ako je na mjestu drugog argumenta vrijednost „labels=yes“.

Sada imamo izrađene slojeve za koordinatnu mrežu i okvir karte koji će biti podloga karte na koju ćemo stavljati ostale slojeve. To znači da će okvir karte predstavljati oceane i mora, pa će u konačnom izgledu karte biti svijetlo plave boje.

5.2.4 Izrada obalne crte

Prema OSM *wiki* stranici za obalnu crtu (URL 63) obalna crta se smatra bogatim i složenim okruženjem na rubu morskog okoliša uključujući litice, plaže, luke, rtove, uvale, močvare i otoke. Pri kartiranju obalne crte u OSM-u idealno bi bilo ucrtati prosječnu razinu mora. Sve obalne crte imaju obaveznu oznaku natural=coastline. Prikazivanje obale često je složenije od drugih značajki jer bi praktički cijeli kontikenti trebali biti zatvoreni poligoni. Glavni problem prikazivanja obalne crte iz OSM podataka je to što programi koji se koriste u obradi trebaju savršeno zatvorene poligone. Poligon se može (i obično bi trebao) napraviti od nekog broja petova (engl. *ways*) koji bi se na kraju trebali spojiti kako bi se poligon zatvorio. Nesavršena obala s mnogo malih praznina, obrnutih puteva i drugih nedostataka

uzrokovat će razbijanje poligona, a obalna linija će biti nepravilno prikazana. Zbog toga što je obalna crta korištena u mnogim projektima koji upotrebljavaju OSM podatke, dosta pažnje je posvećeno njenoj konzistentnosti i potencijalni problemi se brzo otklanjaju, a postoje i načini na koji se ona kontrolira. Taj dio je već opisan u poglavlju 2.4.2.

Izvršna skripta *_2_Coastlines.sh* kreirat će sloj s obalnom crtom za cijeli svijet. Postupak započinjemo filtracijom podskupa objekata koji u *Planet* datoteci predstavljaju obalnu crtu:

```
osmfilter ../planet.osm.o5m --keep-nodes= --keep-relations= --keep-ways="natural=coastline" >osm_coastlines.osm
```

U navedenoj naredbi `osmfilter` poziva program *osmfilter*. `../planet.osm.o5m` je ulazna datoteka koju ćemo filtrirati. `--keep-nodes= --keep-relations= --keep-ways="natural=coastline"` je filter koji odbacuje sve čvorove i relacije, a zadržava samo puteve s oznakom „natural= coastline“ jer bi tako u OSM-u trebala bit definirana obalna crta. `>osm_coastlines.osm` kreira izlaznu datoteku osm_coastlines.osm u kojoj će biti samo obalna crta, a za izlazni format je izabran OSM XML jer je nam je taj format nužan za sljedeći korak.

Slijedi konverzija obalne crte iz OSM XML formata u vektorski *Shapefile* format, a to ćemo napraviti korištenjem modula *ogr2ogr* u dva koraka zbog toga što *ogr2ogr* pri čitanju OSM XML datoteka kategorizira značajke (engl. *features*) u pet slojeva (vidi poglavlje 3.3). Manji otoci koji su digitalizirani i zatvoreni s manje od 2000 čvorova su poligoni i biti će u sloju „multipolygons“. Ostali dijelovi obalne crte će biti u sloju „lines“.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql "SELECT osm_id FROM lines" -overwrite -skipfailures -nlt LINESTRING -lco ENCODING=UTF-8 02_coastlines osm_coastlines.osm
```

U navedenoj naredbi `ogr2ogr` poziva *ogr2ogr* modul. `--config OSM_CONFIG_FILE` je opcija koja poziva na korištenje konfiguracijske datoteke i aktiviranje alternativne putanje do nje. `../osmconf.ini` je naziv konfiguracijske datoteke i njena lokacija u odnosu na radni direktorij. `-f "ESRI Shapefile"` definira ESRI Shapefile (*.shp) kao format za izlaznu datoteku. `-sql "SELECT osm_id FROM lines"` je *sql* upit koji iz sloja „lines“ povlači sve objekte i za njih preuzima vrijednosti (engl. *value*) onih oznaka (engl. *tags*) koje imaju ključ (engl. *key*) „osm_id“; `-overwrite` briše izlazni sloj (ukoliko postoji) i ponovno ga kreira

praznog. `-skipfailures` nastavlja proces i nakon neuspjeha, preskačući neuspješnu značajku. `-nlt LINESTRING` definira LINESTRING kao vrstu geometrije za kreirani sloj. `-lco ENCODING=UTF-8` definira UTF-8 kao zadani encoding. `02_coastlines` definira naziv i lokaciju izlaznog direktorija u odnosu na radni direktorij. `osm_coastlines.osm` je naziv i lokacija ulazne datoteke u odnosu na radni direktorij.

Na identičan način se sljedećom naredbom iz sloja „multipolygons“ dobije vektorska datoteka s multipoligonima koji predstavljaju obalnu crtu:

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql "SELECT osm_id FROM multipolygons" -overwrite -skipfailures -nlt LINESTRING -lco ENCODING=UTF-8 02_coastlines osm_coastlines.osm
```

Sada smo dobili dvije *Shapefile* datoteke koje sadrže sve obalne crte unešene u OSM i potrebno ih je spojiti u jednu datoteku. Podaci iz datoteke *multipolygons.shp* će se prepisati u datoteku *lines.shp* naredbom:

```
ogr2ogr -f "ESRI Shapefile" -append -update ./02_coastlines/lines.shp ./02_coastlines/multipolygons.shp
```

U navedenoj naredbi `-append` dodaje podatke u postojeći sloj umjesto stvaranja novog. `-update` otvara postojeću izlaznu datoteku u načinu ažuriranja (engl. *update mode*) umjesto pokušaja stvaranja nove. `./02_coastlines/lines.shp` definira naziv i lokaciju izlazne datoteke u odnosu na radni direktorij. `./02_coastlines/multipolygons.shp` je naziv i lokacija ulazne datoteke u odnosu na radni direktorij.

U ovom dijelu izvršne skripte ćemo koristit *GRASS GIS* izravno iz komandne linije kako bi iz podataka u *lines.shp* datoteci dobili poligone s obzirom da su dobivene obale trenutačno razložene na linijske segmente. Iako i org2ogr ima tu mogućnost, *GRASS GIS* ima izuzetno robusne alate. Naprimjer, ako imamo situaciju da na nekom dijelu jedan segment (linija) ide u jednom smjeru, a da se na njega „spaja“ drugi segment koji ide u drugom smjeru, *ogr2ogr* tada neće zatvoriti poligon dok *GRASS GIS* to prepoznaje kao zatvoreni poligon.

Prvo iz *Terminala* pokrećemo opću izvršnu skriptu *grass_wgs84_location.sh* kojom definiramo lat/lon projekciju i kreiramo potrebne radne direktorije. Naredbom `echo` izravno u *Terminalu* kreiramo posebnu izvršnu skriptu *grass_coastlines.sh*, a zatim niz naredbi

```
chmod u+x grass_coastlines.sh
```

```
export GRASS_BATCH_JOB=./grass_coastlines.sh
grass -text "$PWD/grassdata/wgs84/data"
```

daje dopuštenje korisniku za izvršavanje kreirane skripte *grass_coastlines.sh*, otvara skriptu kao posao u *GRASS GIS*-u i definira radni direktorij. Slijedi popis naredbi, i njihovih objašnjenja, koje će se izvršiti u *GRASS GIS*-u u sklopu posebne izvršne skripte *grass_coastlines.sh*.

```
v.in.ogr input=./02_coastlines/lines.shp layer=lines output=coastlines --
overwrite -c -t
```

`v.in.ogr` je naredba koja uvozi vektorske podatke u *GRASS GIS*-ovu vektorsku mapu koristeći OGR biblioteku. `input=./02_coastlines/lines.shp` je naziv OGR izvora podataka za uvoz, u ovom slučaju direktorij s *lines.shp* datotekom koju ćemo koristiti u *GRASS GIS*-u. `layer=lines` je naziv sloja koji će se koristiti. `output=coastlines` je naziv izlazne mape koja se pohranjuje u *GRASS GIS* radni direktorij. `--overwrite` dopušta izlaznim datotekama da pišu preko postojeće datoteke (ukoliko postoji). Kad je uključena opcija `-c` program neće probati „čistiti“ poligone i složiti topologiju nego će samo učitati podatke kakvi jesu. Kad ova opcija ne bi bila uključena, program bi automatski pokušao stvoriti topologiju od linija iz učitane ulazne datoteke *lines.shp* i svtorio bi ogroman broj malih poligona na način da bi povezao prvu i zadnju točku svake linije i tako kreirao poligone. Kad je uključena opcija `-t` program neće kreirati atributnu tablicu, a za obalnu crtu nam ne trebaju nikakvi atributi nego samo geometrija.

```
v.build.polylines input=coastlines output=coastlines_polygons cats=first
type=line
```

`v.build.polylines` je naredba koja izrađuje polilinije od linija ili granica u vektorskoj mapi. Liniju definira jedan početni čvor, jedan krajnji čvor i bilo koji broj vrhova između početnog i završnog čvora. Jedna polilinija se može kreirati samo do grananja linije, a u ovom slučaju ne bi trebalo biti takvih situacija. To automatski znači da će sve linije koje se međusobno spajaju u čvorovima biti povezane u jednu poliliniju i tako uspješno zatvoriti poligon. `input=coastlines` je naziv ulazne vektorske mape koja se ubacuje iz *GRASS GIS* radnog direktorija, a koju smo kreirali u prošlom koraku. `output=coastlines_polygons` je naziv izlazne vektorske mape koja se pohranjuje u *GRASS GIS* radni direktorij.

`cats=first` dodijeljuje broj kategorije prve linije cijeloj poliliniji. `type=line` određuje da će jedino linije ući u proces izrade polilinija.

```
v.out.ogr output_type=boundary input=coastlines_polygons type=line
format=ESRI_Shapefile output=./02_coastlines --overwrite
```

`v.out.ogr` pretvara sloj *GRASS GIS* vektorske mape u bilo koji od podržanih OGR vektorskih formata. `output_type=boundary` izvozi linije kao poligone. `input=coastlines_polygons` je naziv ulazne vektorske mape za izvoz. `type=line` određuje vrstu značajki koje želimo izvesti. Moguće opcije su: point, line, boundary, centroid, area, face, kernel, auto. `format=ESRI_Shapefile` definira izlazni format. Mi smo izabrali *Shapefile*. `output=./02_coastlines` je naziv izlazne datoteke i lokacija na koju će se spremiti. `--overwrite` dopušta izlaznim datotekama da pišu preko postojeće datoteke (ukoliko postoji).

Ovdje završava dio s naredbama koje su se izvršavale u *GRASS GIS*-u i daljnje naredbe se izvršavaju u naredbenom retku *Terminala*. Za izlazak iz *GRASS GIS*-a koristimo naredbu

`unset GRASS_BATCH_JOB`. Zatim se brišu nepotrebni direktoriji i datoteke kreirane tijekom procesa u *GRASS GIS*-u.

Kao što je to Jogun (2016) opisao u svom radu, Kaspijsko jezero tretira se kao more, i zbog toga se i granice država koje imaju izlaz na njega trebaju izrezati na obalnoj crti jezera. Inače je za granice na jezerima kod političkih karti uobičajeno da se državna granica crta iznad sloja s jezerima, te tako pokazuje kako je površina jezera podijeljena između susjednih država. Trenutačno mi u istom sloju imamo jedan poligon koji predstavlja Euroaziju i drugi koji predstavlja Kaspijsko jezero. Većina alata ne može izvršiti proces izrezivanja ako su objekti koji bi se trebali preklopiti iz istog sloja. Zbog ovog problema potrebno je kreirati multipoligon od Euroazije koji će na mjestu Kaspiskog jezera imati unutarnji prsten. Stoga je napisan poseban *Python* program *_2_1_caspian_sea.py* koji će prestrukturirati podatak da Kaspijsko jezero ne bude objekt za sebe, već da njegov poligon postane druga zatvorena polilinija u poligonu koji definira Euroaziju, što po WKT (engl. *Well-known text*) sintaksi predstavlja unutarnji prsten poligona (URL 64). Navedeni *Python* program pokrećemo sljedećom naredbom:

```
python3 ../_2_1_caspian_sea.py ./02_coastlines/coastlines_polygons.shp
```

S obzirom da svaka generalizacija može dovesti do topoloških problema, tek sad kad imamo topološki korektne objekte možemo raditi nešto po pitanju same generalizacije. Prva generalizacija bit će pojednostavljenje koristeći *ogr2ogr* ugrađenu opciju „simplify“ koja smanjuje broj točaka definirajući toleranciju udaljenosti za proces pojednostavljivanja. Ono što nam je bitno je da korišteni algoritam čuva topologiju po značajci tijekom samog procesa. Naredba je sljedeća:

```
ogr2ogr -f "ESRI Shapefile" -lco ENCODING=UTF-8 -simplify $(echo "scale=6; $OSM_SCALE/1000/6370000" | bc) ./02_coastlines/coastlines_simplified.shp ./02_coastlines/coastlines_polygons.shp
```

Opcija pojednostavljenja `-simplify $(echo "scale=6; $OSM_SCALE/1000/6370000" | bc)` je definirana izrazom u zagradama i računa se kalkulatorom naredbenog retka koji pozivamo izrazom „| bc“, a „scale=6“ definira broj decimalnih mjesta. S obzirom da smo još uvijek u geografskim koordinatama u opciji `simplify` se pri zadavanju tolerancije zadano mjerilo `$OSM_SCALE` dijeli s polumjerom Zemlje da se dobije lučna mjera. Potom se ta vrijednost umanjuje 1000 puta zbog toga što se u ovom procesu pojednostavljenja koristi Douglas-Peucker algoritam koji radi oštre lomove na linijama, a mi to želimo ublažiti kako bi dobili glatke linije primjerene kartografskom prikazu. Na taj način će opcija `simplify` obaviti smanjivanje broja točaka bez da utječe na konačnu vizualizaciju linije. Tako pojednostavljen sloj s obalnim crtama sprema se u datoteku *coastlines_simplified.shp*.

Slijedi proširenje Antartike do južnog pola. *OpenStreetMap* koristi sferoidnu Mercatorovu projekcije (Web-Mercatorova), pa prikazivanje polova nije moguće. Nama to izostavljeno područje ipak treba, pa se koristi za to napisani posebni *Python* program *_2_2_antarctica.py*. Program spušta sve točke koje se nalaze na 85° južne geografske širine ili južnije na 90° južne geografske širine. Ulazna datoteka je *coastlines_simplified.shp* i u njoj će bit pohranjene promjene, a naredba glasi:

```
python3 ../_2_2_antarctica.py ./02_coastlines/coastlines_simplified.shp
```

Dobiveni sloj je potrebno reprojicirati u Winkelovu trostruku projekciju što je učinjeno već spomenutim općim *Python* programom *reproject.py*. Naredba je:

```
python3 ../reproject.py ./02_coastlines/coastlines_simplified.shp
./02_coastlines/coastlines_winkel.shp
```

U navedenoj naredbi `../reproject.py` je naziv općeg *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `./02_coastlines/coastlines_simplified.shp` je ulazna datoteka koju ćemo reprojicirati. `./02_coastlines/coastlines_winkel.shp` je naziv i lokacija za izlaznu datoteku nakon procesa reprojiciranja.

S tom datotekom ulazimo u konačnu generalizaciju koju ćemo provesti općim programom *generalize.py* koji u svom prvom dijelu sadrži softversku implementaciju algoritma za kartografsku generalizaciju linija sa svojstvom čuvanja površina. Taj algoritam daje generalizirane linije koje se općenito sastoje od podskupa polaznih točaka i skupa novih točaka, pa se zbog toga algoritam može smatrati kombinacijom pojednostavljenja i izglađivanja (Tutić i Lapaine, 2009). U drugom dijelu općeg *Python* programa se izvršava sama generalizacija, uključujući zagrađivanje linija. U izvršnoj skripti naredba za taj postupak glasi:

```
python3 ../generalize.py $OSM_SCALE 0 ./02_coastlines/coastlines_winkel.shp
./02_coastlines/coastlines_clip.shp
```

U navedenoj naredbi `../generalize.py` je naziv općeg *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `$OSM_SCALE` je varijabla koja za željeno mjerilo povlači podatak koji smo unijeli na početku pri postavljanju parametara karte. `0` je granična vrijednost, u milimetrima kvadratnim u mjerilu karte, ispod koje će svi objekti biti izbrisani. Vrijednost je postavljena na nulu jer ne želimo da se sad izbrišu ikakvi objekti.

Tako je dobivena datoteka *coastlines_clip.shp* sa zaglađenim linijama obalne crte će se koristiti i pri izradi nekih drugih objektnih skupina, pa ćemo je sačuvati u ovom obliku, a za potrebe ove izvršne skritpe napraviti napraviti njenu kopiju s imenom *coastlines_filtered.shp* naredbom:

```
ogr2ogr -f "ESRI Shapefile" -lco ENCODING=UTF-8
./02_coastlines/coastlines_filtered.shp ./02_coastlines/coastlines_clip.shp
```

S tom *coastlines_filtered.shp* datotekom ulazimo u postupak filtriranja površina manjih od 0,8 mm^2 na karti. Postupak će se izvršiti koristeći *ogr2ogr* modul naredbom:

```
python3 ../simplify_polygon.py 0 $[$OSM_SCALE * $OSM_SCALE / 1000000 * 8 / 10]
./02_coastlines/coastlines_filtered.shp
```

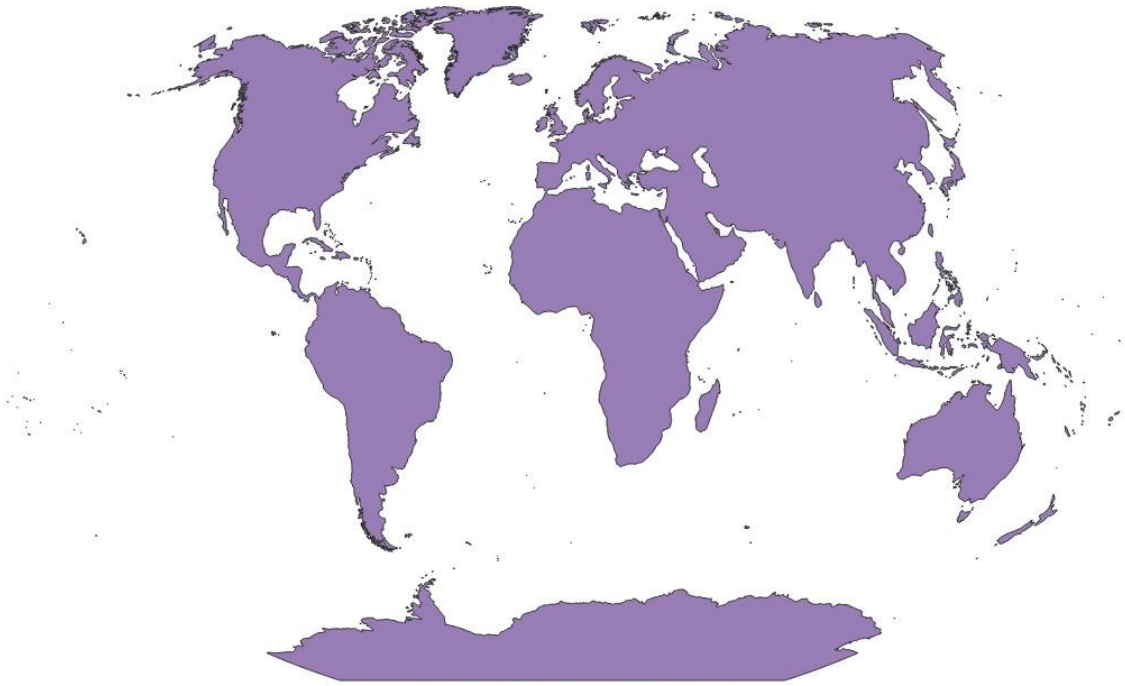
U navedenoj naredbi `../simplify_polygon.py` je naziv posebnog *Python* programa kojeg pokrećemo i lokacija na kojoj se nalazi u odnosu na radni direktorij. `0` je vrijednost tolerancije za duljine linija u procesu pojednostavljenja, a ukoliko je vrijednost nula, tada se pojednostavljanje neće izvršiti. Sintaksa dohvaćanja u *simplify_polygon.py Python* programu je sys.argv[1]. `$[$OSM_SCALE * $OSM_SCALE / 1000000 * 8 / 10]` je varijabla koja osigurava da se izbrišu svi poligoni manji od 0,8 mm^2 na karti. Zadana je kao funkcija mjerila koje smo definirali na početku izrade karte, a sintaksa dohvaćanja je sys.argv[2].

Opći *Python* program *simplify_polygon.py* je jednostavan program za promjenu geometrije koji na temelju zadanih parametara može brisati poligone manje od zadane vrijednosti i pojednostavljivati geometriju zadržavajući topologiju.

Budući da je rezultat filtriranja zapisan u istu datoteku, ona je i dalje bila jednake veličine kao prije procesa filtracije. Zato je dobivena *coastlines_filtered.shp* datoteka kopirana korištenjem *ogr2ogr* modula u datoteku *coastlines_final.shp* sljedećom naredbom:

```
ogr2ogr -f "ESRI Shapefile" -lco ENCODING=UTF-8
./02_coastlines/coastlines_final.shp ./02_coastlines/coastlines_filtered.shp
```

Sad još samo preostaje izbrisati datoteke koje više nisu potrebne i time je izvršna skripta *_2_ Coastlines.sh* stigla do kraja, a kao rezultat imamo sloj s obalnom crtom za cijeli svijet u Winkelovoj trostrukoj projekciji pohranjen u vektorsku datoteku *coastlines_final.shp* (Slika 5.1.) koja će se koristiti u konačnom izgledu karte. Ukupno su dobivena 832 objekta (poligona), a datoteka je veličine 2,83 MB.



Slika 5.1. Konačni sloj s obalnom crtom u Winkelovoj trostrukoj projekciji (coastlines_final)

5.2.5 Izrada sloja s političko-teritorijalnim entitetima

U ovom poglavlju biti će obrađena izrada najvažnijeg sloja na političkim kartama, onog s političko-teritorijalnim entitetima. Prema službenim smjernicama OSM-a, nezavisne države bi za svoje granice trebale imati kombinaciju dviju oznaka: boundary=administrative i admin_level=2. Postoje neke kontroverze oko toga što je točno država. Da ne bi bilo nedoumica, samo se politički subjekti navedeni u standardu ISO 3166 smatraju zemljama u OSM-u. Postoje i neka ovisna područja i posebna područja od geografskog interesa koja imaju vlastiti ISO 3166-1 kôd, ali nisu država (URL 65). Među mjestima koja nemaju dodijeljene ISO kôdove, a smatraju se državama spada Kosovo. S obzirom na to da su autori ovog softvera kojeg koristimo uočili kako dolazi do miješanja administrativnih razina u nekim državama, odlučili su se pri izdvajanju političko-teritorijalnih entiteta koristiti upravo ISO 3166-1 kôd kao njihov identifikator (Jogun, 2016). U bazi OSM-a za taj kôd se koriste oznake „ISO3166-1:alpha2“ i „ISO3166-1“.

Izdvajanje granica i poligona političko-teritorijalnih entiteta obavit će se pokretanjem izvršne skripte *_3_Countries.sh*.

Prvi dio skripte filtrira države iz OSM *Planet* datoteke.

Naredba `python3 ../_3_1_filter_countries.py >filter_countries.sh` pokreće poseban *Python* program *_3_1_filter_countries.py*, a rezultat izvršavanja tog programa bit će spremljen u novu *shell* skriptu *filter_countries.sh*. Navedeni *Python* program služi za dohvaćanje CSV datoteke s popisom imena država i pripadajućih ISO 3166-1 kôdova (URL 66). Na tu listu, nazvanu „data“, je naredbom `data.append(list(['Kosovo','XK']))` dodano Kosovo s kôdom „XK“. Ostatak programa naredbom `print` ispisuje sadržaj dugačkog filtera koji će se zapisati u *shell* skriptu *filter_countries.sh*. Filter je napisan tako da se programom *osmfilter* iz cijele *Planet* datoteke zadržavaju samo relacije, i to one koje za oznake „ISO3166-1:alpha2“ i „ISO3166-1“ imaju vrijednost nekog od dvoslovnih ISO 3166-1 kôdova iz preuzete, a potim i proširene liste. Zadržane relacije će se pohraniti u datoteku *osm_countries.osm*.

Naredba `chmod +x filter_countries.sh` daje dozvolu da navedena skripta s filterom bude izvršiva (engl. *execution permission*), a naredba `./filter_countries.sh` ju pokreće. Nakon što je skripta izvršena, a filtracijom dobivena datoteka *osm_countries.osm* naredba `rm filter_countries.sh` briše izvršenu skriptu.

Sljedećom naredbom se iz sloja „multipolygons“ dobije vektorska *Shapefile* datoteka s multipoligonima koji predstavljaju države, a dobivena datoteka nosit će naziv *multipolygons.shp* po nazivu sloja.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"SELECT * FROM multipolygons WHERE (\"ISO3166-1_alpha2\"!='' OR \"ISO3166-
1\"!='') AND admin_level='2'" -overwrite -skipfailures -nlt MULTIPOLYGON -lco
ENCODING=UTF-8 03_countries osm_countries.osm
```

U navedenoj naredbi `ogr2ogr` poziva *ogr2ogr* modul. `--config OSM_CONFIG_FILE` poziva na korištenje konfiguracijske datoteke i aktiviranje alternativne putanje do nje. `../osmconf.ini` je naziv konfiguracijske datoteke i njena lokacija u odnosu na radni direktorij. `-f "ESRI Shapefile"` definira ESRI Shapefile kao format za izlaznu datoteku. `-sql "SELECT * FROM multipolygons WHERE (\"ISO3166-1_alpha2\"!='' OR \"ISO3166-1\"!='') AND admin_level='2'"` povlači sve objekte iz sloja „multipolygons“ koji imaju unešenu vrijednost za oznake s ključem „ISO3166-1_alpha2“ ili „ISO3166-1“ i oznaku „admin_level='2'“. `-nlt MULTIPOLYGON` definira MULTIPOLYGON kao vrstu geometrije za kreirani sloj. `03_countries` definira naziv i lokaciju izlaznog direktorija u odnosu na radni

direktorij. `osm_countries.osm` je naziv i lokacija ulazne datoteke u odnosu na radni direktorij.

Sada dolazi dio skripte koji rješava iznimke koje nisu obuhvaćene prethodnim postupkom. Probat će se vidjeti postoje li multipoligoni u datoteci „osm_countries.osm“ koji imaju vrijednost nekog od dvoslovnih ISO 3166-1 kôdova koji nisu pronađeni u atributnoj tablici datoteke *multipolygons.shp,* a koji nemaju oznaku „admin_level=2“. To bi značilo da su takvi poligoni države, ali nemaju vrijednosti ključa „admin_level“ u skladu s preporukama OSM-a. Unutar posebnog *Python* programa *_3_2_filter_missing_iso.py* stvorit će se lista *iso* u kojoj će bit pohranjeni svi ISO 3166-1 kôdovi koje imaju objekti u datoteci *multipolygons.shp*. Zatim će ta lista biti uspoređena s listom *data* u kojoj je popis svih država koje imaju ISO 3166-1 kôd (proširen s Kosovom) i u novu listu *missing* će bit pohranjeni oni kôdovi koji nisu pronađeni u atributnoj tablici datoteke *multipolygons.shp*. Potom naredba `print` koristeći listu *missing* ispisuje sadržaj *ogr2ogr* naredbe koji će se zapisati u novu *shell* skriptu *missing_iso.sh*. Naredba sadrži *sql* upit koji iz datoteke *osm_countries.osm* povlači sve poligone koji za *ključ* „boundary“ imaju vrijednosti „administrative“ ili „territorial“ ili „region“ i koji imaju neki od ISO 3166-1 kôdova s liste *missing* tj. one koji nisu pronađeni u ranije dobivenoj *multipolygons.shp* datoteci. Takvi poligoni će se u direktoriju 03_countries_ex pohraniti u datoteci *multipolygons.shp,* a nepotrebna *shell* skripta *missing_iso.sh* biti će obrisana. Opisani postupak se izvršava putem naredbi nabrojanih u nastavku teksta.

```
python3 ../_3_2_filter_missing_iso.py 03_countries/multipolygons.shp
>missing_iso.sh
chmod +x missing_iso.sh
./missing_iso.sh
rm missing_iso.sh
```

Sada u različitim direktorijima imamo dvije *multipolygons.shp* datoteke koje ćemo spojiti na način da ćemo onoj iz direktorija *03_countries* dodati sadržaj one iz direktorija *03_countries_ex* koristeći se *ogr2ogr* modulom.

```
ogr2ogr -f "ESRI Shapefile" -update -append 03_countries/multipolygons.shp
03_countries_ex/multipolygons.shp
```

Potom ćemo preimenovati datoteku ponovo koristeći *ogr2ogr* modul koji zapravo kopira datoteku pod novim imenom *countries_polygons.shp*.

```
ogr2ogr -f "ESRI Shapefile" -lco ENCODING=UTF-8
03_countries/countries_polygons.shp 03_countries/multipolygons.shp
```

Naredbama `rm 03_countries/multipolygons.* -f` i `rm 03_countries_ex -rf` brišemo datoteke i direktorije koji nam više nisu potrebni.

U ovom trenutku i dalje je moguće da neke države ipak nisu u *countries_polygons.shp* datoteci, npr. zbog pogrešaka u podacima OSM-a poput neispravno formiranih poligona zbog nepravilnosti na granicama. Stoga će se posebnim *Python* programom *_3_3_list_missing_iso.py* sastaviti lista ISO kôdova tih država i spremiti ju u datoteku *COUNTRY_ERRORS.log*. Proces je vrlo sličan onom u *Python* programu *_3_2_filter_missing_iso.py*.

```
python3 ../_3_3_list_missing_iso.py 03_countries/countries_polygons.shp
>../COUNTRY_ERRORS.log
```

Ovaj postupak je potreban isključivo zbog nekoherentnosti podataka u OSM-u. Prilikom ispitivanja softvera, ova izvršna skripta je pokrenuta dva puta u razmaku od dva mjeseca i oba puta je bio različit broj kôdova na listi u *COUNTRY_ERRORS.log*. Kao kontrola potpunosti liste može se zbrojiti broj kôdova s liste s brojem država u datoteci *countries_polygons.shp*. Ukupan broj bi trebao biti 250 kao u preuzetoj i nadopunjenoj CSV datoteci. Međutim, nema garancije da su sve države u datoteci *countries_polygons.shp* u cijelosti izdvojene iz OSM-podataka, pa u ovom dijelu izvršna skripta zahtjeva od korisnika da vizualno provjeri podatke iz datoteke *countries_polygons.shp* i da listu *COUNTRY_ERRORS.log* ručnim unosom nadopuni s ISO kôdom država čija geometrija nije u cijelosti ispravno izdvojena. Kad smo mi testirali ovu izvršnu skriptu, nije bilo potrebe za ovim korakom.

Ponovo ćemo iz *Planet* datoteke filtrirati države, ali ovaj put one koje su na popisu u datoteci *COUNTRY_ERRORS.log*. Potom ćemo probati izdvojiti geometriju tih država koristeći alat *osmtogeojson* (URL 67) koji se smatra robusnijim zbog boljeg otkrivanja poligona iz OSM podataka i odgovarajuće podršku za multipoligone.

Za kreiranje potrebnog filtera koristimo posebni *Python* program *_3_4_osmtogeojson_for_exceptions.py* koji će ISO kôdove iz prethodno stvorene datoteke *COUNTRY_ERRORS.log* implementirati kao ključ za filtriranje relacija koje se žele zadržati. Tako kreiran sadržaj za *osmfilter* naredbu se pohranjuje u posebnu izvršnu skriptu *country_exceptions.sh* koja se zatim izvršava.

```
python3 ../_3_4_osmtogeojson_for_exceptions.py ../COUNTRY_ERRORS.log
>country_exceptions.sh
chmod +x country_exceptions.sh
./country_exceptions.sh
```

Izlazni rezultati će biti spremljeni u *osm_countries_add.osm* datoteku.

Slijedi nam kreiranje poligona korištenjem spomenutog *osmtogeojson* alata. Sintaksa naredbe je, u skladu s preporukama autora (URL 68), prilagođena korištenju većih datoteka.

```
node --max_old_space_size=3072 `which osmtogeojson` osm_countries_add.osm >
osm_countries_add.geojson
```

Tako dobivenu izlaznu datoteku *osm_countries_add.geojson* je potrebno konvertirati korrištenjem *ogr2ogr* modula u *Shapefile* format kako bi je mogli koristiti dalje u procesu.

```
ogr2ogr -f "ESRI Shapefile" -sql "SELECT
name,\"name:en\",\"name:hr\",population,type,place,\"ISO3166-
1:alpha2\",\"ISO3166-1\",admin_level,boundary FROM osm_countries_add WHERE
\"ISO3166-1\"!='' OR \"ISO3166-1:alpha2\"!='" -overwrite -skipfailures -nlt
MULTIPOLYGON -lco ENCODING=UTF-8 03_countries_add osm_countries_add.geojson
```

U navedenoj naredbi *sql* upit osigurava da se prilikon konverzije zadrže samo objekti koji imaju neku vrijednost u *ključu* „ISO3166-1“ ili „ISO3166-1:alpha2“ i za te objekte kopira sadržaj stupaca name, name:en, name:hr, population, type, place, ISO3166-1:alpha2, ISO3166-1, admin_level i boundary. Zadana vrsta geometrije za izlazni sloj je MULTIPOLYGON, a izlazna datoteka će imati naziv po ulaznoj datoteci i spremit će se u direktorij *03_countries_add*.

U ovakvom procesu postoji velika mogućnost da se poligoni pojedinih država nalaze i u datoteci *osm_countries_add.shp* i u datoteci *countries_polygons.shp* pa je potrebno napraviti čišćenje takvih duplih objekata. Za to koristimo poseban *Python* program i naredbu ispod.

```
python3 ../_3_5_clean_countries.py ../COUNTRY_ERRORS.log
03_countries/countries_polygons.shp 03_countries_add/osm_countries_add.shp
```

Python program *_3_5_clean_countries.py* prvo iz datoteke *countries_polygons.shp* briše one objekte koji se nalaze na popisu COUNTRY_ERRORS.log datoteke budući da su ti objekti prikazani i u *osm_countries_add.shp* datoteci, a smatraju se kvalitetnijima jer su dobiveni korištenjem *osmtogeojson* alata. Potom se za obje datoteke radi prepisivanje ISO kôdova u slučaju da je za neki objekt unešena vrijednost za *oznaku* „ISO3166-1“, a nije za „ISO3166-1:alpha2“ i obrnuto.

Nakon čišćenja duplih objekata, možemo spojiti ove dvije datoteke. I ovaj put ćemo koristiti *ogr2ogr* modul i opcije `-update` i `-append` uz pomoć kojih ćemo sadržaj datoteke *osm_countries_add.shp* dodati u datoteku *countries_polygons.shp*.

```
ogr2ogr -f "ESRI Shapefile" -update -append 03_countries/countries_polygons.shp
03_countries_add/osm_countries_add.shp
```

Potom brišemo sve nepotrebne datoteke i direktorije koristeći sljedeće naredbe: `rm 03_countries_add -rf`, `rm osm_countries_add.osm -f`, `rm country_exceptions.sh -f`, `rm osm_countries_add.geojson`.

Kao i kod kreiranja obalne crte, i ovdje je potrebno produžiti poligon antartike do Južnog pola, a to ćemo uraditi koristeći indentični *Python* program *_2_2_antarctica.py*. Ulazni i izlazni sloj za naredbu je *countries_polygons.shp*.

```
python3 ../_2_2_antarctica.py 03_countries/countries_polygons.shp
```

Konačno možemo reprojicirati sloj u Winkelovu trostruku projekciju. Izlazna datoteka je *countries_winkel.shp*.

```
python3 ../reproject.py 03_countries/countries_polygons.shp
03_countries/countries_winkel.shp
```

Dolazimo do zahtjevnog dijela izvršne skripte. Naime, naši poligoni s državama obuhvaćaju i kopnene i morske djelove što nije prikladno za prikaz na političkoj karti. Stoga je potrebno izrezati poligone političko-teritorijalnih entiteta tako da završavaju na granici s morem. Za to će nam poslužiti datoteka *coastlines_clip.shp* sa zaglađenim linijama obalne crte.

Prvo iz *Terminala* pokrećemo opću izvršnu skriptu *grass_xy_location.sh* s kojom postavljamo privremenu lokaciju u *GRASS GIS*-u i kreiramo potrebne radne direktorije.

Naredbom `echo` izravno u *Terminalu* kreiramo posebnu izvršnu skriptu *grass_countries.sh*, a zatim niz naredbi daje dopuštenje korisniku za izvršavanje kreirane skripte, otvara skriptu kao posao u *GRASS GIS*-u i definira radni direktorij.

```
chmod u+x grass_countries.sh
export GRASS_BATCH_JOB=./grass_countries.sh
grass -text "$PWD/grassdata/xy/data"
```

Slijedi popis naredbi, i njihovih objašnjenja, koje će se izvršiti u *GRASS GIS*-u u sklopu posebne izvršne skripte *grass_countries.sh*.

```
v.in.ogr -o input=./02_coastlines/coastlines_clip.shp output=coastlines_clip
min_area=0.0001 snap=1e-6 --overwrite
```

`v.in.ogr` je naredba koja uvozi vektorske podatke u *GRASS GIS*-ovu vektorsku mapu koristeći OGR biblioteku. `-o` je opcija za zaobilaženje provjere projekcije. Program pretpostavlja da skup podataka ima istu projekciju kao trenutačna lokacija. `input=./02_coastlines/coastlines_clip.shp` je naziv OGR izvora podataka za uvoz, u ovom slučaju direktorij s *coastlines_clip.shp* datotekom koju ćemo koristiti u *GRASS GIS-u*. `output=coastlines_clip` je naziv izlazne mape koja se pohranjuje u *GRASS GIS* radni direktorij. `min_area=0.0001` je minimalna veličina površine za uvoz u kvadratnim metrima, zanemaruju se manja područja i otoci. `snap=1e-6` je prag pucanja za granice u jedinicama karte.

```
v.in.ogr -o input=./03_countries/countries_winkel.shp output=countries_winkel
min_area=0.0001 snap=1e-6 --overwrite
```

`input=./03_countries/countries_winkel.shp` je naziv OGR izvora podataka za uvoz, u ovom slučaju direktorij s `countries_winkel.shp` datotekom koju ćemo koristiti u *GRASS GIS-u*. `output=countries_winkel` je naziv izlazne mape koja se pohranjuje u *GRASS GIS* radni direktorij.

```
v.overlay ainput=countries_winkel atype=area alayer=1 binput=coastlines_clip
btype=area blayer=1 output=countries_clip operator=and olayer=0,1,0 snap=1e-8 -
-overwrite
```

`v.overlay` je naredba koja preklapa dvije vektorske mape i koja nudi isječak, presjek, razliku, simetričnu razliku i korištenje operatora. `ainput=countries_winkel` zadaje ime prve ulazne vektorske mape (A), u našem slučaju to je *countries_winkel*. `atype=area` definira

vrstu značajke prve ulazne mape (A), u našem slučaju to je površina. `alayer=1` je naredba koja definira broj ili naziv sloja za vektorsku mapu A. `binput=coastlines_clip` zadaje ime druge ulazne vektorske mape (B), u našem slučaju to je *coastlines_clip*. `btype=area` definira vrstu značajke druge ulazne mape (B), u našem slučaju to je površina. `blayer=1` je naredba koja definira broj ili naziv sloja za vektorsku mapu B. `output=countries_clip` zadaje ime izlazne vektorske mape. `operator=and` definira značajke zapisane na izlaznu vektorsku mapu, u ovom slučaju to je presjek. `olayer=0,1,0` definira izlazni sloj za novu kategoriju, ainput i binput. Vrijednost je nula ako nije zadano. U našem slučaju izlazni sloj će biti *countries_clip* što je definirano brojem jedan na srednjem mjestu koje se odnosi na ainput. `snap=1e-8` je prag pucanja za granice u jedinicama karte.

```
v.build map=countries_clip option=build --overwrite
```

`v.build` je naredba koja kreira topologiju za vektorsku mapu. `map=countries_clip` zadaje ime ulazne vektorske mape, u našem slučaju to je *countries_clip*. `option=build` definira izgradnju topologije.

```
v.out.ogr -c input=countries_clip layer=1 type=area output=./03_countries/clip
format=ESRI_Shapefile lco=\"ENCODING=UTF-8\"" >grass_countries.sh
```

`v.out.ogr` naredba koja pretvara sloj *GRASS GIS* vektorske mape u bilo koji od podržanih OGR vektorskih formata. `-c` je opcija koja osigurava da se izvoze i značajke bez kategorije. U suprotnom se izvoze samo značajke s kategorijom. `input=countries_clip` je naziv ulazne vektorske mape za izvoz. `layer=1` je naziv ili broj sloja ulazne vektorske mape za izvoz. Vektorske značajke mogu imati vrijednosti kategorije u različitim slojevima. Ovaj broj određuje koji će sloj koristiti. Kada se koristi s izravnim OGR pristupom, ovo je naziv sloja. Zadana vrijednost je 1. `type=area` je vrsta značajki koje želimo izvesti, u našem slučaju to je površina. `output=./03_countries/clip` je naziv izlazne datoteke i lokacija na koju će se spremiti.

Ovdje završava dio s naredbama koje su se izvršavale u *GRASS GIS*-u i daljnje naredbe se izvršavaju u naredbenom retku *Terminala*. Za izlazak iz *GRASS GIS*-a koristimo naredbu `unset GRASS_BATCH_JOB`. Slijedi brisanje nepotrebnih direktorija i datoteka sljedećim naredbama: `rm grassdata -rf` i `rm grass_countries.sh -f`.

Političko-teritorijalni entiteti u izlaznoj datoteci *countries_clip.shp*, koju smo dobili nakon obrade u *GRASS GIS*-u, više nisu cjeloviti jer su nakon procesa izrezivanja multipoligoni pretvoreni u poligone. Stoga ih je potrebno spojiti i očistiti kako bi opet imali cjelovite države. Prvo ćemo *Python* programom *_3_6_clean_antarctica.py* očistiti Antartiku. Ovaj jednostavni program prvo oko Antartike postavlja granični okvir (engl. *bounding box*) u zadanoj projekciji, a zatim uklanja sve objekte koji nemaju ISO kôd „AQ“ koji pripada Antartici. Program se poziva sljedećom funkcijom.

```
python3 ../_3_6_clean_antarctica.py 03_countries/clip/countries_clip.shp
```

Sljedeći korak je kreiranje multipoligona od svih dijelova pojedine države.

```
python ../_3_7_dissolve_countries_by_iso.py
03_countries/clip/countries_clip.shp 03_countries/countries_multipart.shp
```

Izlazna datoteka *countries_multipart.shp* će ponovo sadržavati 250 političko-teritorijalnih entiteta. Slijedi korak u kojem ćemo kod većih država ukloniti malene otoke koji će biti ispod minimalne veličine u zadanom mjerilu i gdje ćemo povećati malene otočne države kako ne bi bile izbrisane zbog mjerila karte. Nije potrebno unositi nikakve parametre jer se podatak provodi koristeći zadano mjerilo preko varijable *$OSM_SCALE*.

```
python ../_3_8_clean_and_enlarge_countries.py $OSM_SCALE 1
03_countries/countries_multipart.shp 03_countries/countries_clean.shp
```

Nakon toga slijedi konačna generalizacija država korištenje općeg *Python* programa *generalize.py*.

```
python ../generalize.py $OSM_SCALE 0 03_countries/countries_clean.shp
03_countries/countries_final.shp
```.

U navedenoj naredbi `../generalize.py` je naziv općeg *Python* programa i lokacija na kojoj se nalazi u odnosu na radni direktorij. `$OSM_SCALE` je varijabla koja za željeno mjerilo povlači podatak koji smo unijeli na početku pri postavljanju parametara karte. `0` je granična vrijednost, u milimetrima kvadratnim u mjerilu karte, ispod koje će svi objekti biti izbrisani. Vrijednost je postavljena na nulu jer smo brisanje obavili u prošlom koraku.

Pomoćni *Python* program *_3_9_extra_coastlines.py* služi za kreiranje obalne crte manjih otoka koji su povećani u pretprošlom koraku, a naredba za njegovo izvršavanje je:

`python ../_3_9_extra_coastlines.py 02_coastlines/coastlines_final.shp 03_countries/countries_final.shp 02_coastlines/extra_coastlines.shp`.

Sada je korištenjem *ogr2ogr* modula napravljeno spajanje dobivene datoteke *extra_coastlines.shp* s dodatnim obalnim crtama i već ranije kreiranog sloja s obalnom crtom *coastlines_final.shp*, kao je opisano u poglavlju 5.2.4.

`ogr2ogr -f "ESRI Shapefile" -update -append 02_coastlines/coastlines_final.shp 02_coastlines/extra_coastlines.shp`

Za ispravan prikaz političko-teritorijalnih entiteta na političkoj karti, osim obojenih poligona, potrebno je istaknuti kopnene granice država. Za to je korišten postupak u *GRASS GIS*-u. Naredbom `echo` izravno u *Terminalu* kreiramo posebnu izvršnu skriptu *grass_admin.sh* koja definira postupak koji će bit izvršen u *GRASS GIS*-u. Slijedi popis naredbi koje će se izvršiti u programu zajedno s opisnim objašnjenjima.

`v.in.ogr -o --overwrite dsn=./03_countries/countries_winkel.shp output=countries_winkel snap=1e-8` učitava poligone političko-teritorijalnih entiteta u Winkelovoj trostrukoj projekciji i sprema ih u vektorsku mapu *GRASS GIS*-a.

`v.type --overwrite input=countries_winkel output=countries_lines type=boundary,line` poligone iz vektorske mape pretvara u linijske objekte.

`v.in.ogr -o --overwrite dsn=./02_coastlines/coastlines_winkel.shp output=coastlines_winkel snap=1e-8` učitava poligone s obalama u Winkelovoj trostrukoj projekciji i sprema ih u vektorsku mapu *GRASS GIS*-a.

`v.overlay -t ainput=countries_lines atype=line binput=coastlines_winkel output=admin_clip operator=and` provodi presijecanje učitanih poligona i linija kopnenih granica, a izlazna datoteka je admin_clip.

`v.out.ogr input=admin_clip dsn=./03_countries olayer=admin_clip` izvozi kreirani sloj s administrativnim granicama u *admin_clip.shp* datoteku i pohranjuje ga u direktoriju *03_countries*.

Nakon postupka u *GRASS GIS*-u provodi se generalizacija i zaglađivanje dobivenog sloja po već poznatom postupku.

`python ../generalize.py $OSM_SCALE 0 03_countries/admin_clip.shp 03_countries/admin_final.shp`

Slijedi brisanje administrativnih granica na području Antartike. To ćemo obaviti već korištenim *Python* programom *_3_6_clean_antarctica.py*.

```
python ../_3_6_clean_antarctica.py 03_countries/admin_final.shp
```

Sada imamo kreiran konačni sloj s granicama država koji će se koristiti u konačnom izgledu karte. Nakon brisanja nepotrebnih direktorija i datoteka kreiranih tijekom izvršavanja ove skripte postavljaju se potrebni stupci u atributnoj tablici.

```
python ../set_fields.py osm_id,name,name_$OSM_LANG,cat,ISO3166_1,admin_leve
labels=yes 03_countries/countries_final.shp
```

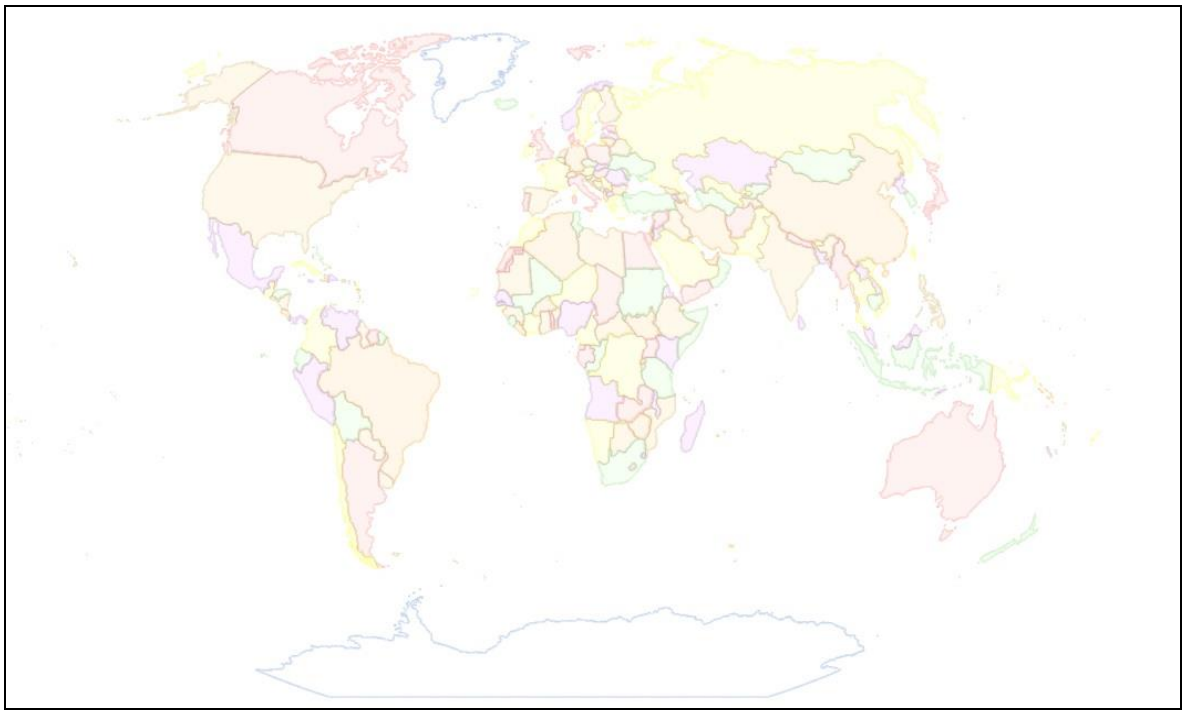
Zadnji korak u izvršnoj skripti *_3_Countries.sh* je kopiranje vrijednsti iz atributne tablice referentne karte. Za to se u originalnoj verziji softvera koristio opći *Python* program *_8_1_copyfields.py*, ali je primjećeno kako se Rusija i Zapadna Sahara ne prikazuju u završnom izgledu karte. Uzrok problema je u „osm_id“ oznaci kod tih država. Naime, na referentnoj karti Zapadna Sahara nije imala unešenu nikakvu vrijednost, a Ruska Federacija je imala unešenu vrijednost „None“. Budući da opći *Python* program *_8_1_copyfields.py* iz referentne karte preuzima podatke za stilove i smještaj nazivlja upravo preko oznake „osm_id“ podaci za te dvije zemlje nisu mogli biti pravilno kopirani. Stoga je napravljen poseban *Python* program *_8_1_copyfields_for_countries.py* koji će za sloj s političko-teritorijalnim entitetima podatke iz referentne karte kopirati po oznaci „ISO3166_1“ jer je pretpostavka da se ta vrijednost neće mijenjati u OSM bazi podataka, budući da se radi o standardu. Naredba za kopiranje vrijednsti iz atributne tablice referentne karte prikazana je u sljedećem redu.

```
python3 ../_8_1_copyfields_for_countries.py cat,_label_x,_label_y,_label_r,wrap
../PK/03_countries/countries_final.shp
```

U navedenoj naredbi `cat,_label_x,_label_y,_label_r,wrap` je popis stupaca u atributnoj tablici ulazne datoteke iz kojih ćemo kopirati vrijednosti u identične stupce atributne tablice datoteke s državama. U ovom slučaju to su vrijednost „cat“ po kojoj će se obojati pligoni država te x i y koordinata, rotacija, i prelamanje teksta naziva država. Sintaksa dohvaćanja u *Python* programu je sys.argv[1]. `../PK/03_countries/countries_final.shp` je naziv i lokacija datoteke s referentnim slojem, u odnosu na radni direktorij, iz čije atributne tablice

će se kopirati podaci, a sintaksa dohvaćanja u *_8_copyfields_for_countries.py Python* programu je sys.argv[2].

Ovime je izvršna skripta *_3_Countries.sh* stigla do kraja, a kao rezultat imamo slojeve s poligonima država i njihovim kopnenim granicama u Winkelovoj trostrukoj projekciji pohranjene u *Shapefile* datotekama *countries_final.shp* i *admin_final.shp* (Slika 5.2.). Ukupno je prikazano 250 političko-teritorijalnih entiteta kako je i bilo predviđeno, pa se izrada ove objektne skupine može smatrati uspješno izvršenom.



Slika 5.2. Konačni slojevi s državama i njihovim kopnenim granicama u Winkelovoj trostrukoj projekciji (countries_final.shp i admin_final.shp)

5.2.6 Izrada sloja s jezerima

Izvršna skripta *_4_Lakes.sh* koristi se za dobivanje objektne skupine jezera. Originalna skripta je ispitana i pokazala je određene nedostatke koji će biti opisani niže u tekstu, kao i način na koji su otklonjeni. Ažurirana verzija izvršne skripte *_4_Lakes.sh* dostupna je na digitalnom mediju priloženom uz ovaj rad.

Prvi korak je filtriranje objekata koji predstavljaju jezera naredbom

```
osmfilter ../planet.osm.o5m --keep="natural=water" -o=osm_lakes.osm
```

gdje je `osmfilter` pozivanje programa *osmfilter*, `planet.osm.o5m` ulazna datoteka koju ćemo filtrirati, `--keep="natural=water"` filter koji zadržava samo one objekte koji imaju ključ (engl. *key*) „natural“ s vrijednošću (engl. *value*) „water“, a `-o=osm_lakes.osm` kreira izlaznu datoteku osm_lakes.osm. Za izlazni format je izabran *.osm jer je nam je taj format nužan za sljedeći korak.

Prema službenim smjernicama na *web* stranici OSM-a sva jezera bi trebala za prvu oznaku imati `natural=water`, a potom oznake s ključevima `water=` i `name=`. Vrijednost ključa name= predstavlja ime objekta, a neke od vrijednosti koje ključ water= može imati su: lake, basin, reservoir, pond, lagoon (URL 69). S obzirom da i objekti koji za vrijednost ključa water= nemaju vrijednost lake, a zbog površine koju zauzimaju trebaju biti prikazani na karti, filtracija će se vršiti isključivo po oznaci natural=water. Međutim, iz smjernica za oznaku natural=water vidljivo je da se ta oznaka korisit za sve površinske vode iz čitavog OSM-a, kako za stajaćice, tako i za tekućice (URL 70). Zbog toga bi nam nakon filtracije u izlaznoj datoteci trebali biti svi objekti koji predstavljaju površinske vode, što uključuje i rijeke. Iako će se dobivanje sloja s rijekama izvršiti s izvršnom skriptom *_5_Rivers.sh* on će sadržavati samo linijski prikaz rijeka. Ipak, zbog širine svojeg korita pojedini djelovi rijeka mogu se prikazati kao poligoni, a njihov broj ovisi o mjerilu karte koja se izrađuje. Upravo takvi slučajevi će biti zadržani u sloju s jezerima u izlaznoj datoteci *lakes_final.shp* nakon što se izvršna skripta *_4_Lakes.sh* izvrši.

Sljedeći korak je konverzija iz OSM-ovog formata *.osm u Shapefile format (*.shp) pomoću GDAL/OGR modula *ogr2ogr* naredbom:

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"SELECT osm_id,name,name_$OSM_LANG,intermittent FROM multipolygons WHERE
OGR_GEOM_AREA>0.01" -overwrite -skipfailures -lco ENCODING=UTF-8 04_lakes
osm_lakes.osm
```

U toj naredbi `-sql "SELECT osm_id,name,name_$OSM_LANG,intermittent FROM multipolygons WHERE OGR_GEOM_AREA>0.01"` je sql upit koji iz OSM sloja multipolygons povlači sve objekte veće od 0.01 kvadratnog stupnja i za njih preuzima vrijednosti oznaka *osm_id, name, name_$OSM_LANG* i *intermittent*. `04_lakes` definira naziv i lokaciju odredišnog direktorija u odnosu na radni direktorij, a `osm_lakes.osm` je naziv i lokacija

ulazne datoteke u odnosu na radni direktorij. Ukoliko naziv izlazne datoteke nije zadan izlazna datoteka će dobiti naziv po OSM sloju iz kojeg se podaci filtriraju. U ovom slučaju datoteka će dobit naziv *multipolygons.shp*.

Gore navedena *ogr2ogr* konverzija razlikuje se od one iz originalne verzije izvršne skripte za jezera jer u *sql* upitu sadrži dio s filtriranjem objekata prema površini. Originalna verzija skripte glasi:

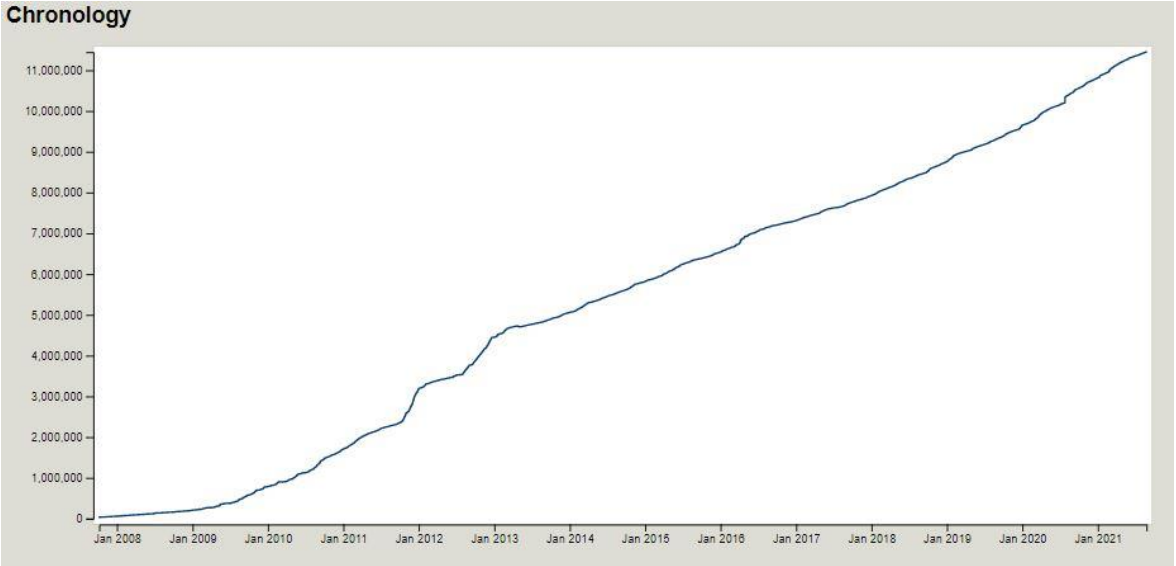
```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"SELECT osm_id,name,name_$OSM_LANG,intermittent FROM multipolygons" -overwrite
-skipfailures -lco ENCODING=UTF-8 04_lakes osm_lakes.osm
```

Nakon pokretanja iste u *Terminalu* javljala su se sljedeća upozorenja:

```
Warning 6: Normalized/laundered field name: 'intermittent' to 'intermitte'
Warning 1: 2GB file size limit reached for 04_lakes/multipolygons.shp. Going
on, but might cause compatibility issues with third party software
Warning 1: 2GB file size limit reached for 04_lakes/multipolygons.dbf. Going
on, but might cause compatibility issues with third party software
ERROR 1: Failed to write shape object. File size cannot reach 4294967084 +
1096.
ERROR 1: Failed to write shape object. File size cannot reach 4294967084 + 792.
ERROR 1: Failed to write shape object. File size cannot reach 4294967084 + 936.
(...)
More than 1000 errors or warnings have been reported. No more will be reported
from now.
```

Upozorenja se javljaju zbog veličine *Shapefile* datoteke i prateće *.dbf datoteke koje nastaju kao izlazne datoteke pri *ogr2ogr* konverziji. Nakon što Shapefile datoteka dosegne svoj limit (~4,3 GB) proces se i dalje nastavlja dok se ne „provuče“ cijela ulazna *osm_lakes.osm* datoteka kroz postupak konverzije, ali se novi podaci ne zapisuju u izlaznu *multipolygons.shp* datoteku. Zbog toga jedan dio podataka ostaje izgubljen i takva konverzija ne može se smatrati potpunom ni ispravnom jer mi nemamo kontrolu nad ovim procesom i ne znamo koji su se podaci (objekti) izgubili. Ukupno su u nepotpunoj *multipolygons.shp* datoteci bila zapisana 3 269 552 objekta. Kao što je ranije rečeno datoteka *osm_lakes.osm* je dobivena iz *planet.osm.o5m* datoteke korištenjem programa *osmfilter* i filtera *--keep="natural=water"* što znači da se u datoteci *osm_lakes.osm* nalaze sve, pa čak

i one najmanje vodene površine, koje su ikad unešene u OSM. Analizom podataka s *web* stranice OSM-a utvrđeno je da trenutačno 11 446 653 objekta imaju oznaku (engl. *tag*) „natural=water“(URL 71). Također je utvrđeno da je 2016. godine, u vrijeme kad je rađena originalna verzija skripte, takvih objekata bilo manje od 7 milijuna (Slika 5.3.) i to je razlog zašto je konverzija tada uspješno u potpunosti izvršena.



Slika 5.3. Kronologija rasta broja objekata s oznakom „natural=water“ u OSM-u

Iako će kasnije kroz postupak velika većina objekata dobivenih nakon *ogr2ogr* konverzije otpasti zbog premalene površine i neće biti prikazani u završnom sloju jezera bilo je potrebno pronaći način da se količina podataka koji se zapisuju u *multipolygons.shp* smanji. U originalnoj verziji skripte idući korak je brisanje malih poligona koji su manji od ~1 km^2 naredbom `python ../simplify_polygon.py 0 0.01 04_lakes/multipolygons.shp`, ali mi u novoj verziji skripte tako nešto moramo odraditi prije ili tijekom kreiranja *multipolygons.shp* datoteke. Stoga smo u dio s sql upitom dodali dio `WHERE OGR_GEOM_AREA>0.01` koji će osigurati da se prilikom *ogr2ogr* konverzije u *multipolygons.shp* zapisuju samo oni objekti koji su veći od 0,01 kvadratnog stupnja. U ovom trenutku naši podaci su u sustavu lat/lon i zato zadajemo vrijednosti u kvadratnim stupnjevima. Sada se naša *ogr2ogr* konverzija uredno izvrši do kraja, a svi objekti koji prođu filtraciju po zadanoj površini su uspješno zapisani u *multipolygons.shp*. Tih objekata (poligona) je kupno 2916, a sama datoteka je veličine 545 MB.

Kao što je gore rečeno, u originalnoj verziji izvršne skripte tek nakon *ogr2ogr* konverzije slijedi brisanje manjih poligona (manjih od ~1 km^2) što znači da će u novoj verziji dio

```
echo "Deleting small polygons less than ~1 km2 on Earth..."
python ../simplify_polygon.py 0 0.01 04_lakes/multipolygons.shp
```

biti izostavljen. Tako dolazimo do dijela izvršne skripte koji će podatke iz vektorske *shapefile* datoteke reprojicirati u Winkelovu trostruku projekciju:

```
python3 ../reproject.py 04_lakes/multipolygons.shp 04_lakes/lakes_winkel.shp
```

U toj naredbi `../reproject.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, `04_lakes/multipolygons.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, s ulaznim podacima koje treba reprojicirati, a sintaksa dohvaćanja u *reproject.py Python* programu je sys.argv[1]. `04_lakes/lakes_winkel.shp` je naziv i lokacija izlazne datoteke, u odnosu na radni direktorij, koja će biti u Winkelovoj trostrukoj projekciji, a sintaksa dohvaćanja u *reproject.py Python* programu je sys.argv[2].

Slijedi nam konačna generalizacija i brisanje onih jezera koja su površinom manja od 0,4 mm^2 u mjerilu karte. U izvršnoj skripti naredba za taj postupak glasi:

```
python3 ../generalize.py $OSM_SCALE 0.4 04_lakes/lakes_winkel.shp
04_lakes/lakes_final.shp
```

U navedenoj naredbi `../generalize.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, `$OSM_SCALE` je varijabla koja za željeno mjerilo povlači podatak koji smo unijeli na početku pri postavljanju parametara karte, a sintaksa dohvaćanja u *generalize.py Python* programu je sys.argv[1]. `0.4` je granična vrijednost, u milimetrima kvadratnim u mjerilu karte, ispod koje će svi objekti biti izbrisani, a sintaksa dohvaćanja je sys.argv[2]. `04_lakes/lakes_winkel.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, s ulaznim podacima koje treba provesti kroz generalizaciju i brisanje, a sintaksa dohvaćanja je sys.argv[3]. `04_lakes/lakes_final.shp` je naziv i lokacija izlazne datoteke, u odnosu na radni direktorij, a sintaksa dohvaćanja u *generalize.py Python* programu je sys.argv[4].

Sada smo dobili *lakes_final.shp*, završnu *Shapefile* datoteku za jezera (Slika 5.4) koja će se koristiti u konačnom izgledu karte gdje broj objekata i njihova geometrija više neće biti dirani. Ipak potrebno je napravit još nekoliko stvari. Krenut ćemo s brisanjem nepotrebnih

datoteka koje su dosad nastale prilikom izvršavanja ove izvršne skripte. Potom se sljedećom naredbom postavljaju potrebni stupci u atributnoj tablici.

```
python3 ../set_fields.py osm_id,name,name_$OSM_LANG,intermitte,water labels=yes 04_lakes/lakes_final.shp
```

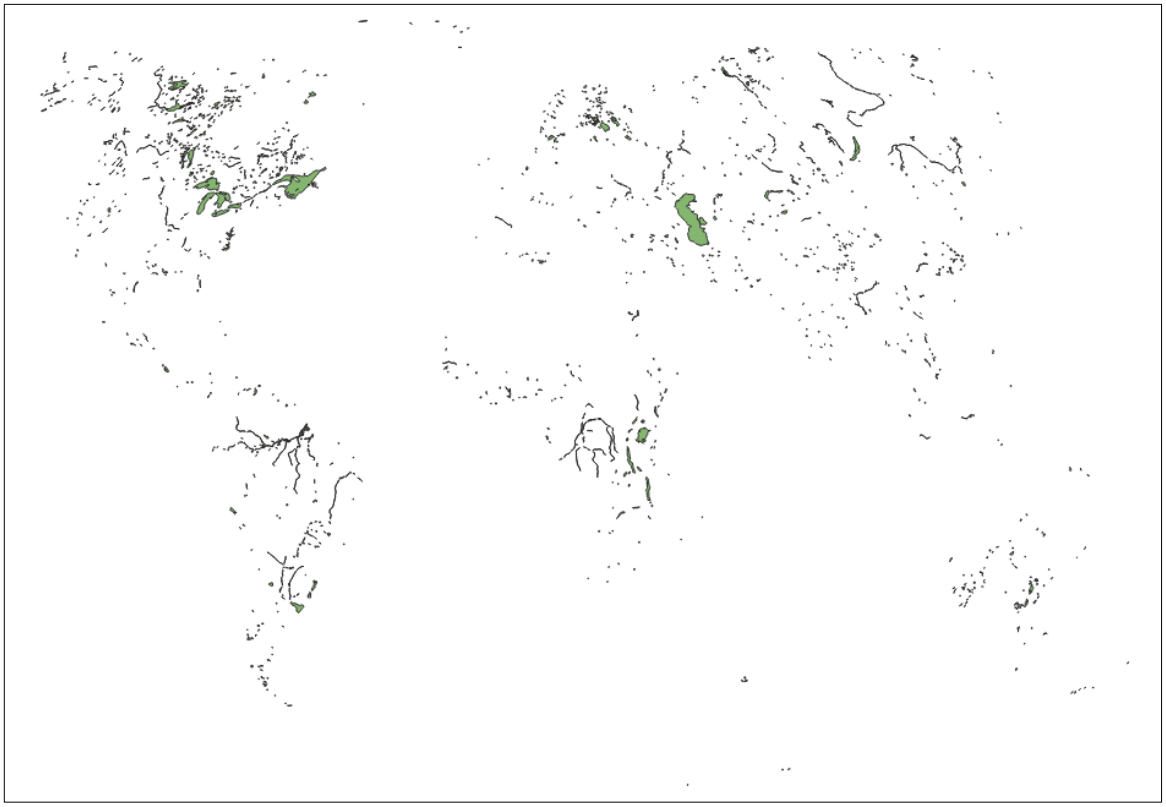
U toj naredbi `../set_fields.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti. `osm_id,name,name_$OSM_LANG,intermitte` je popis stupaca koje ćemo zadržati u atributnoj tablicu. U ovom slučaju to su osm_id, lokalno ime, ime na jeziku kojeg smo definirali na početku izrade ove karte i podatak je li jezero privremeno bez vode tj. presušilo. Zadnje navedeni podatak nam je bitan jer će linije takvih jezera biti isprekidane. Sintaksa dohvaćanja ovog popisa u *set_fields.py Python* programu je sys.argv[1]. Opcija `labels=yes` govori da želimo dodatne stupce za ručno pomicanje i rotaciju naziva. Sintaksa dohvaćanja je sys.argv[2]. `04_lakes/lakes_final.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, za koju ćemo napraviti promjene u atributnoj tablici. Sintaksa dohvaćanja je sys.argv[3].

Zadnji korak u izvršnoj skripti *_4_Lakes.sh* je kopiranje vrijednsti iz atributne tablice referentne karte.

```
python3 ../_8_copyfields.py _label_x,_label_y,_label_r,wrap ../PK/04_lakes/lakes_final.shp ../$OSM_DIR/04_lakes/lakes_final.shp
```

U navedenoj naredbi `../_8_copyfields.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, `_label_x,_label_y,_label_r,wrap` je popis stupaca u atributnoj tablici ulazne datoteke iz kojih ćemo kopirati vrijednosti u identične stupce atributne tablice datoteke s jezerima koju izrađujemo. U ovom slučaju to su x i y koordinata, rotacija, i prelamanje teksta naziva. Sintaksa dohvaćanja u *_8_copyfields.py Python* programu je sys.argv[1]. `../PK/04_lakes/lakes_final.shp` je naziv i lokacija datoteke s referentnim slojem, u odnosu na radni direktorij, iz čije atributne tablice će se kopirati podaci. Sintaksa dohvaćanja je sys.argv[2]. `../$OSM_DIR/04_lakes/lakes_final.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, u čiju atributnu tablicu će se zapisati kopirani podaci. Sintaksa dohvaćanja u *_8_copyfields.py Python* programu je sys.argv[3].

Ovime je izvršna skripta *_4_Lakes.sh* stigla do kraja, a kao rezultat imamo sloj s jezerima u Winkelovoj trostrukoj projekciji pohranjen u *Shapefile* datoteku *lakes_final.shp* (Slika 5.4). Ukupno su dobivena 1673 objekta (engl. *features*), a datoteka je veličine 2,08 MB.



Slika 5.4. Konačni sloj s jezerima u Winkelovoj trostrukoj projekciji (lakes_final.shp)

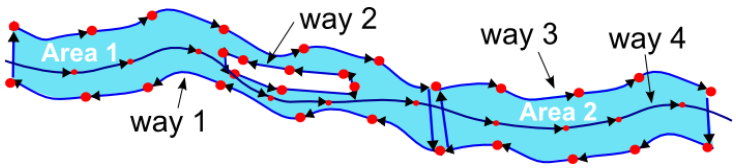
U originalnoj verziji izvršne skripte za jezera jezero Huron je zbog pogreške u podacima OSM-a moralo biti izdvojeno posebno (Jogun, 2016). Radi se o izuzetno kompliciranoj obalnoj crti i postojala je neka greška u toplogiji OSM-a koja nije bila problematična za vizualizaciju unutar samog OSM-a, ali je radila problem zbog kojeg taj objekt nije bio uspješno izdvojen nakon filtracije programom *osmfilter*. U međuvremenu je taj problem ispravljen unutar OSM-a i taj dio originalne izvršne skripte za jezera je izbačen u ažuriranoj verziji.

5.2.7 Izrada sloja s rijekama

Izvršna skripta *_5_Rivers.sh* koristi se za dobivanje objektne skupine rijeka. Originalna skripta je ispitana i pokazala se zadovoljavajućom nakon nekih sitnih ispravaka. Uglavnom se to odnosi na nove oznake za pozivanje opcija unutar *GRASS GIS*-u. Tako je

npr. oznaka `column` u naredbama `v.db.addcol` i `v.to.db` po novom `columns`, oznaka `thres` u naredbi `v.clean` je sada `threshold`, a u svaki redak naredbe `r.mapcalc` dodane su oznake `--overwrite` i `expression=`. Ažurirana verzija izvršne skripte *_5_Rivers.sh* dostupna je na digitalnom mediju priloženom uz ovaj rad.

Prema službenim smjernicama na web stranici OSM-a rijeke se mogu prikazati na dva načina – kao linearni put (engl. *linear way*) ili kao vodena površina (engl. *water area*). Za svaku rijeku put s oznakom `waterway=river` (way 4, Slika 5.5.) mora se povući linearni put u smjeru riječnog toka tj. od izvora do ušća. U idealnom slučaju ova linija bi trebala biti položena po najdubljim točkama korita. Naravno da je to teško odrediti pa je stoga prihvatljivo i približno određivanje. Veliki dijelovi rijeke koji su dovoljno široki mogu se kartirati kao vodene površine. Za male rijeke nije potrebno osim linearnog puta ucrtavati vodenu površinu, ali je svakako dobrodošlo. Vodena površina rijeke treba biti označeno površinom koja se prostire duž obala rijeke. Budući da su široke rijeke obično i dugačke, rijeka će se u praksi crtati kao niz susjednih područja (way 1 i way 3, Slika 5.5.) ili kao multipoligon. Ove vodene površine trebaju biti označena ili oznakom `waterway=riverbank` ili oznakama `natural=water` i `water=river`. Preporuka je da se korištenje prve oznake zamjeni s kombinacijom ove druge dvije oznake, ali prva oznaka i dalje ostaje u upotrebi budući da ga neki kartografi još uvijek preferiraju (URL 72).



Slika 5.5. Ilustracija načina kartiranja rijeka u OSM-u (URL 73)

Izvršna skripta *_5_Rivers.sh* počinje filtriranjem onih objekata koji predstavljaju rijeke u OSM Planet datoteci. To ćemo, iz gore navedenih razloga, učiniti sljedećom naredbom.

```
osmfilter ../planet.osm.o5m --keep= --keep-relations="waterway=river" -
o=osm_rivers.osm
```

U toj naredbi `osmfilter` poziva program *osmfilter*, `planet.osm.o5m` je ulazna datoteka koju ćemo filtrirati, `--keep= --keep-relations="waterway=river"` je filter koji zadržava samo one relacije koje imaju ključ „waterway“ i vrijednost ključa „river“, dok `-o=osm_rivers.osm`

kreira izlaznu datoteku *osm_rivers.osm* u koju pohranjuje objekte koji su prošli filtraciju. Za izlazni format je izabran *.osm jer je nam je taj format potreban za daljnji postupak.

Sljedeći korak je konverzija rijeka iz OSM-ovog formata *.osm u Shapefile format (*.shp) pomoću modula *ogr2ogr*.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"select * from other_relations where type='waterway'" -overwrite -skipfailures
-nlt MULTILINESTRING -lco ENCODING=UTF-8 05_rivers osm_rivers.osm
```

U ovoj naredbi `-sql "select * from other_relations where type='waterway'"` je *sql* upit koji iz sloja „other_relations“ povlači sve objekte s oznakom type='waterway' i za njih preuzima sve dostupne oznake, `-nlt MULTILINESTRING` opcija definira MULTILINESTRING kao vrstu geometrije za kreirani sloj. `05_rivers` definira naziv i lokaciju izlaznog direktorija u odnosu na radni direktorij, a `osm_rivers.osm` je naziv i lokacija ulazne datoteke u odnosu na radni direktorij. Kako naziv izlazne datoteke nije definiran, izlazna datoteka će dobiti naziv po OSM sloju iz kojeg se podaci filtriraju, a u ovom slučaju to znači da će izlazna datoteka dobit naziv *other_relations.shp*.

Sada dolazimo do dijela u kojem dobivene podatke reprojiciramo u Winkelovu trostruku projekciju.

```
python3 ../reproject.py 05_rivers/other_relations.shp
05_rivers/rivers_winkel.shp
```

U navedenoj naredbi `05_rivers/other_relations.shp` je naziv i lokacija datoteke s ulaznim podacima koje treba reprojicirati, a `05_rivers/rivers_winkel.shp` je naziv i lokacija izlazne datoteke sa svim jezerima u Winkelovoj trostrukoj projekciji.

Kako bi preglednost na karti bila zadovoljavajuća, i kao jedan vid generalizacije rijeka, izbrisat ćemo sve one koje bi na karti bile kraće od 3 centimetra.

```
python3 ../simplify_line.py 0 $[$OSM_SCALE / 100 * 3]
05_rivers/rivers_winkel.shp
```

U navedenoj naredbi `../simplify_line.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, `0` je granična vrijednost za „prag udaljenost“ koji nam u ovom slučaju ne treba, pa je postavljen na 0, `$[$OSM_SCALE / 100 * 3]` je izraz po kojem se u odnosu na zadano

mjerilo računa granična vrijednost praga duljine. `05_rivers/rivers_winkel.shp` je naziv i lokacija datoteke s ulaznim podacima koje treba provesti kroz generalizaciju i brisanje.

Zbog načina na koji se u OSM-u kartiraju rijeke, a koji je opisan ranije, jasno je da bi moglo biti problema s topologijom. Rijeke su napravljene iz velikog broja manjih objekata i vrlo su topološki nečiste. Izgledno je da u mnogim slučajevima susjedne površine neće imati savršen preklop u istim točkama (npr. area 1 i area 2 na Slika 5.5.) ili da će jeadn dio rijeke biti kartiran multipoligonima, a drugi samo linearnim putem. Može se dogoditi i da jedan dio rijeke, npr. 10-ak metara nije iskartiran, pa nemamo cijelu rijeku topološki povezanu. U tim slučajevima ne bi dobili jedan objekt za jednu rijeku.

Stoga smo se odlučili u *GRASS GIS-u* naš *Shapefile* s rijekama pretvoriti u raster, pa ga zatim vratiti u vektor. Kad podatke prebacimo u rasterski oblik, sve one manje nepravilnosti i „rupe“ u topologiji ne bi se trbale vidjeti. Pretvaranjem natrag u vektorski format trebali bi dobiti jedan, topološki čist, objekt. Prvo iz *Terminala* pokrećemo opću izvršnu skriptu *grass_xy_location.sh* naredbom `../grass_xy_location.sh` s kojom postavljamo privremenu lokaciju u *GRASS GIS*-u i kreiramo potrebne radne direktorije. Naredbom `echo` izravno u *Terminalu* kreiramo posebnu izvršnu skriptu *grass_rivers.sh*, a zatim niz naredbi `chmod u+x grass_rivers.sh`, `export GRASS_BATCH_JOB=./grass_rivers.sh` i `grass -text "$PWD/grassdata/xy/data"` daje dopuštenje korisniku za izvršavanje kreirane *grass_rivers.sh* skripte, otvara skriptu kao posao u *GRASS GIS*-u i definira radni direktorij. Slijedi popis naredbi, i njihovih objašnjenja, koje će se izvršiti u *GRASS GIS*-u u sklopu posebne izvršne skripte *grass_rivers.sh*.

```
v.in.ogr -o --overwrite input=./05_rivers/rivers_winkel.shp
output=rivers_winkel
```

`v.in.ogr` je naredba koja uvozi vektorske podatke u *GRASS GIS*-ovu vektorsku mapu koristeći OGR biblioteku. `-o` je opcija za zaobilaženje provjere projekcije, a u procesu će se koristiti projekcija trenutne lokacije. `input=./05_rivers/rivers_winkel.shp` definira naziv OGR izvora podataka za uvoz, u ovom slučaju direktorij s *rivers_winkel.shp* datotekom koju ćemo koristiti u *GRASS GIS*-u. `output=rivers_winkel` je naziv izlazne mape koja se pohranjuje u *GRASS GIS* radni direktorij.

```
g.region vector=rivers_winkel res=$[$OSM_SCALE / 10000]
```

Naredba `g.region` upravlja definiranjem granica za geografsko područje, a `vector=rivers_winkel` postavlja regiju tako da odgovara vektorskoj mapi za *rivers_winkel*. `res=$[$OSM_SCALE / 10000]` definira rezoluciju 2D mreže (veličinu piksela), a ovdje je dodano da to računa u odnosu na zadano mjerilo ($OSM_SCALE). U našem slučaju jedan piksel će imati duljinu i širinu 2000 metara.

```
v.to.rast --overwrite input=rivers_winkel layer=1 type=point,line,area
output=smooth use=cat
```

Naredba `v.to.rast` pretvara vektorske slojeve mape u rasterske slojeve mape. Opcionalno se atributi mogu pretvoriti u oznake rasterskih kategorija. Naziv ulazne vektorske mape koja se ubacuje iz *GRASS GIS* radnog direktorija je `input=rivers_winkel`, a `layer=1` je broj sloja. `type=point,line,area` određuje vrstu značajke koja ulazi u proces rasterizacije. `output=smooth` je naziv izlazne rasterske mape, a opcija `use=cat` definira izvor rasterskih vrijednosti koje ćemo koristiti.

Nakon konverzije u raster potrebno je ispraviti nesavršenosti samog rastera. To se odnosi na „rubne rupe“ i piksele koji „strše“ u rasteru linijskog objekta kakve su rijeke. Stoga slijedi dio u kojem se koristi naredba *r.mapcalc*. Ta naredba izvodi aritmetičke operacije na slojevima rasterske mape. Mogu se stvoriti novi slojevi rasterske mape koji su aritmetički izrazi koji uključuju postojeće slojeve rasterske mape, cjelobrojne ili promjenjive konstante i funkcije. Mape su datoteke baze podataka pohranjene u rasterskom formatu, tj. dvodimenzionalne matrice cijelih brojeva. U *r.mapcalc*, mape mogu biti procesuirane modifikatorom susjedstva koji specificira relativno odstupanje od trenutne ćelije koja se procjenjuje. Format je map [r, c], gdje je r pomak retka, a c pomak stupca. Na primjer, map [1,2] odnosi se na ćeliju jedan red ispod i dva stupca desno od trenutne ćelije. Prikaz vrijednosti lokalnih koordinata svakog piksela je prikazan na slici ispod.

[-1, -1]	[-1, 0]	[-1, 1]
[0, -1]	trenutni piksel	[0, 1]
[1, -1]	[1, 0]	[1, 1]

Slika 5.6. Vrijednosti lokalnih koordinata susjednih piksela

Naredba *r.mapcalc* će nam pomoći da zagladimo naš raster na razini jednog piksela. To znači da će se proces odraditi za svaki piksel u rasterskoj mapi. Prvo će se izvršiti „popunjavanje rupa“ svim varijacijama sljedeće dvije naredbe oko pojedinog piksela:

```
r.mapcalc --overwrite expression=\"smooth1 = if(smooth[-1,0] &&& smooth[0,1]
&&& smooth[1,0],smooth[0,1],null())\"
```

```
r.mapcalc --overwrite expression=\"smooth =
if(isnull(smooth1),smooth,smooth1)\"
```

Za vizualnu predodžbu predlažem slučaj sa Slika 5.7. gdje je „P“ piksel na kojem se izvodi proces. Bijela polja predstavljaju piksele koji nemaju vrijednost (vrijednost je NULL), a plava polja su pikseli s vrijednošću i predstavljaju rubnu crtu neke rijeke.

[-1, -1]	[-1, 0]	[-1, 1]
[0, -1]	P	[0, 1]
[1, -1]	[1, 0]	[1, 1]

Slika 5.7. Primjer „rupe“ na rubnoj crti u rasteru s rijekama

Kao što vidimo, piksel „P“ nema vrijednost, a lijevo, desno i ispod njega su pikseli koji imaju vrijednost. Da bi raster bio zaglađen, u ovom slučaju bi trebalo pikselu „P“ dati neku vrijednost da „popunimo rupu“. U prvoj *r.mapcalc* naredbi ispitat će se imaju li pikseli s lokalnim koordinatama smooth[-1,0], smooth[0,1], smooth[1,0] vrijednost, a ukoliko sva tri imaju vrijednost varijabla `smooth1` će dobiti vrijednost koju ima piksel smooth[0,1]. U drugoj *r.mapcalc* naredbi će za ovaj slučaj varijabla `smooth` (koja definira naš trenutni piksel „P“) dobiti vrijednost varijable `smooth1` tj. vrijednost koju ima piksel smooth[0,1] – piksel ispod trenutnog piksela „P“.

Navedene dvije naredbe *r.mapcalc* su napisane tako da idu u paru jedna iza druge i da pikselu bez vrijednosti daju vrijednost samo ukoliko tri piksela koja ga ukružuju imaju vrijednost.

Slijedi drugi dio obrade rastera u svrhu „zaglađivanja“ rubova linijskih objekata – brisanje piksela koji „strše“. To će za svaki pojedini piksel odraditi sljedeća `r.mapcalc` naredba uključujući i preostale moguće varijacije:

```
r.mapcalc --overwrite expression=\"smooth = if(isnull(smooth[0,-1]) &&
isnull(smooth[-1,-1]) && isnull(smooth[-1,0]) && isnull(smooth[-1,1]) &&
isnull(smooth[0,1]),null(),smooth)\"
```

U svrhu vizualne predodžbe prikazan je jedan takav slučaj na Slika 5.8 gdje je „P“ piksel koji „strši“ i na kojem se izvodi proces. Kao i u prošlom primjeru, bijela polja predstavljaju piksele bez vrijednosti (vrijednost je *NULL*), dok ona plava simboliziraju piksele koji imaju vrijednost i predstavljaju rubnu crtu neke rijeke.

[-1, -1]	[-1, 0]	[-1, 1]
[0, -1]	P	[0, 1]
[1, -1]	[1, 0]	[1, 1]

Slika 5.8. Primjer piksela koji „strši“ na rubnoj crti u rasteru s rijekama

Piksel „P“, koji je u navedenoj naredbi *r.mapcalc* definiran varijablom `smooth`, u ovom slučaju ima neku vrijednost, a lijevo, desno i iznad njega su pikseli koji nemaju vrijednost i ne predstavljaju rijeku. Da bi zagladili ovakav dio rastera trebalo bi piksel „P“ izbrisati tj. pridružiti mu vrijednost *NULL*. Naredba *r.mapcalc* ispitat će je li vrijednost svih pet piksela (smooth[0,-1], smooth[-1,-1], smooth[-1,0], smooth[-1,1], smooth[0,1]) koji okružuju piksel „P“ *NULL* i samo u tom slučaju će pikselu „P“ dati novu vrijednost *NULL* te na taj način „zagladiti“ ovaj dio rasterske mape. Naredba *r.mapcalc* je za ovaj dio obrade napisana tako da za sve ostale slučajeve, kad okolnih pet piksela nema vrijednost *NULL*, ne mijenja vrijednost ternutnog piksela.

Za detaljnija objašnjenja kako navedene *r.mapcalc* naredbe funkioniraju i kako se iz if-upita dobiju vrijednosti za varijable „smooth1“ i „smooth“ predlažem poglavlja „NULL and conditions“ i „NULL support“ na *GRASS GIS* službenoj *web* stranici (URL 74).

Bilo bi idealno kad bi rijeke, kao linijski objekti na karti, u konačnom vektorskom sloju bile prikazane linijama. U tome će nam pomoći *GRASS GIS* naredba `r.thin` koja stanjuje ćelije koje nisu nulte (vrijednost im nije *NULL*), a predstavljaju linearne značajke značajke u sloju rasterske mape. Tom naredbom ćemo stanjiti sve linije koje predstavljaju rijeke na debljinu jednog piksela (engl. *sceleton*) prije nego vratimo naš sloj s rijekama natrag u vektorski oblik.

Navedena naredba se izvršava sve dok ne dođe do dijela kad se više ne mogu ukloniti daljnji pikseli bez da dođe do „pucanja“ ili skraćivanja linija. U postupaku se uvijek zadržavaju linije debele jedan piksel što nam je vrlo bitno jer se tako čuva topologija. Rijeke će ostati povezane sa svim svojim pritocima i rukavcima. Broj ponavljanja ovisi o izvornoj debljini objekta. Svaka iteracija „oguli“ samo jedan vanjski „sloj“ tj. vanjske piksele objekta. Slijedi opis naše `r.thin` naredbe.

```
r.thin --overwrite input=smooth output=thin iterations=200
```

Naredba `r.thin` poziva *GRASS GIS* na korištenje funkcije za stanjivanje. `input=smooth` je naziv ulazne rasterske mape koja se ubacuje iz *GRASS GIS* radnog direktorija, a koju smo kreirali i procesuirali u prošlim koracima. `output=thin` je naziv izlazne mape koja se pohranjuje u *GRASS GIS* radni direktorij. `iterations=200` definira maksimalni broj ponavljanja. Zadana vrijednost je 200. Mi smo ostavili tu vrijednost uz pretpostavku da nijedna rijeka neće imati toliku širinu s obzirom da nam jedan piksel predstavlja područje veličine dva kilometra u duljinu i širinu.

Ovaj proces ipak može stvoriti male "viseće linije" tijekom procesa stanjivanja, a to su u našem slučaju pritoci rijeka. One se mogu ukloniti alatom `rmdangle` u naredbi `v.clean` što ćemo mi i napraviti jer tako izbacujemo kraće pritoke rijeka koji se neće vidjeti na karti. Za početak podatke treba prebacit u vektorski oblik.

```
r.to.vect -v --overwrite input=thin output=thin type=line
```

`r.to.vect` je funkcija koja skenira ulazni sloj rasterske mape, iz njega izvlači točke, crte ili rubne značajke područja i pretvara podatke u *GRASS GIS* vektorski format. Kad je uključena opcija `-v` koristit će se rasterske vrijednosti kao kategorije. `input=thin` definira naziv ulazne rasterske mape koja se ubacuje iz *GRASS GIS* radnog direktorija, a `output=thin` definira naziv izlazne vektorske mape koja se pohranjuje u *GRASS GIS* radni direktorij. Vrsta izlazne

značajke definira opcija `type=line`. Moguće vrijednosti su: *point, line, area*. Rijeke su linijski objekti, pa mi želimo da samo linije budu jedine značajke u izlaznoj datoteci.

Slijedi brisanje manjih pritoka rijeka naredbom koja je navedena ispod.

```
v.clean input=thin type=line output=clean tool=rmdangle,rmdangle
threshold=$[$OSM_SCALE / 1500],$[$OSM_SCALE / 500] --overwrite
```

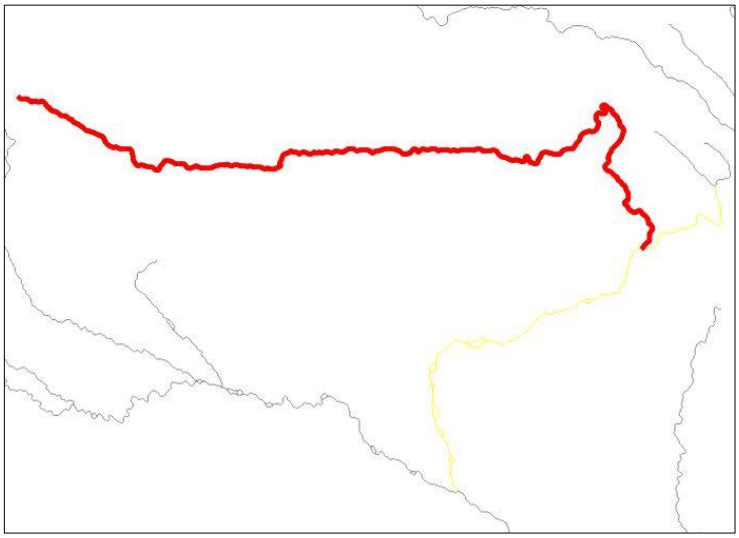
U navedenoj naredbi `v.clean` omogućuje korisniku automatsko popravljanje topologije vektorske mape. Može se navesti nekoliko alata koji se izvode uzastopno, a u tom slučaju mora biti navedeno i jednako toliko parametara za graničnu vrijednost praga. `input=thin` je naziv ulazne vektorske mape koju smo kreirali u prošlom koraku, a `output=clean` naziv izlazne vektorske mape koja se pohranjuje u *GRASS GIS* radni direktorij. `tool=rmdangle,rmdangle` je popis alata za čišćenjem u onom redoslijedu kojim će se koristiti. Mi želimo dva puta, s različitim graničnim vrijednostima praga, provesti čišćenje alatom `rmdangle` koji uklanja manje pritoke rijeka. `threshold=$[$OSM_SCALE / 1500],$[$OSM_SCALE / 500]` je opcija za zadavanje granične vrijednosti pragova za pozvane alate u jedinicama karte. Zadaje se jedna vrijednost za svaki pozvani alat. Prva granična vrijednost će osigurati da se u ovom slučaju (u odnosu na definirano mjerilo) izbrišu pritoci rijeka kraći od 13 kilometara, a druga granična vrijednost briše pritoke kraće od 40 kilometara.

Sada su naše rijeke u malim djelovima i potrebno je umjesto toga kreirati velike polilinije gdje god je to moguće.

```
v.build.polylines --overwrite input=clean output=builded_rivers cats=first
```

`v.build.polylines` je naredba koja izrađuje polilinije od linija ili granica u vektorskoj mapi. Liniju definira jedan početni čvor, jedan krajnji čvor i bilo koji broj vrhova između početnog i završnog čvora. Jedna polilinija se može kreirati samo do grananja rijeke, kao što je prikazano na primjeru rijeke Brahmaputra koja se nakon obrade ovom naredbom sastoji od 27 polilinija, a najduža je označena crvenom bojom i jasno se vidi da završava gdje se rukavci spajaju (Slika 5.9.). Općenito govoreći, polilinije pružaju najprikladniji prikaz zakrivljenih linija kada je važno da čvorovi služe za definiranje topologije. `input=clean` je naziv ulazne vektorske mape iz prošlog koraka, a `output=builded_rivers` je naziv izlazne

vektorske mape koja se pohranjuje u *GRASS GIS* radni direktorij. Opcija `cats=first` dodijeljuje broj kategorije prve linije cijeloj poliliniji.



Slika 5.9. Rijeka Brahmaputra nakon obrade naredbom „v.build.polylines“

Potrebno je prepisati podatke u atributne tablicu nove vektorske datoteke. To ćemo napraviti naredbom ispod.

```
v.db.connect -o map=builded_rivers table=rivers_winkel
```

Naredba `v.db.connect` ispisuje ili postavlja vezu do baze podataka za vektorsku mapu. Opcija `-o` prebrišite parametar povezivanja za određeni sloj. `map=builded_rivers` je naziv vektorske mape ili izvor podataka za izravan OGR pristup. To je u prošlom koraku stvorena datoteka u čiju atributnu tablicu ćemo prepisati podatke. `table=rivers_winkel` je naziv atributne tablice iz koje će se podaci prepisati. U ovom slučaju to je atributna tablica *rivers_winkel.shp* datoteke koja je bila ulazna datoteka za cijeli postupak u *GRASS GIS-u.*

Preostaje nam izvoz iz *GRASS GIS*-ove vektorske mape u vektorski format.

```
v.out.ogr --overwrite -c input=builded_rivers type=line
output=./05_rivers/builded_rivers.shp format=ESRI_Shapefile lco=ENCODING=UTF-8
```

Naredba `v.out.ogr` pretvara sloj *GRASS GIS* vektorske mape u bilo koji od podržanih OGR vektorskih formata. Kad je uključena opcija `-c`, izvozit će se i značajke bez kategorije. `input=builded_rivers` definira naziv ulazne vektorske mape za izvoz, a `type=line` određuje vrstu značajki koje želimo izvesti. Naziv izlazne datoteke i lokacija na koju će se spremiti definiramo s `output=./05_rivers/builded_rivers.shp`.

Ovdje završava dio s naredbama koje su se izvršavale u *GRASS GIS*-u i daljnje naredbe se izvršavaju u naredbenom retku *Terminala*. Za izlazak iz *GRASS GIS*-a koristimo naredbu `unset GRASS_BATCH_JOB`, pa slijedi brisanje nepotrebnih direktorija i datoteka nastalih za vrijeme prethodnog procesa naredbama: `rm grassdata -rf` i `rm grass_rivers.sh -f`.

Potom na red dolazi konačna vektorska generalizacija sa zaglađivanjem linija.

```
python3 ../generalize.py $OSM_SCALE 0 ./05_rivers/builded_rivers.shp
./05_rivers/rivers_final.shp
```

U toj naredbi `$OSM_SCALE` je varijabla koja za željeno mjerilo povlači podatak koji smo unijeli na početku procesa izrade karte. Sintaksa dohvaćanja u *generalize.py Python* programu je sys.argv[1]. `0` je granična vrijednost, u milimetrima kvadratnim u mjerilu karte, ispod koje će svi objekti biti izbrisani. Ova vrijednost nam ne treba jer ćemo sad kroz proces provesti samo linije i zato je vrijednost 0. Sintaksa dohvaćanja ove vrijednosti je sys.argv[2]. `./05_rivers/builded_rivers.shp` je naziv i lokacija datoteke s ulaznim podacima koje treba provesti kroz generalizaciju. `./05_rivers/rivers_final.shp` je naziv i lokacija izlazne datoteke u koju će se spremiti generalizirane rijeke.

Sada imamo završnu *Shapefile* datoteku za rijeke (Slika 5.10.) koja će se koristiti u konačnom izgledu karte. Objekti i njihova geometrija više neće biti dirani. Sad brišemo nepotrebne datoteke nastale prilikom izvršavanja ove izvršne skripte.

Slijedi postavljanje potrebnih stupaca u atributnoj tablici.

```
python3 ../set_fields.py osm_id,name,name_$OSM_LANG labels=yes
05_rivers/rivers_final.shp
```

U navedenoj naredbi `../set_fields.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, a `osm_id,name,name_$OSM_LANG` je popis stupaca koje ćemo zadržati u atributnoj tablici. U ovom slučaju to su osm_id, lokalno ime, ime na jeziku kojeg smo definirali na početku izrade ove karte u poglavlju 5.2.1. Sintaksa dohvaćanja ovog popisa u *set_fields.py Python* programu je sys.argv[1]. Opcija `labels=yes` govori da želimo dodatne stupce za ručno pomicanje i rotaciju naziva. Sintaksa dohvaćanja opcije je sys.argv[2]. `05_rivers/rivers_final.shp` je naziv i lokacija datoteke, u odnosu na radni direktorij, za koju ćemo napraviti promjene u atributnoj tablici.

Zadnji korak u izvršnoj skripti *_5_Rivers.sh* je kopiranje vrijednsti iz atributne tablice referentne karte.

```
python3 ../_8_copyfields.py _label_x,_label_y,_label_r
../PK/05_rivers/rivers_final.shp ../$OSM_DIR/05_rivers/rivers_final.shp
```

U toj naredbi `../_8_copyfields.py` je naziv i lokacija općeg *Python* programa koji će se izvršiti, a `_label_x,_label_y,_label_r` je popis stupaca u atributnoj tablici ulazne datoteke iz kojih ćemo kopirati vrijednosti u identične stupce atributne tablice datoteke s rijekama. U ovom slučaju to su x i y koordinata i rotacija. `../PK/05_rivers/rivers_final.shp` je naziv i lokacija datoteke s referentnim slojem iz čije atributne tablice će se kopirati podaci, a `../$OSM_DIR/05_rivers/rivers_final.shp` je naziv i lokacija datoteke u čiju atributnu tablicu će se zapisati kopirani podaci.

Izvršna skripta *_5_Rivers.sh* izvršena je do kraja, a dobili smo sloj s rijekama u Winkelovoj trostrukoj projekciji pohranjen u *Shapefile* datoteku *rivers_final.shp* (Slika 5.10). Ukupno je dobiveno 2030 objekta, a datoteka je veličine 1,1 MB.

Slika 5.10. Konačni sloj s rijekama u Winkelovoj trostrukoj projekciji (rivers_final.shp)

U originalnoj verziji izvršne skripte za rijeke u dijelu koji se odnosi na procesuiranje u *GRASS GIS*-u bile su korištene još dvije naredbe prije konverzije podataka iz vektorske mape u rastersku mapu.

```
v.db.addcolumn map=rivers_winkel columns=\"length_km double precision\"
```

Naredba iz prošlog reda je dodavala stupac „length_km“ u atributnu tablicu u vektorskoj mapi „rivers_winkel“. Sljedeća naredba je u toj vektorskoj mapi za linijske objekte računala duljinu i zapisivala je te vrijednosti u kilometrima u prethodno kreirani stupac „length_km“.

```
v.to.db map=rivers_winkel type=line option=length units=k columns=length_km --
overwrite
```

Vrijednosti duljina koje su tako dobivene su služile kod generalizacije za izbor rijeka koje će ući u završni izgled karte. Sada se za izostavljanje manjih pritoka i rijeka koristi gore opisana „v.clean“ funkcija u kojoj se prag filtera definira u odnosu na zadano mjerilo karte što je univerzalnije rješenje. Primjećeno je da podaci iz stupca „length_km“ nisu korišteni nigdje u procesu izrade karte, pa su navedena dva retka uklonjena iz izvršne skripte *_5_Rivers.sh*.

5.2.8 Izrada sloja s gradovima

Izvršna skripta *_6_Cities.sh* koristi se za dobivanje objektne skupine gradova. Originalna skripta je ispitana i pokazala se zadovoljavajućom.

Prema službenim smjernicama na *wiki* straniciama OSM-a svi gradovi bi trebali imati oznaku „place=city“(URL 75). Dakle, kod filtriranja podataka iz OSM *Planet* datoteke koristit će se navedena *oznaka*.

```
osmfilter ../planet.osm.o5m --keep= --keep-nodes="place=city" -o=osm_cities.osm
```

Navedena naredba iz OSM datoteke za čitavi svijet zadržava samo one čvorove koji imaju oznaku `"place=city"`. Filtriranjem smo dobili datoteku *osm_cities.osm* koju ćemo konvertirati u *Shapefile* koristeći se *ogr2ogr* modulom.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"SELECT * FROM points" -overwrite -skipfailures -lco ENCODING=UTF-8 06_cities
osm_cities.osm
```

U navedenoj naredbi korišteni *sql* upit povlači sve objekte iz sloja „points“ i sprema ih u *Shapefile* datoteku *points.shp* u direktorij *06_cities*. U toj datoteci su točkasti objekti koji predstvljaju sve gradove svijeta unešene u OSM. Za izdvajanje glavnih gradova koristi se poseban *Python* program *_6_1_admin_centers.py* koji će izraditi *osmfilter* naredbu i spremiti ju u *shell* skriptu *capitals.sh* koja se zatim izvršava. Spomenuti *Python* program će iz OSM XML datoteke *osm_countries.osm,* stvorene pri izradi sloja s političko-teritorijalnim entitetima, pronaći relacije s oznakama „ISO3166-1“ i „ISO3166-1:alpha2“ koje imaju identifikacijske brojeve čvorova „`ref`“ s ulogom administrativnog centra „`role="admin_centre"`“. Potom se identifikacijski brojevi čvorova koji zadovoljavaju ove uvjete spremaju u listu tako da se svakom doda prefiks „`@id=`“. Članovi te liste biti će vrijednosti koje će se zadati opciji `--keep-nodes=` u *osmfilteru*. Drugim riječima, *osmfilter* će izvući čvorove koji imaju te `@id` ključeve.

Slijedi popis naredbi koje u izvršnoj skripti izvršavaju gore navedeni postupak.

```
python3 ../_6_1_admin_centers.py osm_countries.osm >capitals.sh
chmod +x capitals.sh
./capitals.sh
```

Dobiveni podaci će se spremiti u datoteku naziva *osm_capitals.osm*. Potom slijedi konverzija u *Shapefile* format korištenjem *ogr2ogr* modula.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql
"SELECT * FROM points" -overwrite -skipfailures -lco ENCODING=UTF-8 06_capitals
osm_capitals.osm
```

Izlazna datoteka *points.shp* sprema se u direktorij *06_capitals.* Tu datoteku, kao i ranije dobivenu istoimenu datoteku sa svim gradovima, je potrebno reprojicirati u projekciju karte već spominjanim postupkom koristeći opći *Python* program *reproject.py*.

```
python3 ../reproject.py 06_cities/points.shp 06_cities/cities_final.shp
python3 ../reproject.py 06_capitals/points.shp 06_cities/capitals_final.shp
```

Tako su dobivene datoteke *cities_final.shp* sa svim gradovima svijeta i *capitals_final.shp* s glavnim gradovima država. Budući da u datoteci sa svim gradovima imamo i glavne gradove, a kako bi se izbjeglo dvostruko pojavljivanje na karti, potrebno je iz datoteke *cities_final.shp* izbrisati glavne gradove. Za to nam služi posebni *Python* program

_6_2_clean_cities.py koji ujedno služi i za generalizaciju i čišćenje, a pokreće se sljedećom naredbom:

```
python3 ../_6_2_clean_cities.py $OSM_SCALE 7 06_cities/capitals_final.shp
06_cities/cities_final.shp 500000.
```

U ovoj naredbi `$OSM_SCALE` je varijabla koja za željeno mjerilo povlači podatak koji smo unijeli na početku pri postavljanju parametara karte. Sintaksa dohvaćanja u *_6_2_clean_cities.py Python* programu je sys.argv[1]. Opcija `7` je vrijednost u milimetrima na karti unutar koje može biti prikazan samo jedan grad, onaj s najvećom populacijom (engl. *buffer*). Sintaksa dohvaćanja je sys.argv[2]. `500000` je vrijednost koja definira generalizaciju gradova prema broju stanovnika. Gradovi s manjim brojem stanovnika od zadanog će se izbrisati. Sintaksa dohvaćanja je sys.argv[5].

Slijedi brisanje nepotrebnih direktorija i datoteka, pa postavljanje potrebnih stupaca u atributnoj tablici za obje datoteke s gradovima.

```
python3 ../set_fields.py osm_id,name,name_$OSM_LANG,population labels=yes
06_cities/cities_final.shp
python3 ../set_fields.py osm_id,name,name_$OSM_LANG,population labels=yes
06_cities/capitals_final.shp
```

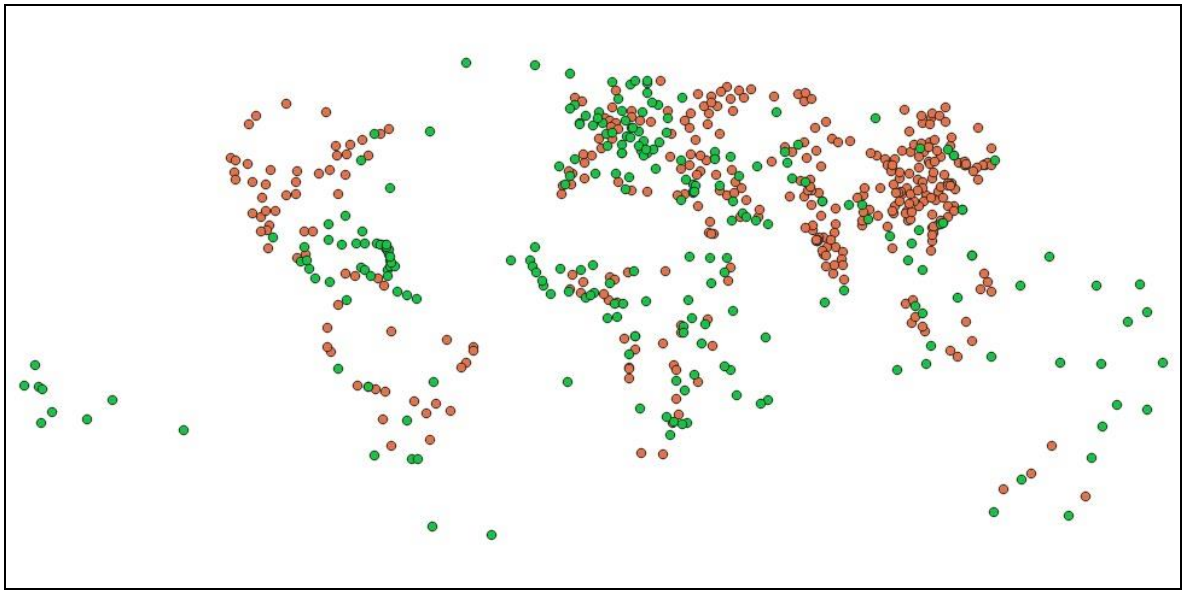
Popis stupaca koje ćemo zadržati u atributnim tablicama su: `osm_id`, `name`, `name_$OSM_LANG` i `population`, a `labels=yes` govori kako želimo da se dodaju stupci za ručno pomicanje i rotaciju naziva.

Posljednji korak je kopiranje vrijednsti iz atributne tablice referentne karte za obje datoteke s gradovima.

```
python3 ../_8_copyfields.py _label_x,_label_y ../PK/06_cities/cities_final.shp
../$OSM_DIR/06_cities/cities_final.shp

python3 ../_8_copyfields.py _label_x,_label_y,_label_r
../PK/06_cities/capitals_final.shp ../$OSM_DIR/06_cities/capitals_final.shp
```

S završetkom ovog koraka izvršna skripta *_6_Cities.sh* je stigla do kraja, a dobili smo slojeve s gradovima i glavnim gradovima država u Winkelovoj trostrukoj projekciji (Slika 5.11.).



Slika 5.11. Konačni slojevi s gradovima u Winkelovoj trostrukoj projekciji

5.2.9 Izrada sloja s oceanima, morima i zaljevima

Kao što smo već rekli u poglavlju 5.2.3, okvir karte biti će podloga iznad koje se stavljaju svi ostali slojevi. Mora, oceani i veći zaljevi neće biti prikazani geometrijom, nego će ih, tamo gdje nije kopno iznad njega, predstavljati upravo okvir karte. Iz tog razloga je okvir karte bit će u svijetlo plavoj boji. Ipak, potrebno je izvući točke nužne za vizualizaciju naziva tih morskih površina.

Prema *wiki* straniciama OSM-a svi oceani bi trebali imati oznaku place=ocean, mora oznake place=sea ili natural=sea, a zaljevi place=bay. U skladu s tim izvršna skripta *_7_Oceans.sh* će početi s filtriranjem OSM *Planet* datoteke.

```
osmfilter ../planet.osm.o5m --keep= --keep-nodes="place=ocean or place=sea or natural=sea or place=bay" -o=osm_oceans.osm
```

Filtriranjem smo dobili datoteku *osm_oceans.osm* koju ćemo konvertirati u *Shapefile* koristeći se *ogr2ogr* modulom.

```
ogr2ogr --config OSM_CONFIG_FILE ../osmconf.ini -f "ESRI Shapefile" -sql "SELECT * FROM points WHERE name != ''" -overwrite -skipfailures -nlt POINT -lco ENCODING=UTF-8 07_oceans osm_oceans.osm
```

U toj naredbi `-sql "SELECT * FROM points WHERE name != ''"` je *sql* upit koji iz sloja „points“ povlači sve objekte koji imaju upisano neko ime i za njih preuzima sve dostupne

oznake, a opcija `-nlt POINT` definira točke kao vrstu geometrije za kreirani sloj. `07_oceans` definira naziv i lokaciju izlaznog direktorija, a `osm_oceans.osm` je naziv i lokacija ulazne datoteke.

Tako je kreirana *points.shp* vektorska datoteka s točkama, koju potom reprojiciramo općim *Python* programom *reproject.py*. Navedeni *Python* program povlači podatak o zadanoj projekciji preko već definirane varijable `OSM_PROJ` i kroz proces konverzije provlači objekte iz datoteke *points.shp*. Nakon konverzije, točke su u Winkelovoj trostrukoj projekciji i spremaju se u datoteku naziva *oceans_final.shp* koja će se koristiti u konačnom izgledu karte.

Zatim brišemo nepotrebne datoteke nastale prilikom izvršavanja ove izvršne skripte i postavljamo potrebne stupace u atributnoj tablici datoteke *oceans_final.shp*.

```
python3 ../set_fields.py osm_id,name,name_$OSM_LANG labels=yes
07_oceans/oceans_final.shp
```

Posljednji korak je kopiranje vrijednosti za atributnu tablicu s referentne karte.

```
python3 ../_8_copyfields.py _label_x,_label_y,_label_r,size
../PK/07_oceans/oceans_final.shp ../$OSM_DIR/07_oceans/oceans_final.shp
```

U navedenoj naredbi `../PK/07_oceans/oceans_final.shp` je naziv i lokacija datoteke s referentnim slojem iz čije atributne tablice će se kopirati podaci, a `../$OSM_DIR/07_oceans/oceans_final.shp` je naziv i lokacija datoteke u čiju atributnu tablicu će se zapisati kopirani podaci.

Ovime je izvršna skripta *_7_Oceans.sh* izvršena do kraja, a dobili smo sloj s oceanima, morima i jezerima u Winkelovoj trostrukoj projekciji pohranjen u *Shapefile* datoteku *oceans_final.shp*. Ukupno je dobiveno 112 objekta, a datoteka je malene veličine (3,16 KB).

5.3 PRIPREMA KARTE ZA ISPIS

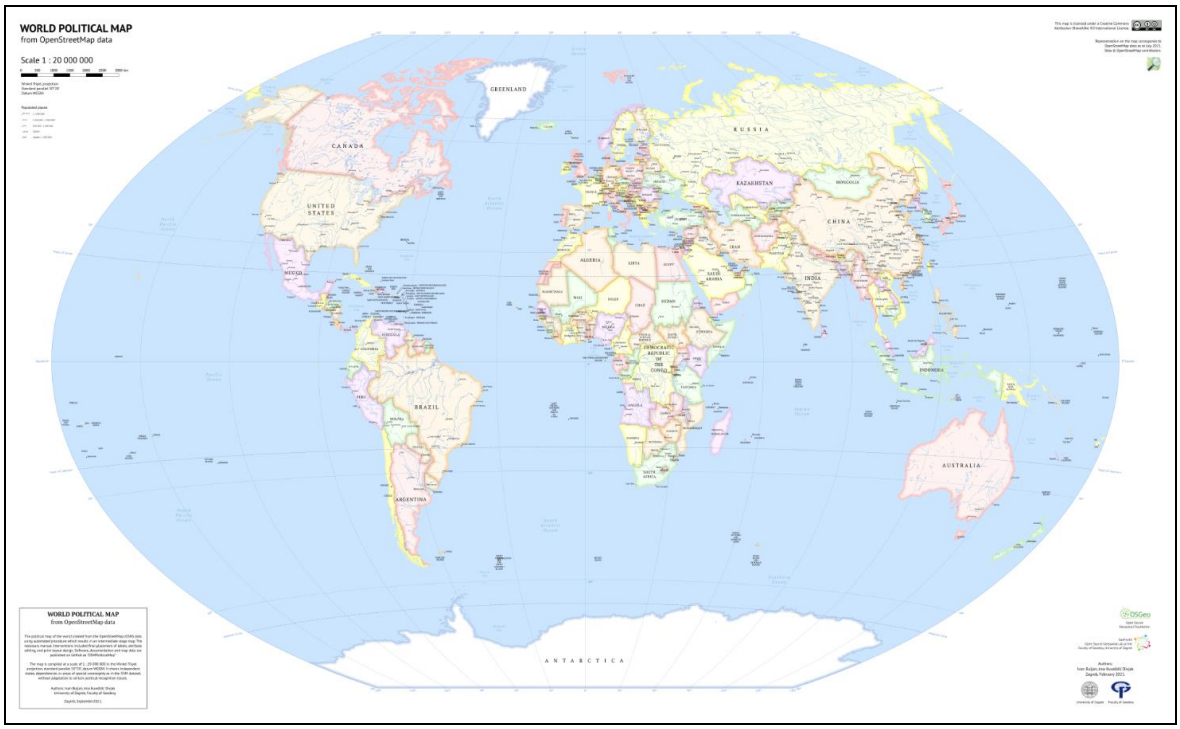
Nakon što je završio automatizirani postupak, pristupamo ručnom uređivanju karte i pripremi za ispis. Na našem računalu u direktoriju NEW_MAP nalazi se *QGIS* projekt *OSM_World_Political_Map.qgs* u kojem se učitavaju svi automatski kreirani slojevi. Osim tih slojeva učitava se i sloj s pomoćnim linijama za smještaj imena malih karipskih država koji je kopiran iz referentne karte, kao i datoteke koje nam trebaju na verziji za ispis, i

datoteke s preddefiniranim stilovima za oblikovanje pojedinih objektnih. Na taj način je maksimalno olakšan i ovaj manualni dio izrade karte, pa sve što korisnik treba je po potrebi ručno pomaknuti nazive gradova čiji smještaj nije zadan putem referentne karte. Broj gradova, kao i rijeka i jezera, ovisi o korištenim filterima i parametrima u automatiziranom procesu. Također, gradovi se šire i populacija u njima se povećava, stoga je moguće da nam se pojavi prevelik broj „novih" gradova u odnosu na referentnu kartu. Tada će ručno pomicanje biti nužno. Ukoliko ipak želimo smanjiti broj nekih objekata, jer ih ima previše s obzirom na mjerilo, moguće je njihovo ručno brisanje, ali je preporuka korištenje filtera unutar samog *QGIS*-a, kako podaci ne bi bili trajno izbrisani. Iz referentne karte su se tijekom automatiziranog dijela postupka prepisivali podaci za smještaj naziva (stupci „_label_x" i „_label_y" u atributnoj tablici), kut rotacije naziva („_label_r"), prijelom teksta naziva država i jezera („wrap") i kategorizaciju po veličini kod oceana i mora („size"). Prijelom teksta je preddefiniran za 18 država s dugačkim imenima odn. za one čije ime je maleno u odnosu na površinu. Kod sloja *oceans_final.shp* se prikazuju samo oni nazivi čiji objekti u atributnoj tablici imaju vrijednost u stupcu „size" (39 objekata) zbog korištenja preddefiniranog filtera za nazive „`CASE WHEN "size" is not null THEN "name_en" END`" u stilu samog sloja koji je učitan iz referentne karte. S obzirom da je cijeli postupak izvorno namijenjen izradi zidne političke karte, autori su na taj način spriječili prikazivanje manjih mora i zaljeva na karti svijeta. Nadalje, prema vrijednosti atributa „size" je određena i veličina samog naziva, pa tako npr. oceani imaju najveće nazive, a manja mora poput Baltičkog imaju najmanje nazive.

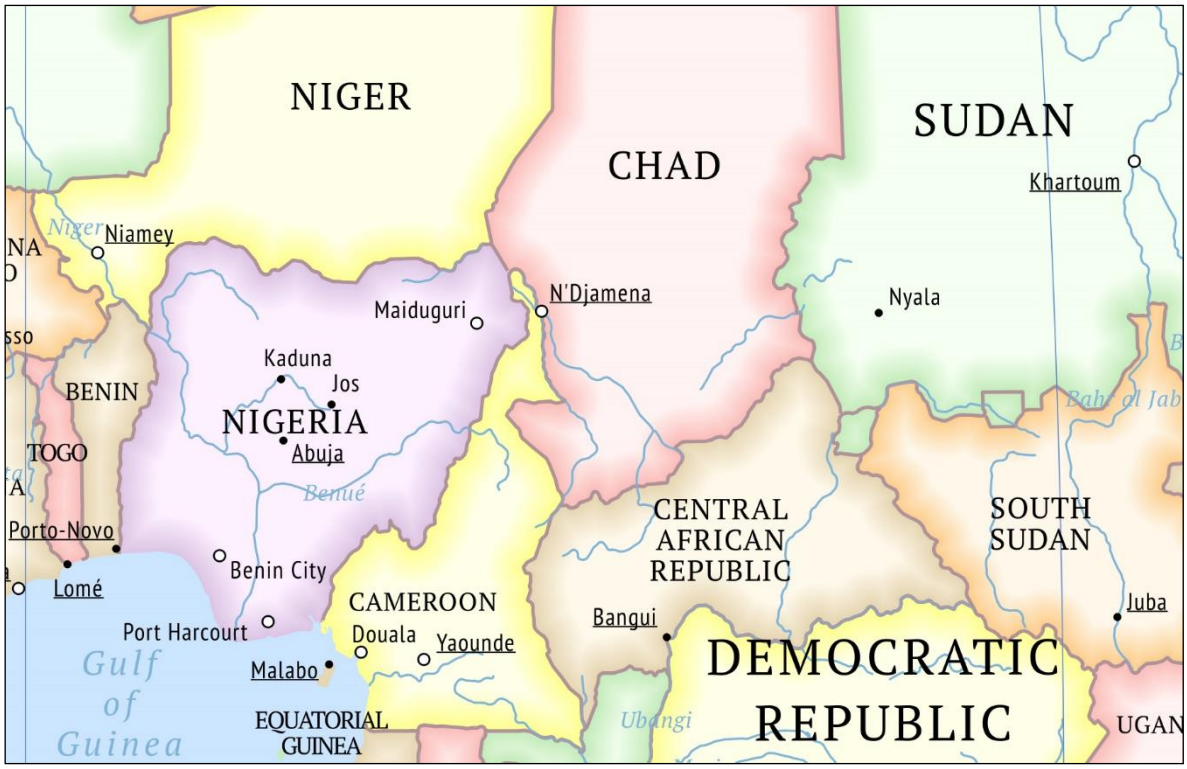
Što se tiče stilova za pojedine slojeve, autori su sve potrebne vrijednosti poput veličine fonta za nazive, razmaka između slova i riječi, debljina linija, itd., zadali preko varijabli, atributa samih objekata i u jedinicama karte (engl. *map units*), pa jedino za što se korisnik mora pobrinuti je da na svoje računalo preuzme i instalira fontove „PT Serif" (URL 76), „PT Sans" (URL 77) i „PT Sans Narrow" (URL 78) koji se koriste za nazive i kojima je prilagođen smještaj naziva i izgled karte općenito.

Priprema za ispis karte je obavljena u *QGIS*-ovom sučelju *Print Layout*, a predložen je preddefinirani izgled (engl. *layout*) pod nazivom „print_1_30000000" sa mjerilom 1:30 000 000 kojeg je moguće urediti po svojim potrebama ili od njega kreirati novi s obzirom da su

svi elementi vanjskog opisa karte (naslov, mjerilo, legenda, projekcija, autori, mjesto i vrijeme izrade, izvori podataka, licenca, institucije, projekti i opis procesa izrade) već učitani. S obzirom da je mjerilo karte koju izrađujemo 1:20 000 000 napravljen je novi izgled pod nazivom „print_1_20000000“ u kojem je pripremljena karta za ispis. Konačni izgled karte spremne za ispis je prikazan na Slika 5.12.



Slika 5.12. Konačni izgled političke karte svijeta iz podataka OpenStreetMapa

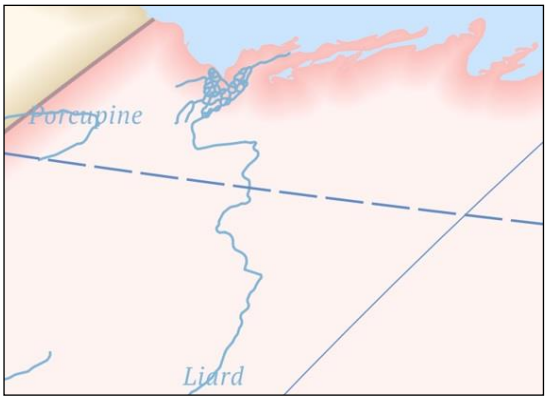


Slika 5.13. Detalj karte u izvornom mjerilu 1:20 000 000

6. REZULTATI I RASPRAVA

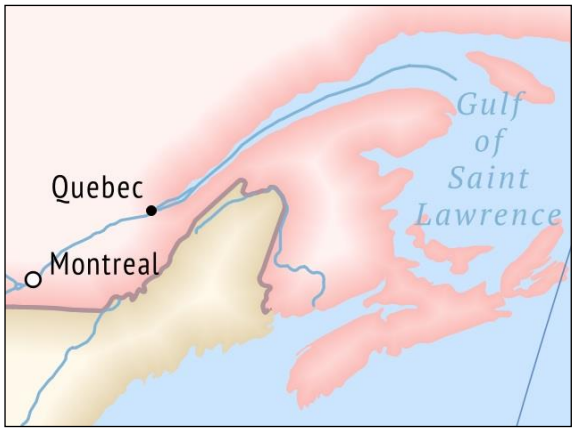
Evaluaciju ostvarenih rezultata najbolje je sagledati u odnosu na ciljeve postavljene na početku ovog rada. Može se reći da su zadani ciljevi u najvećoj mjeri ostvareni. Uspješno je iz podataka *OpenStreetMapa* izrađena zidna politička karta svijeta za 2021. godinu u mjerilu 1:20 000 000. Testirana je funkcionalnost softvera i postupka te su napravljene neke nužne promjene i unaprijeđenja da bi proces automatske izrade i dalje bio moguć.

Kad je riječ o kvaliteti i koherentnosti OSM podataka podataka, primjećen je napredak u odnosu na razdoblje kad je izrađen originalni softver. Svakako najistaknutiji primjer je jezero Huron koje se tretiralo kao specijalni slučaj i za čije izdvajanje iz OSM podataka je bio potreban poseban dio u izvršnoj skripti, a sada se uspješno izdvaja zajedno s drugim jezerima. Analizom podataka koje su autori dobili 2016. godine (Jogun, 2016) i usporedbom s dobivenim podacima u ovom postupku primjećeno je da su rijeke i dalje objektna skupina s najviše vidljivih nedostataka. Iako su ukupno gledajući rijeke kvalitetnije iscrtane u OSM-u i s manje nepovezanih dijelova, njihova generalizacija i dalje ponekad daje nezadovoljavajuće rezultate poput deltastog ušća rijeke Mackenzie (Slika 6.1.). Ipak, većina takvih pogrešaka zbog sitnog mjerila nije toliko očigledna.



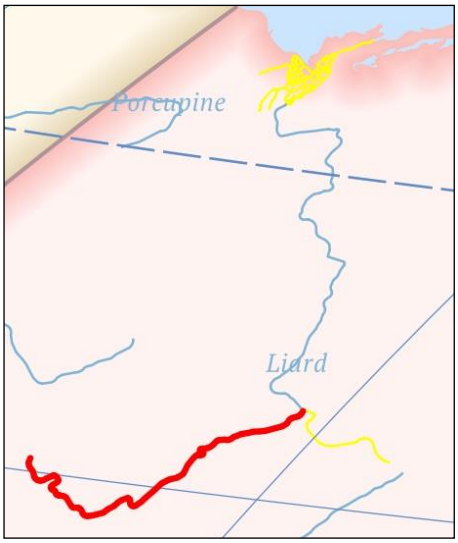
Slika 6.1. Pogreške generalizacije u sloju s rijekama

Postoji još jedan problem kod riječnih ušća, ovaj put kod estuarija gdje je teško odrediti granicu između rijeke i zaljeva tj. mora, pa na takvim mjestima izgleda kao da rijeka ide preko mora (Slika 6.2.). Nasuprot tome, postoje slučajevi gdje rijeke koje se ulijevaju u more nisu iscrtane do same obalne crte, što može biti prouzrokovano generalizacijom.



Slika 6.2. Pogreška u sloju s rijekama prouzročena OSM podacima

Kontrolom naziva rijeka uočeno je da su nekim rijekama dodijeljena pogrešna imena. Zaključak je da se greške javljaju nakon obrade sloja s rijekama u *GRASS GIS*-u, najvjerojatnije prilikom izvršavanja naredbe „v.build.polylines“ u situacijama kad se jedna rijeka ulijeva u drugu. Slika 6.3. prikazuje takvu situaciju gdje se u rijeku Mackenzie (početak i delta rijeke označeni žutom bojom na slici) ulijeva rijeka Liard (označena crvenom bojom na slici). Srednji dio rijeke Mackenzie (označen plavom bojom na slici) je pogrešno pripojen rijeci Liard kao njen nastavak, što je vidljivo i po nazivu „Liard“. Imajući na umu kako prikaz rijeka na političkim kartama nije obavezan i zbog svega navedenog bilo bi preporučljivo ne prikazivati imena rijeka na karti ako su rijeke dobivene ovakvim postupkom. S druge strane, postupak u *GRASS GIS*-u koji je korišten za kreiranje sloja s rijekama je i dalje najbolji način za dobivanje geometrije rijeka.



Slika 6.3. Pogrešno dodijeljeno ime rijeci Mackenzie

Ovakve situacije nam iznova nameću pitanje kontroliraju li automatski alati topološku konzistentnost podataka i koliko su u tome uspješni.

S obzirom da se mnogi podaci kopiraju iz referentne karte, bilo bi dobro da se ona ažurira i nadopuni tako da se u njoj većem broju objekata zadaju koordinate i rotacija za nazive jer što se krupnija mjerila koriste pri izradi, to će se prikazivati veći broj objekata koji nisu u referentnoj karti. Od ostalih nadogradnji bilo bi dobro da se doda sloj sa sjenčanim reljefom kakav ima karta T. Pattersona analizirana u poglavlju 2.1.1. Svakako bi trebalo otkloniti opisanu pogršku s nazivima nekih rijeka nakon procesa u *GRASS GIS*-u. Za filtraciju gradova bi se mogle dati preporuke u skladu s potencijalnim mjerilima koje bi korisnici mogli koristiti. *Shapefile* je pokazao određene manjkavosti po pitanju maksimalne veličine datoteke, a kako se očekuje nastavak povećanja količine podataka u OSM-u, moglo bi se razmisliti o prilagodbi cijelog automatiziranog procesa korištenju drugog formata, npr. *GeoPackage*.

7. ZAKLJUČAK

Na kraju ovog rada doneseni su zaključci temeljeni na početnim hipotezama.

H1 – Moguće je, u skladu s kartografskim načelima, visoko automatiziranim postupkom izraditi zidnu političku kartu svijeta za 2021. godinu iz podataka OpenStreetMapa u mjerilu 1:20 000 000.

Hipoteza je potvrđena. Izrađena je politička karta svijeta u zadanom mjerilu u Winkelovoj trostrukoj projekciji. Postupak izrade je u najvećem dijelu izveden automatski, korištenjem preuzetog softvera. Tako izrađena karta je u skladu s kartografskim načelima i dostupnim smjernicama za izradu političke karte.

H2 – Potrebno je provesti postupka ažuriranja i unaprijeđenja postojećeg softvera.

Hipoteza je potvrđena. S obzirom da se proces izvodio u zadnjim verzijama potrebnih programa, dijelovi preuzetog softvera za izradu karte su bili zastarjeli. Pojedine dijelove kôdova je trebalo prilagoditi novim verzijama programa *Python* i *GRASS GIS*.

H3 – Moguće je pojednostavniti postupak zbog bolje kvaltete OpenStreetMap podataka.

Hipoteza je odbačena. Iako imamo primjer poboljšanja kvalitete podataka OSM-a u slučaju jezera Huron koje se sad uspješno izdvaja s ostalim jezerima i ne zahtjeva poseban dio kôda u izvršnoj skripti, što možemo smatrati poboljšanjem, još uvijek je u procesu prisutan velik broj tzv. specijalnih slučajeva koji zahtjevaju posebne softverske tretmane. Tako imamo cijeli niz posebnih *Python* programa pri izradi najvažnijeg sloja – onog s političko-teritorijalnim entitetima. Jedan od glavnih uzroka za to je miješanje administrativnih razina pojedinih entiteta u OSM-u, odnosno njihova neusklađnost, zbog čega su autori odlučili da će se prikazati entiteti koji imaju ISO 3166-1 kôd.

Proces izrade karte koji smo prošli kroz ovaj rad je izvrstan primjer servisno orijentirane kartografije gdje se u vrlo automatiziranom procesu izrađuje karta iz podataka koje nam pruža određeni servis. Nakon cijelog postupka zaključujemo da nikad neće biti moguće automatiziranim procesom izraditi savršenu kartu ukoliko podaci nisu konzistentni i da se kvaliteta karte izrađene u automatiziranom procesu podiže s brojem ručnih intervencija zbog čega je neosporna uloga kartografa u kartografiji, pa čak i servisno orijentiranoj.

LITERATURA

Jobst, M., Gartner, G. (2019): Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management—An Introduction, Springer International Publishing, Basel.

Eckert, M. (1925): Kartenwissenschaft. Forschungen und Grundlagen zu einer Kartographie als Wissenschaft, Zweiter Band, Walter de Gruyter & Co, Berlin i Leipzig.

Frančula, N. (2003) Kartografska generalizacija, skripta, Geodetski fakultet, Zagreb.

Frančula, N. (2004): Kartografske projekcije, skripta, Geodetski fakultet, Zagreb.

Frangeš, S. (2019) Sažetak predavanja iz kolegija Tematska kartografija - 2. dio, Geodetski fakultet, Zagreb.

Frangeš, S. (2019): folije s predavanja iz kolegija Tematska kartografija, Geodetski fakultet, Zagreb.

Jogun, T. (2016): Izrada političke karte svijeta iz podataka openstreetmapa, diplomski rad, Geodetski fakultet, Zagreb.

Neumann, J. (1997): Enzyklopädisches Wörterbuch Kartographie in 25 Sprachen, 2. erweiterte Ausgabe, Saur, München, 462.

Patterson, T. (2010): Outside the Bubble: Real-world Mapmaking Advice for Students, Cartographic Perspectives 65, 7–15.

Raisz, E. (1948): General cartography, McGraw-Hill Series in Geography (2nd Ed.). New York.

Raisz, E. (1962): Principles of Cartography, McGraw-Hill Book Company, New York, London.

Stilinović, S. (2013.): Prikupljanje i izvoz OpenStreetMap podataka, diplomski rad, Geodetski fakultet, Zagreb, https://bib.irb.hr/datoteka/658621.Diplomski_rad_sstilinovic.pdf, (28.09.2021.)

Tutić, D., Lapaine, M. (2009): Kartografska generalizacija linija sa svojstvom čuvanja površina, Kartografija i geoinformacije, Vol. 8 No. 11, Hrvatsko kartografsko društvo, Zagreb, https://hrcak.srce.hr/file/62110 (02.09.2021.)

POPIS URL-ova

URL 1. https://icaci.org/mission/ (30.08.2021.)

URL 2. https://www.openstreetmap.org/ (30.08.2021.)

URL 3. https://github.com/GEOF-OSGL/OSMPoliticalMap (30.08.2021.)

URL 4. http://struna.ihjj.hr/naziv/politicka-karta/1911/#naziv (30.08.2021.)

URL 5. https://www.natgeomaps.com/re-world-classic (30.08.2021.)

URL 6. https://www.mapsinternational.co.uk/pub/media/catalog/product/x/n/a/national-geographic-world-classic-map_ng01093.jpg (30.08.2021.)

URL 7. https://www.mapsinternational.co.uk/pub/media/catalog/product/x/n/a/national-geographic-world-classic-map_ng01093.jpg (30.08.2021.)

URL 8. http://www.shadedrelief.com/political/ (31.08.2021.)

URL 9. http://www.shadedrelief.com/political/Political_Map_NE2.jpg (31.08.2021.)

URL 10. http://www.shadedrelief.com/political/Political_Map_NE2.jpg (31.08.2021.)

URL 11. https://www.cia.gov/the-world-factbook/static/94795a76638f6385b051647ac2ac1c4b/world_pol.jpg (31.08.2021.)

URL 12. https://www.cia.gov/the-world-factbook/static/94795a76638f6385b051647ac2ac1c4b/world_pol.jpg (31.08.2021.)

URL 13. https://www.cantorsparadise.com/the-four-color-theorem-8eece6ab6b12 (31.08.2021.)

URL 14. https://commons.wikimedia.org/wiki/File:World_map_with_four_colours.svg#/media/File:World_map_with_four_colours.svg (31.08.2021.)

URL 15. https://en.wikipedia.org/wiki/Mercator_projection#/media/File:Mercator_projection_Square.JPG (31.08.2021.)

URL 16. http://www.shadedrelief.com/NE2_proj/ (31.08.2021.)

URL 17. https://www.cartography.at/jobstm/ (05.09.2021.)

URL 18. https://dl.acm.org/doi/10.1145/3220228.3220261 (05.09.2021.)

URL 19. https://www.gartner.com/en/information-technology/glossary/service-oriented-architecture-soa (05.09.2021.)

URL 20. https://www.oracle.com/technical-resources/articles/javase/soa.html (05.09.2021.)

URL 21. https://www.svgroup.hr/rjesenja/service-oriented-architecture-soa/ (05.09.2021.)

URL 22. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap (05.09.2021.)

URL 23. https://www.openstreetmap.org/stats/data_stats.html (05.09.2021.)

URL 24. https://wiki.openstreetmap.org/wiki/File:Osmdbstats1_users.png (06.09.2021.)

URL 25. https://wiki.openstreetmap.org/wiki/FAQ#Is_it_OpenStreetMap_or_Open_Street_Maps.3F (06.09.2021.)

URL 26. https://www.bing.com/maps (06.09.2021.)

URL 27. https://www.google.com/maps/ (06.09.2021.)

URL 28. https://www.waze.com/hr/live-map?utm_source=waze_website&utm_campaign=waze_website&utm_medium=website_menu (06.09.2021.)

URL 29. https://www.mapquest.com/ (06.09.2021.)

URL 30. https://wiki.openstreetmap.org/wiki/History_of_OpenStreetMap#Founding_and_Early_History (06.09.2021.)

URL 31. https://wiki.openstreetmap.org/wiki/Yahoo!_Aerial_Imagery (06.09.2021.)

URL 32. https://wiki.openstreetmap.org/wiki/File:Active_contributors_month.png (06.09.2021.)

URL 33. https://wiki.openstreetmap.org/wiki/Web_front_end#Permissions_and_roles (06.09.2021.)

URL 34. https://wiki.openstreetmap.org/wiki/Quality_assurance (07.09.2021.)

URL 35. https://wiki.openstreetmap.org/wiki/OSM_Inspector (07.09.2021.)

URL 36. https://wiki.openstreetmap.org/wiki/Osmose (07.09.2021.)

URL 37. https://wiki.openstreetmap.org/wiki/Coastline_error_checker (07.09.2021.)

URL 38. http://tools.geofabrik.de/osmi/?view=coastline (07.09.2021.)

URL 39. https://wiki.openstreetmap.org/wiki/Detect_Vandalism (07.09.2021.)

URL 40. https://wiki.openstreetmap.org/wiki/Elements#Elements (07.09.2021.)

URL 41. https://wiki.openstreetmap.org/wiki/Node (07.09.2021.)

URL 42. https://wiki.openstreetmap.org/wiki/Way (07.09.2021.)

URL 43. https://wiki.openstreetmap.org/wiki/Elements#Relation (07.09.2021.)

URL 44. https://wiki.openstreetmap.org/wiki/Relation:multipolygon (07.09.2021.)

URL 45. https://wiki.openstreetmap.org/wiki/PBF_Format (07.09.2021.)

URL 46. https://wiki.openstreetmap.org/wiki/O5m (07.09.2021.)

URL 47. https://wiki.openstreetmap.org/wiki/Osmfilter (07.09.2021.)

URL 48. https://wiki.openstreetmap.org/wiki/OGR (07.09.2021.)

URL 49. https://gdal.org/drivers/vector/osm.html#osm-openstreetmap-xml-and-pbf (07.09.2021.)

URL 50. https://gdal.org/programs/ogr2ogr.html (07.09.2021.)

URL 51. https://github.com/OSGeo/gdal/blob/master/gdal/data/osmconf.ini (07.09.2021.)

URL 52. https://gdal.org/drivers/vector/osm.html#configuration (07.09.2021.)

URL 53. https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html (07.09.2021.)

URL 54. https://www.python.org/ (03.09.2021.)

URL 55. https://www.qgis.org/en/site/ (03.09.2021.)

URL 56. https://www.osgeo.org/projects/qgis/ (03.09.2021.)

URL 57. https://grass.osgeo.org/ (03.09.2021.)

URL 58. https://grass.osgeo.org/about/ (03.09.2021.)

URL 59. https://github.com/GEOF-OSGL/OSMPoliticalMap#readme (03.09.2021.)

URL 60. https://wiki.ubuntu.com/FocalFossa/ReleaseNotes#Python3_by_default (03.09.2021.)

URL 61. https://wiki.openstreetmap.org/wiki/Planet.osm (03.09.2021.)

URL 62. https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d450_polaznik.pdf (23.08.2021.)

URL 63. https://wiki.openstreetmap.org/wiki/Coastline (01.09.2021.)

URL 64. https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry#Geometric_objects (01.09.2021.)

URL 65. https://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative (01.09.2021.)

URL 66. https://raw.githubusercontent.com/datasets/country-list/master/data.csv (01.09.2021.)

URL 67. https://github.com/tyrasd/osmtogeojson (01.09.2021.)

URL 68. https://github.com/tyrasd/osmtogeojson (01.09.2021.)

URL 69. https://wiki.openstreetmap.org/wiki/Tag:water%3Dlake (27.08.2021.)

URL 70. https://wiki.openstreetmap.org/wiki/Tag:natural%3Dwater (28.08.2021.)

URL 71. https://taginfo.openstreetmap.org/tags/natural=water#wiki (26.08.2021.)

URL 72. https://wiki.openstreetmap.org/wiki/Rivers (29.08.2021.)

URL 73. https://wiki.openstreetmap.org/wiki/File:Make_river.png (29.08.2021.)

URL 74. https://grass.osgeo.org/grass78/manuals/r.mapcalc.html (29.08.2021.)

URL 75. https://wiki.openstreetmap.org/wiki/Tag:place%3Dcity (04.09.2021.)

URL 76. https://fonts.google.com/specimen/PT+Serif (05.09.2021.)

URL 77. https://fonts.google.com/specimen/PT+Sans?query=PT+Sans (05.09.2021.)

URL 78. https://fonts.google.com/specimen/PT+Sans+Narrow (05.09.2021.)

POPIS SLIKA

PRILOZI

Prilog 1. Sadržaj priloženog optičkog medija:

Direktorij/datoteka	Opis
Diplomski_Buljan.pdf	Diplomski rad u pdf formatu
Diplomski_Buljan.docx	Diplomski rad u docx formatu
World_Political_Map_2021.pdf	Izrađena karta u pdf formatu
OSMPoliticalMap_GitHub	Direktorij s originalnim softverom preuzetim s digitalnog repozitorija Github
OSMPoliticalMap	Direktorij cijelog projekta uključujući: • ažurirane verzije izvršnih skripti i *Python* programa • direktorij „PK“ s referentnom kartom *Reference_OSMPoliticalMap.qgs* i svim podacima potrebnim za njeno generiranje • direktorij „NEW_MAP“ s poddirektorijima svih dobivenih objektnih skupina i *QGIS* projektom *OSM_World_Political_Map.qgs* u kojem je izrađena karta
Fonts	Direktorij s fontovima potrebnim za izradu karte