Generiranje tlocrta zgrada iz podataka zračnog laserskog skeniranja korištenjem DBSCAN i Alpha Shape algoritama

Krajtner, Marija

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:725114

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-24**

Repository / Repozitorij:

University North Digital Repository

Sveučilište
Sjever

Završni rad br. 005/GIG/2024

Generiranje tlocrta zgrada iz podataka zračnog snimanja korištenjem DBSCAN i Alpha Shape algoritma

Marija Krajtner, 0336054511

Varaždin, lipanj 2024. godine

Sveučilište
Sjever

Odjel za Ime odjela

Završni rad br. 005/MMGIG/2024

Generiranje tlocrta zgrada iz podataka zračnog laserskog snimanja korištenjem DBSCAN i Alpha Shape algoritma

Student

Marija Krajtner, 0336054511

Mentor

Hrvoje Matijević,doc.dr.sc

Varždin, lipanj 2024. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Geodezija i geomatika

STUDIJ Sveučilišni prijediplomski studij geodezije i geomatike

PRISTUPNIK Marija Krajtner | MATIČNI BROJ 0336054511

DATUM 03.07.2024. | KOLEGIJ Geoinformacijski sustavi

NASLOV RADA Generiranje tlocrta zgrada iz podataka zračnog laserskog skeniranja korištenjem DBSCAN i Alpha Shape algoritama

NASLOV RADA NA ENGL. JEZIKU Generating building footprints from aerial laser scanning data using DBSCAN and Alpha Shape algorithms

MENTOR Hrvoje Matijević | ZVANJE Doc. dr. sc., znanstveni suradnik

ČLANOVI POVJERENSTVA

1. Prof. dr. sc. Vlado Cetl, predsjednik povjerenstva
2. Doc. dr. sc. Hrvoje Matijević, mentor
3. Doc. dr. sc. Nikola Kranjčić, član
4. Izv. prof. dr. sc. Danko Markovinović, zamjenski član
5.

Zadatak završnog rada

BROJ 005/GIG/2024

OPIS

Cilj završnog rada je iz oblaka točaka dobivenog zračnim laserskim skeniranjem generirati tlocrte zgrada. Za provedbu zadatka će se koristiti Python programski jezik, a tlocrte zgrada će se generirati prvo klasteriranjem točaka svake pojedine zgrade korištenjem DBSCAN algoritma. Za svaki tako dobiveni klaster će se generirati 2D poligon primjenom Alpha Shape algoritma na sve točke predmetnog klastera. Na kraju će se provesti vizualna analiza dobivenih rezultata te zaključiti o primijenjivosti implementiranog pristupa.

REPUBLIKA HRVATSKA 4 SVEUČILIŠTE SJEVER

ZADATAK URUČEN 15.04.2024. | POTPIS MENTORA

UNIVERSITY NORTH

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU

Završni/diplomski/specijalistički rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, MARIJA KRAJTNER *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog/specijalističkog *(obrisati nepotrebno)* rada pod naslovom GENERIRANJE TLOCRTA ZGRADA IZ PODATAKA OBLAKOM SNIMANJA KORIŠTENJEM DBSCAN I ALPHA SHAPE ALGORITMA *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Marija Krajtner
(vlastoručni potpis)

Sukladno članku 58., 59. i 61. Zakona o visokom obrazovanju i znanstvenoj djelatnosti završne/diplomske/specijalističke radove sveučilišta su dužna objaviti u roku od 30 dana od dana obrane na nacionalnom repozitoriju odnosno repozitoriju visokog učilišta.

Sukladno članku 111. Zakona o autorskom pravu i srodnim pravima student se ne može protiviti da se njegov završni rad stvoren na bilo kojem studiju na visokom učilištu učini dostupnim javnosti na odgovarajućoj javnoj mrežnoj bazi sveučilišne knjižnice, knjižnice sastavnice sveučilišta, knjižnice veleučilišta ili visoke škole i/ili na javnoj mrežnoj bazi završnih radova Nacionalne i sveučilišne knjižnice, sukladno zakonu kojim se uređuje umjetnička djelatnost i visoko obrazovanje.

Zahvala

Želim se istinski zahvaliti svima koji su bili uz mene tijekom izrade ovog završnog rada. Prvo, zahvaljujem se svojoj obitelji na neizmjernoj podršci i razumijevanju tijekom školovanja te na svemu što ste mi omogućili kako bih bila u mogućnosti postignuti ovaj uspjeh. Nadalje se zahvaljujem svojim prijateljima, kolegama i partneru koji su mi pružili najveće ohrabrenje tijekom ovog putovanja. Hvala vam što ste bili tu, za sve savjete, riječi ohrabrenja i najvažnije za sve trenutke smijeha koje smo dijelili. Posebno se zahvaljujem svome mentoru Hrvoju Matijeviću čije je stručnost i strpljenje bilo ključno pri izradi ovog rada. Hvala Vam što ste uvijek bili dostupni za sva pitanja i nedoumice. Vaši savijati su bili od neizmjerne važnosti za završetak ovog rada. Hvala svima što ste mi pomogli ostvariti svoj cilj.

Sažetak

Rad objašnjava osnovne pojmove vezane uz podatke oblaka točaka, moguće tehnologije i alate za obradu takvih podatka, te formate pohranjivanja. Detaljnije opisuje generalizaciju poligona iz podataka zračnog laserskog skeniranja, metode i algoritme korištene za obradu. Pruža objašnjenja pojmova DBSCAN algoritma, Alpha Shape-a. Prikazuje uvid u pregled koda generiranja poligon opisujući varijable i korištene funkcije napisane Python jezikom. Sadrži vizualizaciju generiranih poligona Qgis softverom i analizu rezultata, kao i zaključke donesene na osnovu provođenja programskog zadatka.

Ključni pojmovi: oblak točaka, lasersko skeniranje, DBSCAN algoritam, Alpha Shape, Python, GeoJson.

Abstract

The paper explains basic terms related to point cloud data, possible technologies and tools for processing such data, and storage formats. It describes in more detail the generalization of polygons from aerial laser scanning data, the methods and algorithms used for processing. Provides explanations of DBSCAN algorithm concepts, Alpha Shape. It shows an overview of the polygon generation code describing the variables and functions used written in the Python language. It contains the visualization of polygons in Qgis software and the analysis of the results, as well as the conclusions reached based on the implementation of the programming task.

Key terms: point cloud, laser scanning, DBSCAN algorithm, Alpha Shape, Python, GeoJson.

Popis korištenih kratica

DBSCAN	Density based spatial clustering of application with noise
LiDAR	Light Detection and Ranging
GIS	Geographic information system
GNU	General Public License
OSGeo	Source Geospatial Foundation
BSD	Berkeley Software Distribution
GEOS	Geometry Engine - Open Source
JTS-u	Java Topology Suite
JSON	JavaScript Object Notation
LAS	LASer
PLY	Polygon File Format
VLRs	Variable remote recording
EVLRs	Extended variable length record
DDR	Direct density-reachable
DR	Density reachable
DC	Density-connectedness
AHN4	Actueel Hoogtebestand Nederland 4
WMS-a	Web Map Service

1. Sadržaj

2. Uvod

Razvoj tehnologije i računalnih znanosti u bližoj prošlosti otvorilo je vrata automatizaciji brojnih procesa koji su nekada zahtijevali dugotrajan i složen rad. Jedan od takvih procesa je i generiranje tlocrta zgrada, koje se tradicionalno veže uz geodeziju i arhitekturu. Napredak u metodama računalne geometrije i strojnog učenja omogućuje da ovaj proces postane brži, precizniji i dostupniji široj javnosti. Javna dostupnost podataka dodatno doprinosi ovom trendu, omogućujući svakome s osnovnim znanjem o računalnim alatima da stvara vlastite podatke tlocrta.

Cilj ovog rada je istraživanje mogućnosti automatskog generiranja poligona, odnosno tlocrta zgrada, korištenjem metoda računalne geometrije. U fokusu je uporaba DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algoritma za strojno učenje, koji omogućuje učinkovito grupiranje podataka prepoznajući guste regije unutar skupa podataka. Kombiniranjem DBSCAN algoritma s metodama računalne geometrije, kao što su Alpha Shape i konkavna ljuska, moguće je odrediti geometrijske oblike poput poligona i njihovih granica, što omogućuje vjerniji prikaz stvarnog stanja objekata.

Rad također razmatra povezane tehnologije za obradu podataka oblaka točaka, te programske i softverske alate koji olakšavaju proces izrade tlocrta. U teorijskom dijelu opisuju se različiti alati i biblioteke koje pojednostavljuju proces programiranja, kao i različite metode te se obrađuju formati za spremanje podataka.

Praktični dio rada detaljno prikazuje postupak generiranja poligona korištenjem programskog jezika Python. Kroz isječke koda, opisuje se korištenje funkcija i biblioteka, priprema podataka za obradu, način učitavanja oblaka točaka, prikaz klasteriranja pomoću DBSCAN algoritma, te proces generiranja Alpa Shape i konveksne ljuske. Na kraju se opisuje zapis rezultata i njihova vizualizacija.

Vizualnom analizom rezultata generiranja poligona pruža se pregled mogućih primjena ovih podataka, te se razmatraju problemi i ograničenja koja mogu nastati pri provedbi ove metode. Posebna pažnja posvećena je praktičnosti i efikasnosti metoda, te mogućnosti njihove primjene u različitim scenarijima. Cilj rada je pokazati potencijal naprednih metode računalne geometrije koje mogu transformirati proces izrade tlocrta zgrada čineći ga pristupačnijim i učinkovitijim.

3. Obrada zadatka

3.1. Uvod u obradu oblaka točaka

Oblaci točaka su točke s trodimenzionalnim koordinatama u prostoru obično prikupljene instrumentima i metodama kao što su to LiDAR (Light Detection and Ranging), fotogrametrija, 3D skeneri. Točke na površinama objekata tvore 3D model mjerenog područja skeniranja. Ova vrsta podataka koristi se u različite svrhe te je prisutna u područjima kao što su to geodezija, arhitektura, 3D modeliranja, računarstvo. Princip prikupljanja oblaka točaka je emitiranje zrake svjetlosti s instrumenta odnosno laserskog skenera te mjerenje vremena potrebnog da se emitirana zraka odbije od objekta i vrati natrag do instrumenta, na osnovu izmjerenog vremena izračunava se udaljenost do objekta. Ova metoda prikupljanja podataka ima prednosti ali i mane. Prednosti su brzina i jednostavnost prikupljanja, detaljnost u prikupljanju podataka i široka primjena ovako prikupljenih podataka. Dok su mane veličina odnosno količina prikupljenih podataka što zahtjeva jaku računalnu podršku te je često potrebna dodatna obrada ovakvih podatak u obliku filtriranja i segmentacije kako bi se osigurala kvaliteta i relevantnost podataka. Iz područja geodezije podatke oblaka točaka moguće je koristiti pri 3D modeliranju gradova, izradi digitalnih modela reljefa, kartiranju, praćenju promjena u okolišu, također moguće je i primijeniti ove podatke pri definiranju parcela ali i drugih elementa u prostoru kao što su to linijska infrastruktura, zgrade, te slični elementi.

Zbog široke uporabe podatka širok je raspon metoda obrade oblaka točaka s ciljem dobivanje relevantnih informacija. Filtriranje podatka je metoda koja se koristi pri uklanjanju šumova iz podatka ali i pri smanjenju broja točaka postavljanjem uvjeta kako bi se dobile željne informacije. Klasifikacija podatka odnosno razvrstavanje podataka prema kategorijama, te kategorije mogu biti prema vrsti objekata, visinama i drugim kriterijima. Registracija podatka je važna metoda obrade te služi za smještaj podatka u prostor dodavanjem istog koordinatnog sustava bilo to globalnog ili lokalnog više različitih skupina podataka. Nadalje metoda segmentacije omogućuje razdvajanje oblaka točaka u dijelove (ravninska segmentacija) i grupiranje točaka u klastere s obzirom na njihovu prostornu udaljenost (klastreska segmentacija). Metoda interpolacije je važna i služi za generiranje dodatnih točaka kako bi se podatci progustili, površinska rekonstrukcija omogućuje stvaranje površina iz oblaka točaka (linearna interpolacija, polinomska interpolacija, kriging, najbliži susjed). Normalizacija omogućuje manipulaciju podataka kako bi se uklonile moguće razlike u mjerilima ili jedinicama dok transformacija koristi matematičke metode nad podatcima

radi mogućnosti prilagodbe različitim kordinatnim sustavima te ju je moguće koristiti za ispravljanje deformacija nad podatcima.

3.2. Tehnologije i alati

Ako se obrada oblaka točaka vrši softverski korištenje tehnologija i alata kao što su to CloudCompare i Gis softveri omogućuju jednostavnost i brzinu obrade. CloudCompare je besplatno rješenje za obradu oblaka točaka. Prvobitno je zamišljen i dizajniran za uspoređivanje između dva 3D oblaka točaka te između točaka ili trokutaste mreže [1]. Neke od glavnih funkcionalnosti softvera su: učitavanje i pregled 3D oblaka točaka, filtriranje, registracija, vizualizacija, analiza te skriptiranje i automatizacija. Ove funkcionalnosti korisnicima olakšavaju vizualizaciju objekata i omogućuju efikasno rješavanje problema iz različitih područja vezanih uz obradu podataka [2]. Moguće je korištenje i GIS softvera bilo onih komercijalnih (ArcGis,) ili besplatno dostupnih npr. kao što je to Qgis. Korištenje QGis-a omogućuje stvaranje, uređivanje, vizualizaciju, analizu geoprostornih podataka. Definicija QGis-a na službenoj stranicu glasi: "QGIS je korisnički prijateljski otvoreni geografski informacijski sustav (GIS) licenciran pod GNU (General Public License). QGIS je službeni projekt Open Source Geospatial Foundation (OSGeo)" [3].

Osim korištenja softvera obrada podataka oblaka točaka može se obaviti i programski npr. korištenjem C++, Java, JavaScript, Python i drugih programski jezika u bilo kojem uređivaču koda. Obrada programskim jezicima npr. Python jezikom je efikasno rješenje koje nudi korištenje različitih alata i biblioteka kao što su to: Open 3D i Shapely. Učenje Pythona je snažna osnova za kasnije baratanje drugim jezicima jer sadrži sve njihove karakteristike. Izrazito je jednostavan za razumijevanje i čitanje, pogodan je i koristan za stvaranje aplikacija te je besplatan i dostupan svima čime tvori veliku zajednicu programera koji ga koriste [4].

Open 3D je biblioteka otvorenog koda koja podržava brzi razvoj softvera za rad s 3D podatcima. "Open3D je razvijen od nule s malim i pažljivo odabranim skupom ovisnosti. Može se postaviti na različitim platformama i kompajlirati iz izvornog koda uz minimalan trud" [5]. Neke od funkcionalnosti Open3D biblioteke su učitavanje i spremanje podataka, vizualizacija, filtriranje i automatizacija.

Shapely je Python paket s licencom BSD (Berkeley Software Distribution) koji služi za manipulaciju i analizu ravninskih geometrijskih objekata(točaka, linija, poligona) te je osnovan na GEOS biblioteci (Geometry Engine - Open Source) koja je bazirana na PostGIS-u (geografska

ekstenzija za PostgreSQL) i port JTS-u (Java Topology Suite) [6]. Prirodan je za korištenje Python programerima, nije zavisan o bazi podataka jer direktno radi u programu.

3.3. Formati

Formati su izrazito važni pri obradi podatak oblaka točaka omogućuju razmjenu, analizu i skladištenje podatka. Važniji formati iz područja geoinformatike su JSON(JavaScript Object Notation), GeoJSON, LAZ(LASzip) , LAS (LASer), PLY (Polygon File Format or Stanford Triangle Format).

3.3.1. LAZ (LASzip)

LAZ je komprimirani format za pohranu podataka laserskog skeniranja baziran na LAS formatu (LAS (LASer) format je binarni format za pohranu 3D točaka koje se prikupljaju pomoću LiDAR (Light Detection and Ranging) tehnologije) [7]. LAZ značajno smanjuje veličinu datoteke bez gubitka podatka LAS datoteke. Format koristi algoritam LASzip koji omogućuje brzi prijenos podatka, manje prostora za skladištenje, te pristup podacima i način korištenja ostaje isti kao i kod LAS formata [8]. Struktura LAS format definirana je javnim zaglavljem bloka koji sadrži općenite meta podatke o datoteci, VLRs(varijabilni daljinski zapis) nije obavezna značajka te datoteka može sadržavati nula ili više VLRs -ova koji sadrže dodatne meta podatke , glavni dio formata je zapis podatka o točkama koji sadrži zapise kordinata točke te dodatne informacije o intenzitetu i boji povratnog signala, EVLRs (prošireni varijabilni duljinski zapis) omogućuje proširenje i pruža podršku dodatnim značajkama [8]. Zaključno LAS format pruža standardni način pohrane dok LAZ omogućuje bezgubitnu kompresiju LAS datoteke.

3.3.2. Ply (Polygon File Format)

PLY (Polygon File Format ili Stanford Triangle Format) je format datoteke koji se koristi za predstavljanje 3D objekata. PLY datoteke mogu sadržavati informacije o vrhovima, rubovima i poligonima (najčešće trokutima) koje definiraju površinu 3D objekta te je široko rasprostranjen u računalnoj grafici. Ključne karakteristike ovog formata su struktura datoteke sačinjena od zaglavlja i tijela, tekstualnih i binarnih format, podatci vrhova (sadrži koordinate svakog vrha i ostale atribut), podatci o licima (sadrži definicije poligona gdje svako lice sadrži popis indeksa

vrhova koji ga čine). PLY format ima rasprostranjenu upotrebu u računalnoj grafici pri 3D modeliranju i prikazivanju podatka također u 3D skeniranju za pohranu i obradu podataka [9].

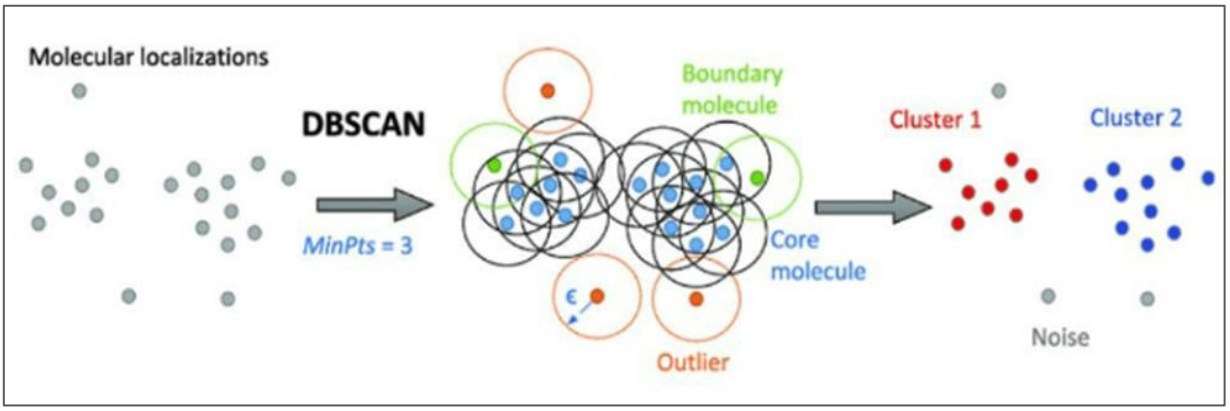
3.3.3. GeoJSON

GeoJSON format je široko korišten u različitim aplikacijama za geografske informacijske sustave (GIS) i web mapping aplikacijama. Popularne biblioteke i alati koji podržavaju GeoJSON uključuju: Leaflet, Mapbox, QGIS, PostGIS. Podržava različite tipove geometrija podataka (točke, linije, poligoni i njihove korelacije) [11]. Osnovne komponente ovog formata su značajaka (geografski objekt s pridruženim svojstvima), geometrija(definira oblik lokaciju objekta), zbirka značajki (omogućuje grupiranje više geometrijskih objekata zajedno i skupa s njihovim svojstvima). Izrazito je čitljiv i jednostavna za rad s geografskim podacima. GeoJSON je format za kodiranje različitih geografski referenciranih podataka koristeći JSON. JSON (JavaScript Object Notation) je format za razmjenu podataka. Tekstualni format koji je lagan za čitanje i pisanje, te je jednostavan za generiranje [12]. JSON je utemeljen na podskupu JavaScript jezika pogotovo što se tiče načina pisanja objekata i nizova podataka. Osnovne komponente formata su :objekti (zbirke parova ključ vrijednosti pri čemu su ključevi stringovi a vrijednosti mogu biti različit tipovi), polja (uređene zbirke vrijednosti, te vrijednosti unutra polja mogu biti bilo kojeg tipa), vrijednosti mogu biti tip: tekstualne vrijednosti, brojevi, logičke vrijednosti, nepostojeće vrijednosti, objekti. Format je jednostavan za korištenje, kompatibilan znači da nije ovisan o jeziku koji se koristi te je standardizirane strukture što olakšava generiranje.

3.4. DBSCAN

DBSCAN (Density based spatial clustering of application with noise) je algoritam za grupiranje podatka koji identificira podatke te kreira klastere s obzirom na njihovu gustoću odnosno bliskost (staviti literaturu) tako da vrijednosti s odstupanjima ostaju u području s manjom gustoćom podatka. DBSCAN algoritam se zasniva na dostupnosti temeljno na gustoći odnosno DDR (direct density-reachable) i DR (density reachable). Dva osnovna parametra DBSCAN algoritma su ε (epsilon) koji predstavlja minimalnu potrebnu udaljenost između dvije točke da bi se smatrale susjednima i MinPts (min_samples) minimalni broj točaka u ε okruženju kako bi točka bila jezgrena točka u klasteru.

Nakon pokretanja algoritam traži nasumičnim odabirom početni podatak odnosno podatak koji još nije posjećen (seed) te postavlja ε(epsilon) okruženje što uključuje sve okolne točke oko početne na udaljenosti ε(epsilon) proces formiranja klastera vidljiv je na slici Slika 1 [13]. Provjerom jezgrenosti definira hoće li točka iz ε(epsilon) okruženja postati jezgrena točka klastera, ako ε (epsilon) okruženje početne točke sadrži najmanje MinPts, točka je jezgrena te se formira klaster s točkama iz okruženja ali ako okruženje ne zadovoljava MinPts tada se početna točka označava kao odstupanje. Nije isključivo da ta točka neće postati dio drugog klastera ako se pronađe unutar okoline druge jezgrene točke.



Slika 1 Shema DBSCAN algoritma

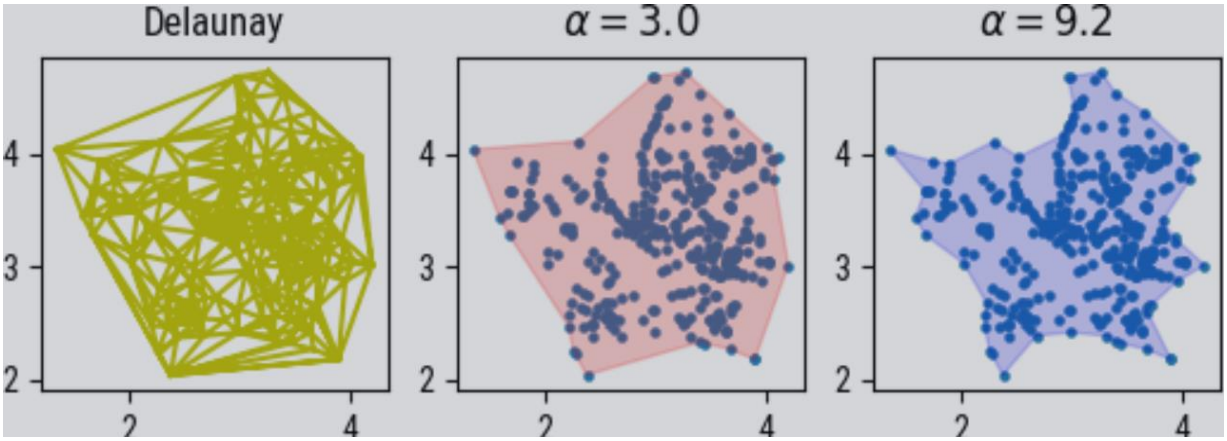
Kada je točka uključena u klaster njeno okruženje se uključuje pomoću koncepta povezanosti gustoće. Kako bi se definirala DC (density-connectedness) povezanost bazirana na gustoći potrebno je definirati DDR (eng. direct densitiy-reachable) i DR (density reachable). DDR odnosno direktna dostupnost gustoće se može prikaza pri čemu je: "Podatak q je direktno dostupan podatku p ako je p udaljen od q za najviše ε (epsilon) te ako postoji dovoljan broj podataka oko p

tako da može biti stvoren klaster oko p i q" [14]. Može se zaključiti da relacija direktne dostupnosti nije simetrična. Također DR odnosno dostupnost gustoće pri čemu je: "podatak q dostupan podatku p ako postoji niz podatka , p_1, p_2, p_3…,p_n pri čemu je p_1 jednak p a p_n jednak q" [14] te se može zaključiti da ova relacija također nije simetrična što znači da podatak p može biti dostupan podatku q ali podatak q možda neće imati dovoljan broj susjeda te se onda neće brojati kao dio klastera. Uz dostupnost definira se i povezanost baziranu na gustoći DC (density-connectedness). Dvije točke su međusobno povezane gustoćom ako postoji niz podatka odnosno točaka koje su međusobno direktno dostupne gustoće što kao rezultat daje formiranje klastera proizvoljnog oblika. Opisani koraci se ponavljaju dok sve točke nisu posjećene, na posljetku algoritam vraća skup klastera i točke odstupanja.

3.5. Alpha Shape

"Najnaprednija metoda za prilagodbu stupnja konkavnosti kako bi se postigao najbolji obris je metoda alpha shapeova" [15]. Alpha shape metoda koristi Delaunay triangulaciju da bi filtrirala konveksnu ljusku te postigla točniji oblik koji najbolje odgovara stvarnom rasporedu točaka. Konveksna ljuska dvodimenzialno u prikazu sadrži trokute dok trodimenzialno je sačinjena od tetraedara. Svaki trokut ili tetraedar sadrži svoje granice s duljinom, cilj alpha shape-ova je ukloniti neke granice koje formiraju prazne regije ljuske.

Generiranje alpha shape-a započinje stvaranjem Delaunay triangulacije odabranog skupa točaka, rezultat triangulacije je konveksna ljuska koja obuhvaća sve točke skupa. Nakon formiranja konveksne ljuske (najmanjeg konveksnog oblika kojim se obuhvaćaju sve točke skupa) provodi se njeno filtriranje čime se uklanjaju rubovi kako bi se uklonile prazne regije unutar ljuske. Parametar α (alpha) diktira koje rubove trokuta ili tetraedra je potrebno ukloniti a koje ne [16]. Na slici (Slika 2) [17] je prikazana razlika s obzirom na mijenjanje parametra α (alpha). Ova metoda je posebno korisna kod generiranja objekata s nepravilnom površinom jer omogućuje stvaranje fleksibilnog oblika koji se može poistovjetiti s stvarnim volumenom i oblikom skupa podataka.

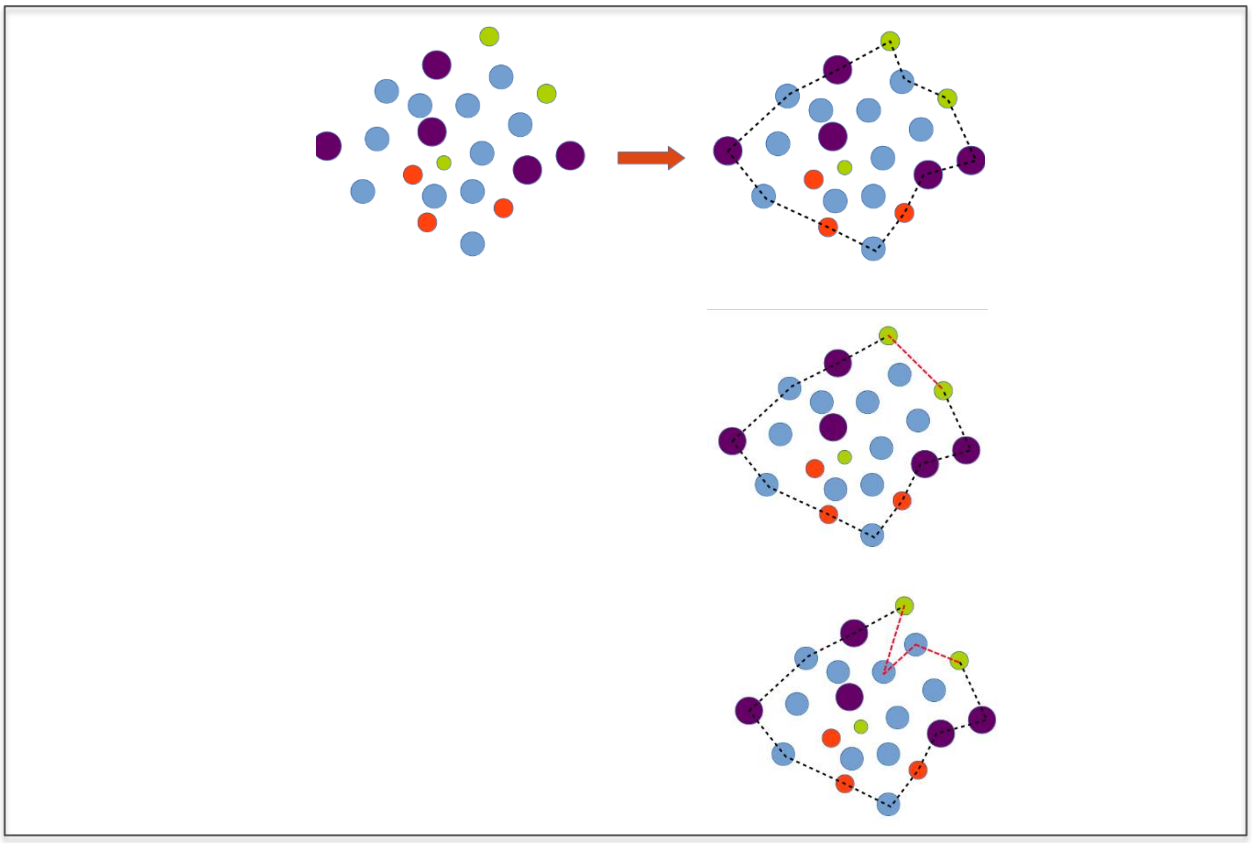


Slika 2 Ovisnost oblika o parametru alpha

3.6. Konveksna ljuska

"Konveksna ljuska skupa točaka definira se kao najmanji konveksni skup koji sadrži sve točke" [18]. Primjena metode konveksne ljuske je izrazito jednostavno i efikasno rješenje pri generiranju poligona. Zasniva se na matematičkim svojstvima konveksnosti i minimalnosti (najmanji konveksni skup točaka koji sadrži sve točke). Prednost konveksne ljuske je lakoća implementacije

radi velikog broja već postojećih algoritama za izračunavanje konveksne ljuske [18]. Ovisno o definiranju različitih kriterija moguće je postići i različite rezultate konveksne ljuske kako i prikazuje Slika 3 [19]. Svaka generirana konveksna ljuska na slici je formirana ispravno te je njen izgled ovisan isključivo o odabiru parametara.

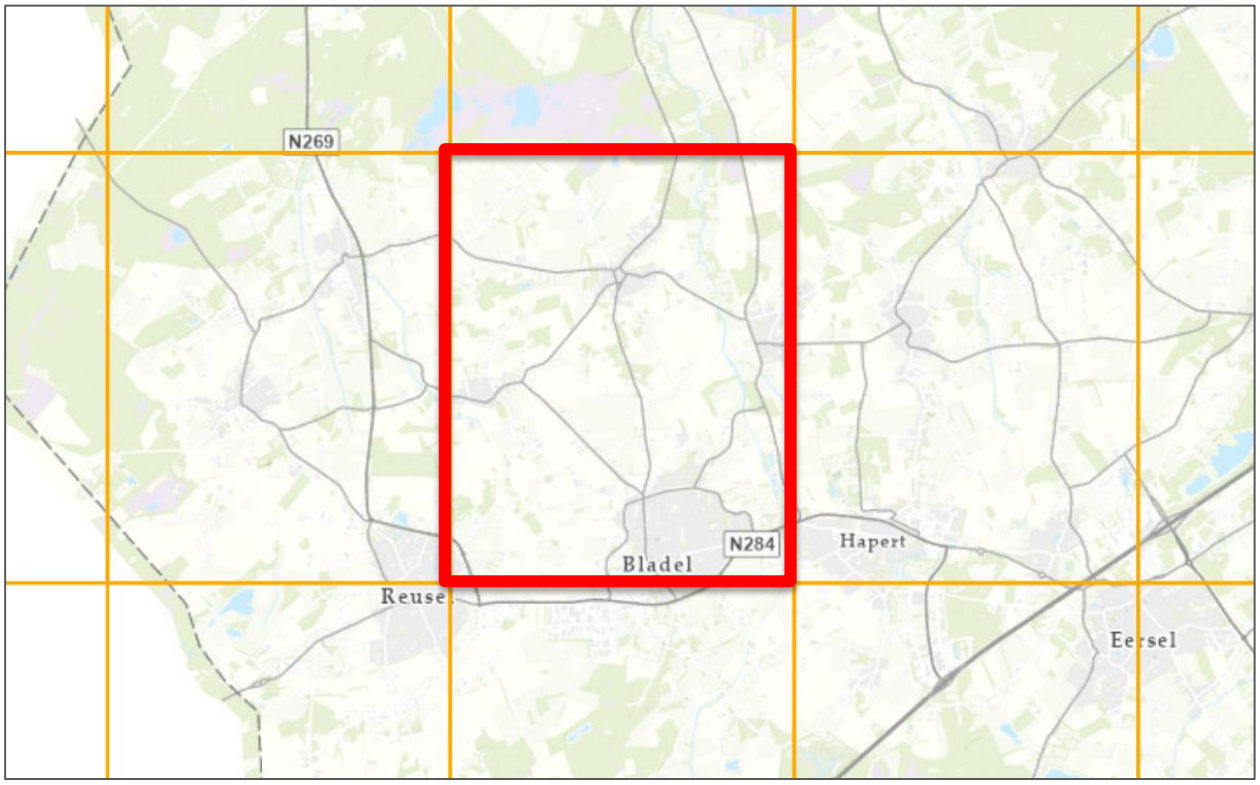


Slika 3 Shema formacija konveksne ljuske

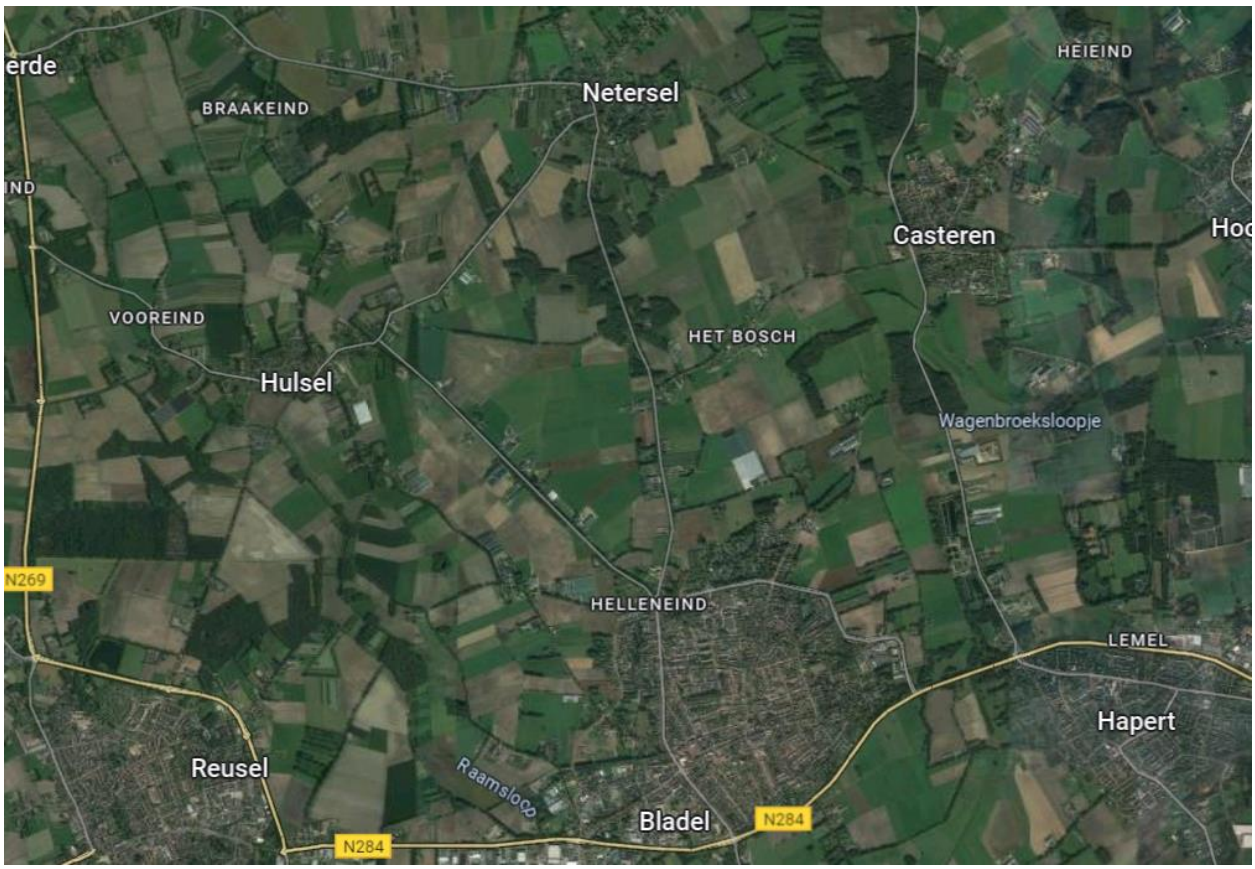
4. Praktični dio

4.1. Priprema podataka

Podatci korišteni za izradu programa su otvoreni javno dostupni klasificirani podatci LIDAR snimanja područja Nizozemske. Za potrebe generiranja tlocrta zgrada neophodno je razdvojiti podatke zgrada od podataka ostalih objekata koji nisu materijal ovog programskog zadatka. Podatke je moguće preuzeti na stranici "AHN4 - Download kaartbladen- ArcGIS Hub" (Slika 4) [20] koji se odnose se na aktualne visinske podatke Nizozemske "Actueel Hoogtebestand Nederland" verzije 4 [20]. Jasniji prikaz područja prikazan je na slici korištenjem Google karte (Slika 5)

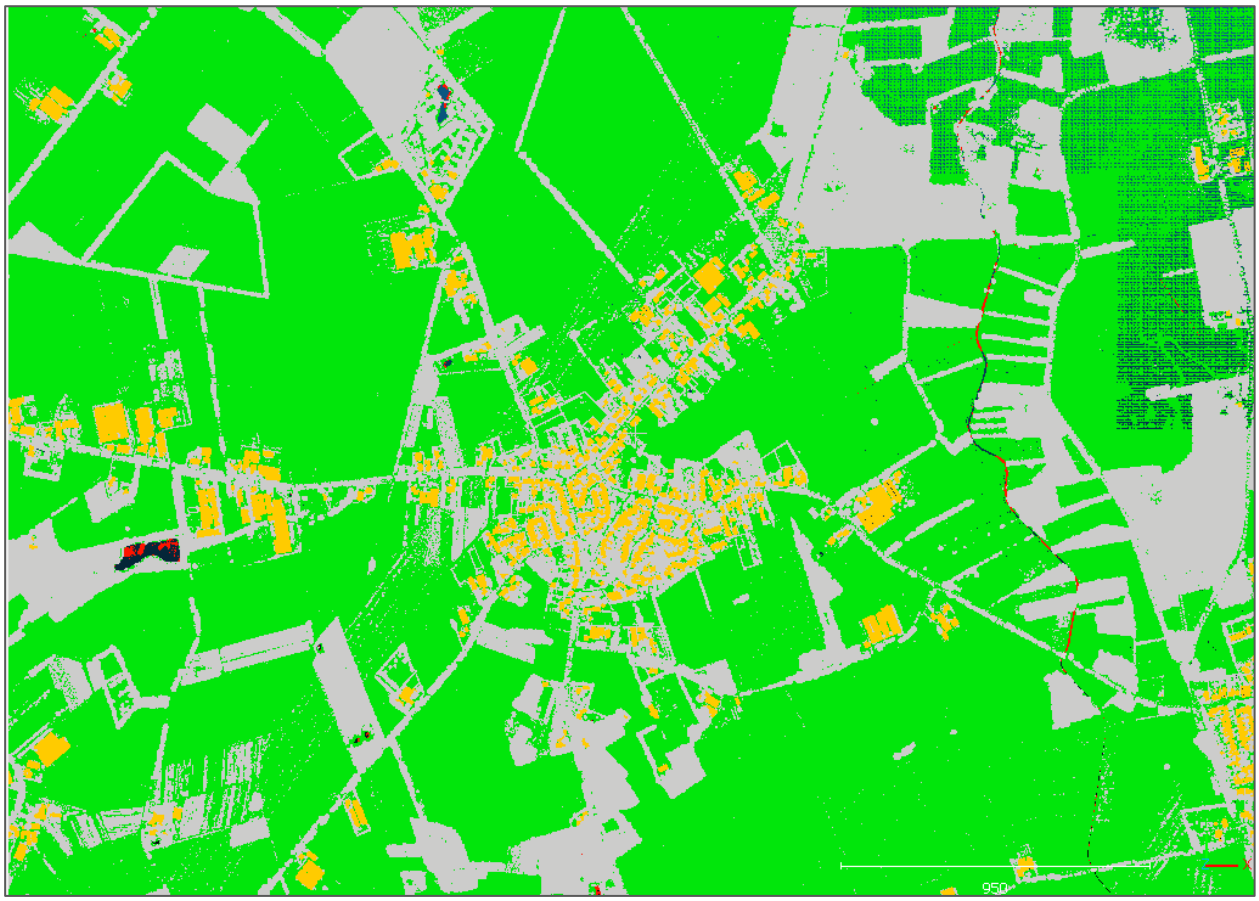


Slika 4 Prikaz područja s AHN4 - Download kaartbladen- ArcGIS Hub stranice

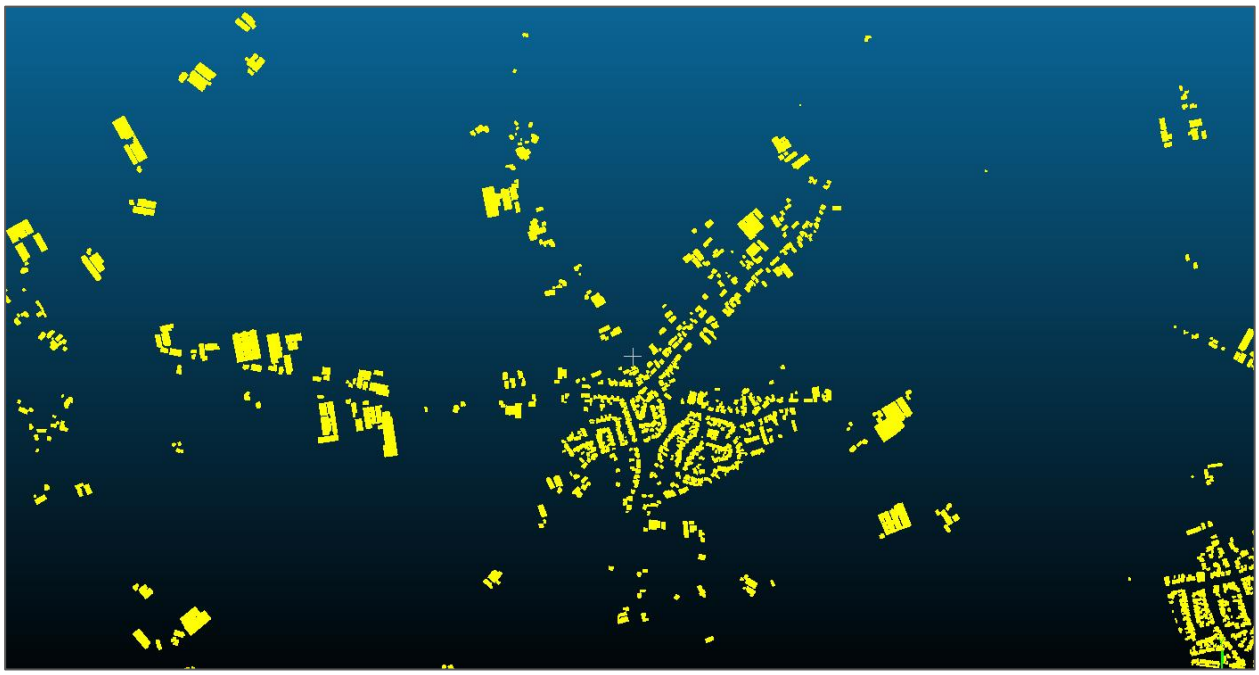


Slika 5 Prikaz područja na Google karte

Obrada podataka je provedena pomoću softvera Cloud Compare, gdje su podaci iz lista 51CZ1 karte razdvojeni na podatke klase 6 (zgrade) koristeći alat "Filter by Value". Taj alat omogućuje selektivno izdvajanje podataka određene klase, u ovom slučaju klase koja predstavlja zgrade. Kako bi se pojednostavila manipulacija podacima i smanjio broj znamenki tijekom korištenja alata Open3D, podaci su pomaknuti za -100000 po x osi i -400000 po y osi. Ova translacija koordinata omogućuje rad s manjim brojevima, što poboljšava preciznost i učinkovitost prilikom daljnje obrade i analize podataka. Nakon razdvajanja dobivena su dva skupa podataka. U prvom skupu nalaze se podaci klase 6, odnosno podaci koji predstavljaju zgrade, što je jasno vidljivo na priloženoj slici (Slika 7). Drugi skup podataka sadrži podatke svih ostalih klasa koji nisu relevantni za trenutnu obradu podataka zgrada (Slika 6). Za spremanje obrađenih podataka korišten je format .ply (Polygon File Format), koji je izuzetno prikladan za rad s Open3D alatom. Format .ply omogućuje pohranu trodimenzionalnih podataka na način koji je optimiziran za daljnju vizualizaciju i analizu u Open3D. Korištenje Cloud Compare softvera za početnu obradu i filtriranje podataka osigurava da su podaci pripremljeni na način koji omogućuje preciznu i učinkovitu daljnju analizu.



Slika 6 Podaci dijela lista karte 51CZ1 prikazanih u Cloud Compare-u



Slika 7 Izdvojeni podaci zgrada s dijela lista karte 51CZ1 prikazanih u Cloud Compare-u

4.2. Opis Python koda

Ovaj kod prvo koristi Open 3D biblioteku za učitavanje oblaka točaka iz PLY datoteke te ih pretvara u NumPy niz što osigurava efikasniju upotrebu DBSCAN algoritma. Nakon pretvorbe primjenjuje se DBSCAN algoritam za klasterizaciju. Potom se svaki pojedini klaster generira u poligon koristeći Alpha Shape algoritam, a ako Alpha Shape ne uspije generirati poligon za pojedini klaster onda se na taj klaster primjenjuje konveksna ljuska te se podatci odvojeno spremaju u GeoJSON datoteku.

4.3. Učitavanje korištenih biblioteka i modula

Slika 8 prikazuje sve korištene module i biblioteka za izradu programskog zadatka a to su: Open 3D biblioteka korištena za rad s podacima oblaka točaka, DBSCAN korišteni algoritam za klasteriranje iz sklearn biblioteke, numpy za rad s numeričkim podatcima, Alpha Shape korišteni algoritam za kreiranje Alpha Shape poligona, shapely biblioteka omogućuje korištenje geometrijskih operacija na poligonima te os i json korišteni za zapisivanje podataka i rad s datotekama.

```
import open3d as o3d
from sklearn.cluster import DBSCAN
import numpy as np
import alphashape
from shapely.geometry import Polygon
from shapely.ops import unary_union
from shapely.geometry import MultiPolygon
from shapely import affinity
import os
import json
from shapely.geometry import mapping
```

Slika 8 Popis biblioteka i modula

4.4. Učitavanje oblaka točaka i pretvprba u NumPy niz

Spremljeni podaci formata .ply su učitani korištenjem funkcije:
"o3d.io.read_point_cloud("C_25DN2_shft_bldgs - Cloud.ply"(Slika 9)
Ova funkcija koristi Open 3D biblioteku za učitavanje "point cloud" tipa podataka iz C-_51AZ1.ply datoteke. Nakon učitavanja podataka dio koda prikazan na slici (Slika 10) konvertira "point cloud" podatke u NupPy niz. NumPy niz je korišten kako bi kasnije o obradi efikasnije i lakše mogao biti primijenjen DBSCAN algoritam koji zahtjeva ovaj oblik podataka. Pretvorbom u NumPy niz dobije se dvodimenzionalni niz u kojemu redci predstavljaju točke niza a stupci redom x, y i z koordinate.

```
# Učitavanje podataka
cloud = o3d.io.read_point_cloud("C_51AZ1_ZGRADE _DIO- Cloud.ply")
```

Slika 9 Učitavanje podataka

```
# Pretvorba u NumPy niz
points = np.asarray(cloud.points)
```

Slika 10 Funkcija za pretvorbu u NumPy niz

4.5. Klasteriranje s DBSCAN algoritmom

4.5.1. Procjena eps vrijednosti

Za korištenje DBSCAN algoritma potrebno je odrediti parametar eps vrijednost (minimalna potrebna vrijednost između dvije točke). Ovaj parametar svojevoljnom procjenom nakon nekoliko iteracija s promjenama vrijednosti je postavljen na 0.8 (Slika 11) (" fixed_eps = 0.8") jer daje naj vjerodostojni izgled pojednostavljenog poligona u usporedbi sa stvarnim stanjem te zadovoljava razinu detalja tlocrta i zahtjeva razumnu količinu vremena za generiranje. Osim parametra epsilon postavljenog fiksno određen je i parametar "MinPts" odnosno minimalni broj točaka u epsilon okruženju koji iznosi 10 (Slika 12).

```
# Postavljanje fiksne eps vrijednosti
fixed_eps = 0.8  # po potrebi promjenjiva vrijednost
```

Slika 11 Postavljanje eps(epsilon) vrijednosti

```
# DBSCAN parametri
eps = fixed_eps  # Koristi fiksnu eps vrijednost
min_samples = 10  # Minimalan broj točaka za formiranje klastera

# Primjena DBSCAN klasteriranja
db = DBSCAN(eps=eps, min_samples=min_samples).fit(points)
labels = db.labels_
```

Slika 12 DBSCAN parametri i primjena klasteriranja

4.5.2. Izdvajanje segmentiranih klastera

U dijelu primjene naredbom DBSCAN ("`eps=eps, min_samples=min_samples`") pripremaju se početne vrijednosti algoritma. dok "`.fit(points)`" definira da se algoritam primjenjuje na vrstu podatka točke ("points"). "`labels = db.labels_`" izvlači oznake klastera pri čemu svaka točka dobiva oznaku kojem klasteru pripada. Slika 13 prikazuje izdvajanje segmentiranih klastera pri čemu se stvaraju oznaka za sve jedinstvene klastere dok šumovi odbijaju oznaku."'-1", "`len(set(labels)) - 1`" računa ukupan broj klastera ali isključuje šumove. "`points[labels == i]`" filtrira točke koje pripadaju klasteru i, "`for i in range(len(set(labels)) - 1) if i != -1`" prikuplja točke iz svakog klastera kroz iteracije svih klastera osim šumova.

```
# Izdvajanje segmentiranih klastera
clusters = [points[labels == i] for i in range(len(set(labels)) - 1)
if i != -1]
```

Slika 13 Izdavanje segmentiranih klastera

4.6. Generiranje poligona aloritmom alpha shape

Funkcija na slici (Slika 14) prikazuje generiranje alpha shape-a pozivanjem na funkciju." `alpha_shape_polygon`" Pri čemu se generira poligon za skup točaka i alpha parametar." `alpha_shape = alphashape.alphashape(points, alpha)`" služi za izračunavanje alfa oblika za točke. Te se provjerava je li rezultat multipoligon i ako je onda funkcija spoji sve u jedan poligon. Također se testira je li rezultat poligon ako on to je i generirao se uspješno vratit će ga naredba " `return alpha_shape`" no ako iz podataka nije moguće generirati poligon podiže se iznimka "`raise ValueError` " pri čemu se one uzimaju i vraćaju pod "`none`" s "`return None`"

```
# Funkcija za generiranje poligona iz alpha shape-a
def alpha_shape_polygon(points, alpha):
    try:
        alpha_shape = alphashape.alphashape(points, alpha)
        if isinstance(alpha_shape, MultiPolygon):
            alpha_shape = unary_union(alpha_shape)
        if isinstance(alpha_shape, Polygon):
            return alpha_shape
        else:
            raise ValueError('Unable to generate polygon')
    except Exception as e:
        return None
```

Slika 14 Generiranje poligona iz Alpha Shape-a

Slika 15 prikazuje dio koda koji određuje parametar za definiranje minimalne površine poligona (25). Poligoni s manjom vrijednosti površine od zadane vrijednosti biti će zanemareni, te će se ispisivanje podataka spremiti pod output files poddirektoriji, ako direktoriji ne postoji kreirat će ga funkcija "`os.makedirs`".

```
# minimalna površina poligona
min_polygon_area = 25  # po potrebi promjenjiva vrijednost

# izlazni direktoriji
output_dir = "output_files"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

Slika 15 Definiranje minimalne površine i kreiranje izlaznog direktorija

4.7. Zapis rezultata

Slika 16 prikazuje funkcije za definiranje putanja GeoJson datoteka koja će sadržavati alfa oblik, konveksne ljuske i neuspješne klastere.

```
# Putanje za GeoJSON izlazne datoteke
geojson_alpha_path = os.path.join(output_dir,
"alpha_shapes.geojson")
geojson_convex_path = os.path.join(output_dir,
"convex_hulls.geojson")
geojson_failed_path = os.path.join(output_dir,
"failed_clusters.geojson")
```

Slika 16 Kreiranje putanja za izlazne datoteke

```
# GeoJSON priprema strukture značajki
alpha_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
convex_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
failed_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
```

Slika 17 Priprema strukture

Slika 17 predstavlja varijablu za pripremu strukture GeoJSON podataka. Odnosno u ovom slučaju tip objekata je "`FeatureCollection`" znači da će objekt sadržavati kolekciju značajki geometrije (features). Te se kreira prazan popis koji se kasnije popunjava poligonima generiranih iz klastera za strukture alfa oblika, konveksne ljuske i neuspješne klastere.

Prikaz koda na slici (Slika 18) sadrži iteraciju kroz svaki klaster pri čemu se postavlja i oznaka valjanosti alfa oblika " `valid_alpha = False`". Iteracija se vrši i kroz različite vrijednosti alfa parametara nakon čega se generira alfa oblik trenutnog klastere te provjerava njegova valjanost s obzirom na postavljeni parametar minimalne površine. Ako se takav alfa oblik definira valjanim dobije oznaku "`True`" i petlja se prekida.

```
# Iteracija kroz klastere i generiranje poligona
for i, cluster in enumerate(clusters):
    valid_alpha = False
    for alpha_value in [2.0, 1.5, 1.0, 0.5]:
        alpha_polygon = alpha_shape_polygon(cluster[:, :2],
alpha=alpha_value)
        if alpha_polygon and alpha_polygon.area >= min_polygon_area:
            valid_alpha = True
            break

    if valid_alpha:
        # Kreiranje GeoJSON značajke  za alpha oblik poligon
        feature = {
            "type": "Feature",
            "geometry": mapping(alpha_polygon),
            "properties": {
                "cluster_id": i + 1,
                "original_area": alpha_polygon.area
            }
        }
          # Dodavanje feature-a u alpha shape Feature Collection
        alpha_feature_collection["features"].append(feature)
```

Slika 18 Iteracija kroz klastere i generiranje poligona (Alpha Shape)

U slučaju generiranja valjanog alfa oblika kreira se GeoJSON značajka za alfa oblik poligon, te se značajka dodaje u varijablu "`alpha_feature_collection `". Slika 19 prikazuje da u slučaju ako alfa oblik nije valjan generira se konveksna ljuska "`convex_hull_polygon = MultiPoint(cluster[:, :2]).convex_hull`", također kreira se GeoJSON značajaka za konveksnu ljusku, te se značajaka dodaje u varijablu " `convex_feature_collection`" U slučaju da niti alpha shaepe niti konveksna ljuska nisu uspješno generirali poligon pod "`except Exception as e`" točke se dodaju u varijablu " `failed Feature Collection` " geometrije oblika točka ("Point") (Slika 20).

```
                else:
                    try:
                # Primjena konveksne ljuske ako alfa oblik nije valjan
                    convex_hull_polygon = MultiPoint(cluster[:,
                    :2]).convex_hull

                # Kreiranje GeoJSON feature za konveksnu ljusku poligona
                    feature = {
                    "type": "Feature",
                "geometry": mapping(convex_hull_polygon),
                    "properties": {
                    "cluster_id": i + 1,
                "original_area": convex_hull_polygon.area
                    }
                    }
                # Dodavanje feature-a u convex hull Feature Collection
                convex_feature_collection["features"].append(feature)
```

Slika 19 Primjena konveksne ljuske

```
        except Exception as e:
            # Ako ni alfa oblik i konveksna ljuska ne uspiju
generirati poligon dodavaju se tačke u failed clusters
            for point in cluster:
                feature = {
                    "type": "Feature",
                    "geometry": {
                        "type": "Point",
                        "coordinates": point[:2].tolist()
                    },
                    "properties": {
                        "cluster_id": i + 1,
                        "error": str(e)
                    }
                }
                # Dodavanje feature-a u failed Feature Collection
                failed_feature_collection["features"].append(
```

Slika 20 Dodavanje točki u neuspješne klastere

Slika 21 prikazuje zapisivanje Alpha Shape-a, konveksne ljuske i neuspješnih klastera. " `with open(geojson_alpha_path, "w"` “ otvara se datoteka za zapisivanje podataka alfa oblika, dok" `json.dump(alpha_feature_collection,geojson_alpha_file,indent=2)` "zapisuje značajke u GeoJSON datoteku varijablom "`alpha_feature_collection`", te se definira uvlaka radi bolje čitljivosti datoteke. Na isti način se zapisuju i kolekcije značajki konveksne ljuske i neuspješnih klastera.

```
# Zapisivanje alpha shape Feature Collection u GeoJSON datoteku
with open(geojson_alpha_path, "w") as geojson_alpha_file:
    json.dump(alpha_feature_collection, geojson_alpha_file,
indent=2)

# Zapisivanje convex hull Feature Collection u GeoJSON datoteku
with open(geojson_convex_path, "w") as geojson_convex_file:
    json.dump(convex_feature_collection, geojson_convex_file,
indent=2)

# Zapisivanje failed clusters Feature Collection u GeoJSON datoteku
with open(geojson_failed_path, "w") as geojson_failed_file:
    json.dump(failed_feature_collection, geojson_failed_file,
indent=2)
```

Slika 21 Zapisivanje rezultata

4.8. Prikazivanje rezultata

Finalni generirani poligoni prikazani su korištenjem QGIS softvera. (Slika 22). Podloga za prikaz postavljena je korištenjem WMS-a (Web Map Service) katastarskog plana Nizozemske sa službene stranice Kadaster [22]. Crvenim rubom bez ispune su vizualizirani poligoni dobiveni algoritmom Alpha Shape, a sivo su zgrade iz navedene WMS usluge.

Slika 22 Vizualizacija generiranih poligona (za dio lista 51CZ1)

5. Analiza rezultata

5.1. Tehnički podaci

Algoritam je uspio generirati poligone podataka lista karte te je uz postavljene parametre DBSCAN algoritma(epsilon:0.8, minimalni broj točaka:10) i Alpha Shape-a (minimalna površina poligona:25) trajanje izvršavanja procesa iznosilo je 38 minuta za odabrano području lista karte (51CZ1). Učitano je 5916726 točaka iz oblaka točaka te grupirano 2957 klastera nakon provedbe DBSCAN algoritma. Računalo korišteno pri obradi ima sljedeće specifikacije: AMD Ryzen 5 3600 6-jezgreni procesor, 96 GB RAM-a, 64-bitni operativni sustav MS Windows 11.

5.2. Praktična primjena rezultata

5.2.1. Primjeri upotrebe

Korištenje generiranih tlocrta iz podatka oblaka točaka može imati primjenu u inženjerskim područjima znanosti. Ovakav krajnji rezultat može biti koristan korisnicima GIS-a pri čemu zbog svoje strukture poligoni su pregledniji i omogućuju lakše analiziranje i vizualizaciju podatka oblaka točaka, što je isključivo korisno planerima i arhitektima u procesu upravljanja okolišem čiji pristup zahtjeva jednostavnu vizualizaciju. Brži pristup ovim podatcima i njihovo često ažuriranje bi bilo pogodno i pri praćenju promjena u urbanim i ruralnim sredinama (praćenje novoizgrađenih prometnica, objekata, promjena u uporabi vlasništva). Također, tu je i uporaba tlocrta za generiranje 3D pojednostavljenih modela za korištenje u analizama koje nisu ovisne o vizualnoj detaljnosti podatka (npr. analiza buke, analize osvjetljenja, analize optimizacije prostora).

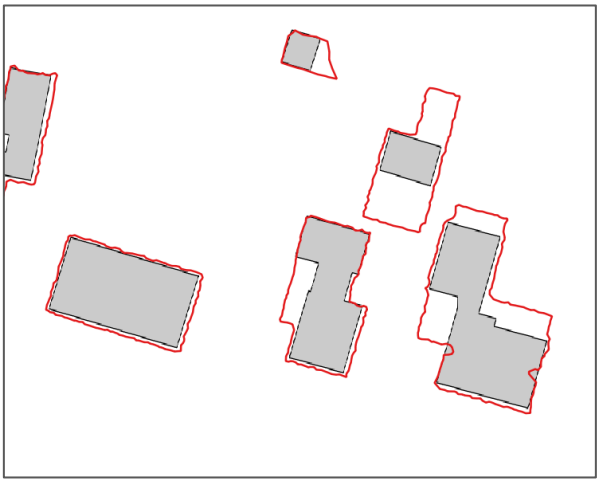
5.2.2. Ograničenja i izazovi

Jedan od izazova je velika količina podatka koja znatno usporava proces generiranja, nakon generalizacije podataka očekivano je provoditi dodatnu obradu u smislu čišćenja i filtriranja poligona za koje se nisu ispravno generirali poligoni.

Ograničenje može proizići iz odabira parametra "min_sample", zbog čega je važno razumno odabrati vrijednost parametra jer od nje zavisi i kvaliteta rezultata klasterirarnja, također i

parametar epsilon može znatno promijeniti krajnji rezultat generiranja te je jednostavno neizbježno testirati različite vrijednosti ovog parametra sve dok se ne postigne zadovoljavajuća kvaliteta

Detaljnijim pregledom se može zaključiti da su službeni podaci (WMS) značajnije generalizirani pa neki dijelovi zgrada koji u stvarnosti postoje nisu prisutni u tom skupu podataka, kao i prikazuje Slika 23.



Slika 23 Usporedba WMS podataka i generiranih poligona

Prema dizajnu algoritma, poligoni se neće generirati za klastere čija je površina manja od 25 m2 (Slika 24).

Slika 24 Vizualizacija generiranih poligona i ne generiranih poligona

Prema implementiranom procesu generiranja, kada se za neki klaster ne uspije generirati poligon Alpha Shape onda se generira konveksna ljuska. Konveksne ljuske su vizualizirane zelenim ispunjenim poligonima.(Slika 25)

Slika 25 Generiranje konveksnih ljuski

Primjer na slici (Slika 26) je slučaj kada algoritma DBSCAN nije uspio generirati jedan veliki klaster već se stvorilo puno manjih koji su onda generirani kao konveksne ljuske.

Slika 26 Slučaj kada DBSCAN nije uspio prikladno klasterirati točke

Potencijalna su buduća istraživanja koja se odnose na kvalitetu generiranih podataka, posebno poligona dobivenih različitim metodama. Ključni fokus je usporediti različite metode generiranja poligona kako bi se utvrdila najbolja točnost moguća s obzirom na podatke dobivene LIDAR snimanjem. Također bi bilo korisno napraviti usporedbu odstupanja generiranih podataka s podacima prikupljenim drugim metodama mjerenja, s ciljem procjene je li korištenje otvorenih podataka prihvatljiva zamjena za druge metode mjerenja u području GIS-a i geoinformatike.

6. Zaključak

U radu predstavljen je sveobuhvatan proces generiranja tlocrta zgrada korištenjem otvorenih podataka laserskog skeniranja Nizozemske. Pristup generiranja uključivao je korištenje DBSCAN algoritma za klasifikaciju podataka oblaka točaka dobivenog zračnim laserskim skeniranjem, te generiranje Alpha Shape oblika korištenog za konstrukciju tlocrta zgrada. Analizom rezultata pokazuje se da generirani tlocrti u značajnom broju slučajeva odgovaraju tlocrtima zgrada iz formalne evidencije što ih čini upotrebljivim u daljnjem korištenju. Iako podatci nisu egzaktni njihova preciznost je dovoljna za primjene koje uključuju urbano planiranje, GIS analize i arhitekturu. Ovi rezultati potvrđuju da je moguće koristiti LiDAR podatke u kombinaciji s algoritmima za generiranje prostornih informacija. Može se zaključiti da bi ova metoda generiranja mogla imati mjesto u budućim testiranjima i istraživanjima integracijom drugih tehnika i alata s ciljem dobivanja preciznijih podataka veće točnosti. Također se može pretpostaviti i da bi daljnja unaprijeđena oblika promjenom parametara mogla otvoriti nove mogućnosti korištenja podatka. Obrađena metoda generiranja podataka demonstrira znatan potencijal korištenja podataka oblaka točaka.

7. Literatura

[1] https://www.cloudcompare.org/presentation.html, Dostupno: 19.06.2024.

[2] Gura, Dmitry & Karamysheva, Ekaterina & Pshidatok, Saida, Using CloudCompare software editing tools for processing a three-dimensional point cloud of an urban development site. (2024).

[3] https://www.qgis.org/en/site/about/index.html, Dostupno: 19.06.2024.

[4] Simon Yuill and Harry Halpin, Python,2006.

[5] https://www.open3d.org/ , Dostupno: 19.06.2024.

[6] https://pypi.org/project/shapely/ , Dostupno: 19.06.2024.

[7] ISENBURG, Martin. LASzip: lossless compression of LiDAR data. Photogrammetric

[8] Open Geospatial Consortium, LAS Specification 1.4 OGC Community Standard, 2018.

engineering and remote sensing, 2013, 79.2: 209-217.

[9] https://gis.stackexchange.com/questions/143821/how-to-find-the-concave-hull-for-a-cloud-of-points-in-3d-space, Dostupno: 19.06.2024.

[10] https://www.loc.gov/preservation/digital/formats/fdd/fdd000501.shtm,Dostupno:19.06.2024

[11] Dušan Petković, SQL/JSON Standard: Properties and Deficiencies, FACHBEITRAG, Published online: 24.10. 2017.

[12] 2.Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T., et al.: The geojson format. In: Internet Engineering Task Force (IETF) (2016)

[13] https://www.nomidl.com/machine-learning/dbscan-algorithm-for-machine-learning/,

[14] A. Habijanić, "Grupiranje podataka", Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku, Osijek, 2020. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:126:501750

Dostupno: 19.06.2024

[15] D. A. Stepanova, V. I. Antonov, V. A. Naumov, International Ural Conference on Electrical Power Engineering (UralCon) Compression of the Training Sample of the Smart Protection Device without Compromising its Information Capacity , 2021.

[16] Alpha shapes: determining 3D shape complexity across morphologically diverse structures, James D. Gardiner1 , Julia Behnsen2 and Charlotte A. Brassey3*, (2018)

[17] https://aliquote.org/post/alpha_shape/, dostupno: 25.06.2024

[18] A. Bienkowski, D. Sidoti and K. R. Pattipati, "Interference Identification for Time-Varying Polyhedra," in *IEEE Access*, vol. 9, pp. 138647-138657, 2021, doi: 10.1109/ACCESS.2021

[19] https://gis.stackexchange.com/questions/143821/how-to-find-the-concave-hull-for-a-

cloud-of-points-in-3d-space Dostupn:19.05.2024.

[20]https://www.arcgis.com/apps/mapviewer/index.html?webmap=f94e188409724e67bb6069e33bfc0cd0 Dostupn:18.05.2024.

[21] https://www.codeblog.rs/clanci/json Dostupno: 20.06.2024.

[22] https://www.kadaster.nl/, Dostupno: 20.06.2024.

Popis slika

Dodaci

Dodatak 1 – cjeloviti kod

```
import open3d as o3d
from sklearn.cluster import DBSCAN
import numpy as np
import alphashape
from shapely.geometry import Polygon, mapping, MultiPolygon, MultiPoint
from shapely.ops import unary_union
import os
import json

# Učitavanje podataka oblaka točaka
cloud = o3d.io.read_point_cloud("C_51CZ1.ply")

# Pretvorba u NumPy niz
points = np.asarray(cloud.points)

# Postavljanje fiksne eps vrijednosti
fixed_eps = 0.8  # po potrebi promjenjiva vrijednost

# DBSCAN parametri
eps = fixed_eps  # Koristi fiksnu eps vrijednost
min_samples = 10  # Minimalan broj točaka za formiranje klastera

# Primjena DBSCAN klasterizacije
db = DBSCAN(eps=eps, min_samples=min_samples).fit(points)
labels = db.labels_

# Ekstrakcija segmentiranih klastera
clusters = [points[labels == i] for i in range(len(set(labels)) - 1) if i != -1]

# Funkcija za generiranje poligona iz alpha oblika
def alpha_shape_polygon(points, alpha):
    try:
        alpha_shape = alphashape.alphashape(points, alpha)
        if isinstance(alpha_shape, MultiPolygon):
            alpha_shape = unary_union(alpha_shape)
        if isinstance(alpha_shape, Polygon):
            return alpha_shape
        else:
            raise ValueError('Unable to generate polygon')
    except Exception as e:
        return None

# Minimalna površina poligona
min_polygon_area = 25 # po potrebi promjenjiva vrijednost
```

```
# Izlazni direktoriji za rezultate
output_dir = "output_files"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Putanje za GeoJSON izlazne datoteke
geojson_alpha_path = os.path.join(output_dir, "alpha_shapes.geojson")
geojson_convex_path = os.path.join(output_dir, "convex_hulls.geojson")
geojson_failed_path = os.path.join(output_dir, "failed_clusters.geojson")
# GeoJSON Feature Collection strukture
alpha_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
convex_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
failed_feature_collection = {
    "type": "FeatureCollection",
    "features": []
}
# Iteracija kroz klastere i generiranje  poligona
for i, cluster in enumerate(clusters):
    valid_alpha = False
    for alpha_value in [2.0, 1.5, 1.0, 0.5]:
        alpha_polygon = alpha_shape_polygon(cluster[:, :2], alpha=alpha_value)
        if alpha_polygon and alpha_polygon.area >= min_polygon_area:
            valid_alpha = True
            break

    if valid_alpha:
        # Kreiranje GeoJSON feature za alpha oblik poligon
        feature = {
            "type": "Feature",
            "geometry": mapping(alpha_polygon),
            "properties": {
                "cluster_id": i + 1,
                "original_area": alpha_polygon.area
            }
        }
        # Dodavanje feature-a u alpha shape Feature Collection
        alpha_feature_collection["features"].append(feature)
    else:
        try:
            # Primjena konveksne ljuske ako alfa oblik nije valjan
            convex_hull_polygon = MultiPoint(cluster[:, :2]).convex_hull
```

```
            # Kreiranje GeoJSON feature za konveksnu ljusku poligon
            feature = {
                "type": "Feature",
                "geometry": mapping(convex_hull_polygon),
                "properties": {
                    "cluster_id": i + 1,
                    "original_area": convex_hull_polygon.area
                }
            }
            # Dodavanje feature-a u convex hull Feature Collection
            convex_feature_collection["features"].append(feature)
        except Exception as e:
            # Ako i alfa oblik i konveksna ljuska ne uspiju, dodaj točke u failed clusters
            for point in cluster:
                feature = {
                    "type": "Feature",
                    "geometry": {
                        "type": "Point",
                        "coordinates": point[:2].tolist()
                    },
                    "properties": {
                        "cluster_id": i + 1,
                        "error": str(e)
                    }
                }
                # Dodavanje feature-a u failed Feature Collection
                failed_feature_collection["features"].append(feature)

# Zapisivanje alpha shape Feature Collection u GeoJSON datoteku
with open(geojson_alpha_path, "w") as geojson_alpha_file:
    json.dump(alpha_feature_collection, geojson_alpha_file, indent=2)

# Zapisivanje convex hull Feature Collection u GeoJSON datoteku
with open(geojson_convex_path, "w") as geojson_convex_file:
    json.dump(convex_feature_collection, geojson_convex_file, indent=2)

# Zapisivanje failed clusters Feature Collection u GeoJSON datoteku
with open(geojson_failed_path, "w") as geojson_failed_file:
    json.dump(failed_feature_collection, geojson_failed_file, indent=2)
```