SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Paula Kokić

SIMULACIJA POZICIONIRANJA AUTONOMNIH LETJELICA U PROSTORU

ZAVRŠNI RAD

Varaždin, 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Paula Kokić
Matični broj: 41987/13-R
Studij: Informacijski sustavi

SIMULACIJA POZICIONIRANJA AUTONOMNIH LETJELICA U PROSTORU

ZAVRŠNI RAD

Mentor:
prof.dr.sc. Neven Vrček

Varaždin, kolovoz 2016.

Sadržaj

1. Uvod

U posljednjih par godina, na području tehnologije, sve se više spominju pojmovi poput „autonomne letjelice“, „dronovi“, „Internet of Things“, „crowdsourcing“, itd... Upravo zato će se i ovaj rad orijentirati tom području, odnosno dotaknuti se barem jednog njegovog dijela i na taj način pokušati doprinjeti njegovom razvoju.

Dakle, ovaj rad bavit će se pozicioniranjem autonomnih letjelica, odnosno simulacijom njihovog pozicioniranja te ispravljanjem GPS pogreške njihove pozicije, tj. lokacije. Naime, radi se o tome da postoji mogućnost minimiziranja pogreške koju pojedini čvor dobije skupa s lokacijom od GPS sustava i to očitavanjem jačine WiFi signala drugih čvorova koji se nalaze u njegovoj blizini. Prema dobivenim očitavanjima, računat će se udaljenost između čvorova te na temelju toga korigirati lokacija pojedinog čvora, odnosno smanjiti pogreška GPS sustava.

Rad će najprije pojasniti teoretski dio, tj. detaljnije objasniti sve pojmove koji će nam biti potrebni za shvaćanje problema i algoritma, a to su: GPS sustav, pojam autonomne letjelice i RF propagacije. Nakon toga bit će iznesena hipoteza na kojoj se zasniva ovaj rad te nakon toga prikazan i opisan rad aplikacije te rezultati koji su nastali korištenjem aplikacije.

2. Pregled literature

2.1. GPS

GPS (eng. *Global Positioning System*) je američki sustav koji korisnicima omogućava pozicioniranje, navigaciju i vremenske usluge. Razvoj ove tehnologije započeo je još u ranim 70-tima prošlog stoljeća, da bi danas već bilo korišten od gotovo svih uređaja kojima je potrebno odrediti lokaciju na Zemlji. Sustav se može koristiti neovisno o vremenskim prilikama, dobu dana ili lokaciji na Zemlji, važno je samo da pojedini čvor ima nesmetan signal prema najmanje četiri satelita GPS-a. (GPS.GOV, 2016)

Postoje tri glavna segmenta rada GPS-a: svemirski, kontrolni i korisnički. Svemirski segment sastoji se od 24-31 satelita, koji se nalaze u srednjoj Zemljinoj orbiti i korisnicima odašilju radio signale. Svaki satelit dnevno obiđe Zemlju dva puta, ali na način da su u svakom trenutku, na gotovo svakoj lokaciji na Zemlji dostupna minimalno četiri satelita kako bi sustav mogao pravilno raditi. Kontrolni sustav sastoji se svjetske mreže koja prati GPS satelite, s njima razmjenjuje podatke, nadzire njihove transmisije i obavlja analize. Konačno, korisnički segment sastoji se od samih GPS prijemnika, bilo samostalnih, bilo ugrađenih u različite uređaje, koji očitavaju signale GPS satelita i određuju lokaciju korisnika. (GPS.GOV, 2016)

Korisnički segment GPS sustava dovodi upravo do njegove primjene. Glavna podjela upotrebe GPS-a je na civilne (dostupne svim građanima na Zemlji) i vojne svrhe (dostupne američkoj vojsci i njihovim vojnim saveznicima). Budući da je ova usluga besplatna, otvorena i poprilično neovisna, omogućila je razvoj stotine aplikacija bez kojih se današnji život ne bi mogao ni zamisliti, krenuvši od mobitela i pametnih telefona, preko ručnih satova do buldožera, bankomata i sl. Ovaj sustav koristi se u poljoprivredi, građevini, rudarstvu, dostavljanju paketa i logističkim sustavima, bankarskim sustavima, komunikacijskim mrežama... (GPS.GOV, 2016) O primjeni GPS sustava može se danima govoriti, no to nije glavna tema ovog rada. Umjesto toga, u nastavku se iznosi zašto je ovom radu potreban GPS sustav i kako će se koristiti.

Svaki GPS prijemnik (pa tako i oni koji se nalaze u autonomnim letjelicama), od GPS satelita prima veću količinu podataka, a najbitnije su one koje se tiču same lokacije: geografska širina (eng. *latitude*) i geografska dužina (eng. *longitude*). Osim toga, jako je bitan i podatak GPS pogreške (eng. *error*) jer ćemo se njime najdetaljnije baviti. Geografska širina i dužina izražene su u stupnjevima, minutama i sekundama, dok je greška prikazana u metrima.

2.2. Autonomna letjelica

Pojam autonomna ili bespilotna letjelica (eng. *Unmanned Aerial Vechile – UAV*) odnosi se na letjelicu (zrakoplov) bez posade, odnosno pilota koji leti s njom. Takvu letjelicu se može nadzirati i upravljati na daljinu (ručno), a neke imaju i sposobnost autonomno letjeti pomoću unaprijed isprogramiranih planova leta ili kompeksnih dinamičkih automatskih sustava. Letjelice su često opremljene različitim senzorima, kamerama, odašiljačima i prijemnicima (kao što je GPS prijemnik ili RF antena), ovisno o svrhi/zadaći koju obavlja. A zadaće ovih letjelica su jednako različite: koriste se u vojne svrhe, ali i civilne – za snimanje iz zraka, dostavu, različita istraživanja, spašavanje, itd. (TheUAV.com, 2016) U ovom radu, koristit će se još pojam dron (eng. *drone*), kojeg različiti izvori drugačije definiraju. Dok ga neki izvori jednostavno poistovjećuju s pojmom bespilotne letjelice (Sayler, 2015), drugi dron smatraju letjelicom s apsolutnom autonomijom (Villasensor, 2012). Budući da za algoritam nije važno stanje autonomije, jednako će se koristiti oba pojma.

Razina autonomije je dio koji se zapravo još uvijek razvija, tako da čovjek ima sve manju ulogu u upravljanju samom letjelicom. Neka od područja koja će biti značajna za napredak autonomije jesu: spajanje podataka iz senzora, komunikacije između pojedinih letjelica u svrhu nadopunjavanja i korigiranja podataka, planiranje kretanja u svrhu pronalaska optimalnog puta između dvije točke (kako prijeći određene zapreke, itd.), raspodjela posla i kooperacija između pojedinih letjelica kako bi se najoptimalnije riješili dani zadaci ili maksimizirala šansa za uspjeh u danoj misiji. (TheUAV.com, 2016)

2.3. RF propagacija

Radio frekvencija (eng. *radio frequency*, RF) definira se kao bilo koja frekvencija elektromagnetskog vala koja leži u rasponu između 3 kHz i 300 GHz, a koristi se u komunikacijske svrhe ili kao signal za radar. (Merriam Webster, 2016) Nas zapravo najviše zanimaju frekvencije od 2.4 GHz i 5GHz, jer na njima leži Wi-Fi signal, koji će odašiljati i primati naši čvorovi, tj. autonomne letjelice.

RF propagacija je pojam koji označava način na koji se radio valovi šire između dvije točke na Zemlji. Zbog prirode valova, odnosno njihovih fizičkih svojstava, propagacija signala može se iskoristiti za izračun udaljenosti između dvije točke. Jedna od te dvije točke je pošiljatelj signala – točka koja odašilje signal. Potrebno je naglasiti da se svaki signal širi iz jedne točke određenom jačinom (mjereno u decibel-miliwatima, dBm; označava električnu snagu u dB koja odgovara 1 mW), i šireći se kroz prostor njegova jačina slabi. Signal nikad

neće prestati, samo će slabiti, ali postoji prag, tj. minimalna jačina signala koju neki prijemnik može očitati, a ona iznosi oko -90 dBm. (Experts Exchange, 2013) Prema tome, druga točka – primatelj – očitava jačinu signala pošiljatelja, tj. ako je ona veća od -90dBm primatelj može „vidjeti“ pošiljatelja. Formula koja će se koristiti za izračun udaljenosti između dva čvora je tzv. formula za gubitke propagacije u neomeđenom prostoru (eng. *free-space path loss,* FSPL), a glasi ovako:

$$r = 10^{\frac{\ln(10)\times(T-R-K-20\log(f))}{10n}}$$

gdje je:

- R – jačina primljenog signala, mjeri se putem odgovarajuće WiFi opreme na letjelici (dBm)
- T – jačina odašiljanja signala, iznosi 17 dBm
- K – konstanta gubitka propagacije, iznosi -147.55
- f – frekvencija WiFi signala, iznosi 2450 MHz
- n – eksponent gubitka propagacije, iznosi 2, jer se radi o letjelicama koje lete u slobodnom prostoru
- r – udaljenost između dva čvora (rezultat u metrima, m)

(Tomaš, 2013)

Prikupljajući na taj način podatke o udaljenosti između čvorova, tj. onih čvorova koji se međusobno „vide“, odnosno mogu očitati jedan drugome jačinu signala, pokušat će se preciznije odrediti lokacija pojedninog čvora i paralelno smanjiti GPS pogreška.

2.4. Masovna podrška

Masovna podrška (eng. *crowdsourcing*) termin je koji predstavlja postupak dobivanja potrebnih usluga, ideja ili podataka od neodređene skupine ljudi. (Frančula, 2015) Prema tome, crowd sourced sustavi (sustavi masovne podrške) su oni sustavi koji se sastoje od jedinica koje međusobno samostalno razmjenjuju podatke.

Dobar primjer crowdsourcinga je motivirajući sustav za prikupljanje podataka s pametnih telefona (eng. *Incentive Mechanism Design for Mobile Phone Sensing*), kojeg su razvili znanstvenici s kineskih sveučilišta Arizona State i Syracuse. Budući da je prodaja pametnih telefona premašila svaka očekivanja i nadišla prodaju prijenosnih računala,

prikupljanje podataka također postaje puno veće područje istraživanja i napretka. Osim toga, današni pametni telefoni su opremljeni sa setovima jeftinih, ali moćnih i snažnih senzora, poput brzinomjera (eng. accelerometer), digitalnim kompasom, žiroskopom, GPS-om, mikrofonom, kamerom, itd. Takvi senzori mogu omogućiti nadzor nad velikim rasponom ljudskih aktivnosti i okoline. Međutim, postojeći sustavi za prikupljanje podataka s mobilnih i pametnih telefona nedovoljno motiviraju korisnike da im omoguće svoje servise za prikupljanje podataka. Naime, kada korisnici i odluče sudjelovati u takvom sustavu prikupljanja podataka s njihovog uređaja, moraju koristiti vlastite resurse, kao što je potrošnja više energije i procesorska snaga, ali i riskiraju svoju privatnost. Prema tome, korisnici ni ne mogu biti zainteresirani za sudjelovanje u takvom sustavu, osim ako nisu dobro nagrađeni za to. Zato je tim stručnjaka, D.Yang, G.Xue, X.Fang i J.Tang osmislio novi, motivirajući sustav za prikupljanje podataka s pametnih i mobilnih telefona, u čak dvije varijante. Prvi mehanizam je platformno-orijentiran (eng. *platform-centric*), gdje platforma, koja se nalazi na serverima, ima potpunu kotrolu nad plaćanjem svim korisnicima, a korisnici mogu samo prilagoditi svoje radnje kojima će pružati uslugu platformi. Drugi mehanizam je korisnički-orijentiran (eng. *user-centric*) u kojem svaki korisnik može definirati svoju najnižu cijenu po kojoj želi davati svoje usluge i tada platforma odabire korisnike i plaća im po iznosu ne manjem od navedene cijene. (Yang i dr., *s.a.*)

U ovom radu, crowdsourcing se prepoznaje prema razmjeni podataka između autonomnih letjelica. Svaka letjelica šalje svoje podatke sa senzora svim drugim letjelicama koje u u njenom dometu, između ostalog i svoju lokaciju prema GPS-u, odnosno pogrešku u preciznosti prema GPS-u. Te informacije koriste ostale letjelice kako bi algoritmom koji će biti definiran u ovom radu, pokušale smanjiti GPS pogrešku letjelice pošiljatelja i time omogućiti preciznije lociranje svake pojedine letjelice.

3. Hipoteze i ciljevi

Iznosi se hipoteza ovog rada:

H1: *Koristeći RF propagacijski model može se minimizirati GPS greška u crowd sourced sustavima.*

Ciljevi:

- simulirati kreatnje jata dronova u otvorenom prostoru
- koristeći RF propagacijski model izračunati udaljenosti između dronova
- razviti algoritam za minimiziranje GPS pogreške jata dronova
- dobiti precizniju lokaciju svakog pojedinog drona unutar jata

4. Model drona

U sustavu koji će biti opisan u sljedećim poglavljima ovog rada, koristit će se definirani model drona kako slijedi.

Svaki dron predstavlja jedan čvor (eng. *node*), kojeg algoritam promatra kao posebnu jedinicu sa svojim svojstvima. Svaki dron ima sposobnost kretanja u 2-dimenzionalnom prostoru, te ga obilježavaju sljedeća svojstva: ID, naziv, lokacija (x,y,z) (z os koja predstavlja visinu, neće se koristiti u algoritmu, pa se neće dalje u radu niti spominjati), greška po x osi, greška po y osi. Za potrebe simulacije, svakom čvoru će se definirati smjer i brzina kretanja. Svi čvorovi su opremljeni GPS prijemnikom, od kojeg dobivaju podatke o lokaciji (x,y) i grešci po x i y osi. Osim GPS prijemnika, svi čvorovi opremljeni su RF antenom, odnosno imaju mogućnost slanja i primanja podataka preko WiFi signala, točnije preko P2P (eng. *peer-to-peer*) mesh mreže. Prema tome, svaki čvor za sebe ima definiranu i listu susjednih čvorova koje „vidi“, odnosno prepoznaje prema jačini signala. Upravo na taj način čvorovi imaju sposobnost međusobno slati podatke, odnosno razmjenjivati informacije o lokaciji i GPS pogrešci. Algoritam u određenom vremenskom intervalu dobiva svaki put nove podatke o svakom dronu i na temelju toga radi izračune. Jedina razlika između simuliranih i realnih podataka je u koordinatnom sustavu: simulirani podaci (lokacije dronova koje se dobivaju putem definiranog smjera i brzine kretanja te GPS pogreška koja je također unaprijed definirana) rade s Kartezijevim koordinatnim sustavom (i mjernim jedinicama u pikselima), dok stvarni podaci koriste geografske koordinate (geografsku širinu i dužinu, izraženu u geografskim stupnjevima). Međutim, to neće utjecati na rad algoritma jer ti sustavi služe samo za definiranje lokacije, odnosno pozicije drona te se njihove specifičnosti i zakonitosti uopće ne koriste u radu algoritma.

5. Algoritam za korekciju

U nastavku je dan opis funkcioniranja algoritma za korekciju GPS pogreške.

Ulazni parametri za algoritam jesu:

- GPS lokacija čvora (x,y): x predstavlja geografsku dužinu, a y geografsku širinu; obje točke izražene u geografskim stupnjevima (°) u decimalnom obliku
- GPS pogreška (x,y): x predstavlja pogrešku po x osi, a y pogrešku po y osi; obje duljine izražene u metrima (m), u decimalnom obliku
- podaci od svih dronova koji čine listu vidljivih dronova navedenog čvora (također trenutna GPS lokacija (x,y,) i GPS pogreška (x,y) – vrijede ista pravila o mjernim jedinicama), te jačine signala svih vidljivih dronova.

  Budući da se gotovo svi ovi podaci na neki način simuliraju, način na koji se to čini bit će opisan u idućem potpoglavlju.

Izlazni parametar jest određena regija (nepravilan geometrijski lik), koja predstavlja površinu na kojoj se može nalaziti čvor kojemu se korigira lokacija.

Cilj algoritma jest da izlazni parametar, tj. dobivena regija bude manja od početne regije (elipse) koju čine početni podaci o trenutnoj lokaciji i pogrešci (iz ulaznih parametara).

Pseudokod algoritma je:

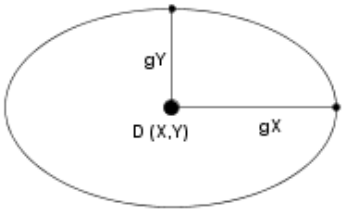
1. na temelju jačine RF signala napravi listu „vidljivih“ čvorova
2. izračunaj vlastitu elipsu pogreške
3. za svaki vidljivi čvor učini sljedeće:
   a. pomoću očitane jačine signala izračunaj udaljenost do vidljivog čvora
   b. prihvati podatke o lokaciji vidljivog čvora
   c. izračunaj širinu i visinu pomoću udaljenosti i greške vidljivog čvora za „malu“ elipsu
   d. izračunaj širinu i visinu pomoću udaljenosti i greške vidljivog čvora za „veliku“ elipsu
   e. označi prostor (vijenac) između izračunate „male“ i „velike“ elipse

4. izračunaj presjek između vlasite elipse pogreške i svih izračunatih „vijenaca“ – dobiveni presjek rezultat je algoritma i predstavlja korigiranu pogrešku

U nastavku su detaljno opisane sve faze, odnosno koraci navedenog algoritma.

Prvi korak – izrada liste „vidljivih“ čvorova. Čvor očitava podatke, tj. jačinu signala svakog čvora sa svog WiFi uređaja i na temelju toga izrađuje se lista „vidljivih“ čvorova.

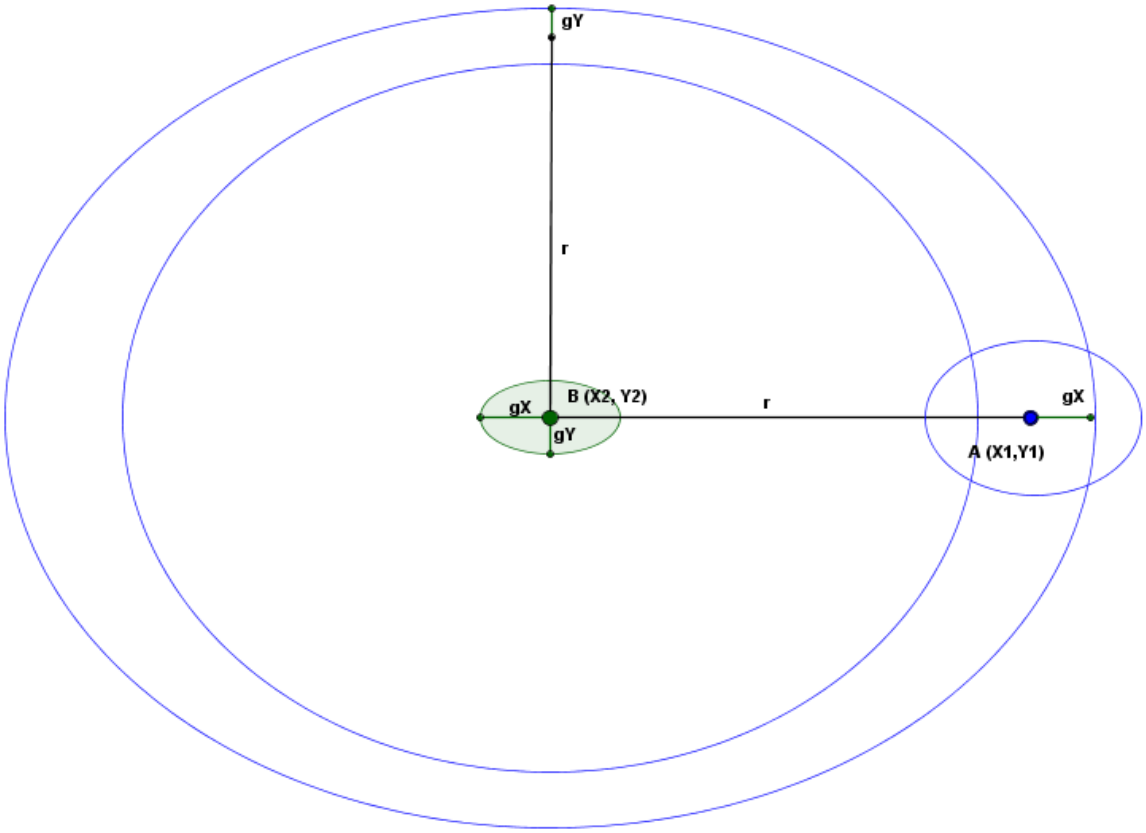
Drugi korak – izrada elipse pogreške. S GPS prijemnika očitavaju se podaci o lokaciji promatranog čvora te njegove x,y pogreške. Prema tome, elipsa pogreške bi bila elipsa koja opisuje čvor, na udaljenosti jednakoj grešci po x u vodoravnom smjeru, te udaljenosti jednakoj grešci po y u okomitom smjeru (vidi Sliku 1. – gY predstavlja vrijednost greške po y osi, analogno tome gX vrijednost greške po x osi, a točka D(X,Y) predstavlja lokaciju čvora) Dakle, površina ove elipse predstavlja prostor u kojem bi se prema GPS sustavu trebao nalaziti promatrani čvor. Površina ove elipse uspoređivat će se s krajnjim rezultatom kako bi se dobio postotak poboljšanja preciznosti, odnosno smanjenja pogreške.



Slika 1. Elipsa pogreške čvora

Treći korak – izračun „vijenaca“. Algoritam prolazi sve vidljive čvorove promatranog čvora i učitava njihove podatke: lokaciju (x,y), grešku (gX, gY) i jačinu RF signala (R) koju prima od svakog. Na Slici 2. prikazan je čvor A s lokacijom (X1,Y1), kojemu će se korigirati lokacija, i čvor B s lokacijom (X2, Y2) kojeg čvor A „vidi“. Pomoću jačine signala najprije se računa udaljenost (r) do tog vidljivog čvora B, a pri tome koristi se formula za gubitke propagacije u neomeđenom prostoru koju smo spomenuli u ranijem poglavlju o RF propagaciji. Nakon toga uzimaju se u obzir greške po x i y vidljivog čvora B. Oko njega se kreira velika elipsa (sa središtem u točki koja predstvalja lokaciju čvora, (X2,Y2)), kojoj je polumjer po širini jednak zbroju izračunate udaljenosti r i greške po x (gX) čvora B, a polumjer po visini jednak zbroju izračunate udaljenosti r i greške po y (gY) čvora B. Prema istoj logici, kreira se i mala elipsa oko istog čvora, samo manjeg polumjera – umjesto zbroja riječ je o razlici između udaljenosti r i grešci po x gX, odnosno po y gY. Na taj način dobivamo dvije elipse sa svoje dvije površine. S matematičke strane, te dvije elipse možemo promatrati i kao dva skupa, recimo da je velika elipsa skup A, a mala elipsa skup B. Budući da je skup B pravi podskup skupa A ($B \subset A$), možemo nad njima primjeniti operaciju razlike. Prema tome, nakon primjene razlike ta dva skupa, odnosno kada se oduzme manji skup od većeg, tj. $A \setminus B$, dobiva se skup točaka koji se nalazi u skupu A, a ne nalazi u skupu B – točnije dobiva se upravo ovaj

eliptični „vijenac“ ili traka oko čvora B. Ista takva traka nastala bi kad bi se uzeo marker oblika elipse pogreške čvora B i na jednakoj udaljenosti, bez zakretanja markera, povukla kružnica oko njega. Jedan dio tog vijenca, ili trake, svakako će prelaziti preko promatranog čvora A, za kojeg računamo korigiranu pogrešku, odnosno njegove elipse pogreške.



Slika 2. Odnos čvora A (primatelja) i čvora B (pošiljatelja)

Četvrti korak – izračun površine korigirane pogreške. U ovom koraku skupljaju se podaci iz prethodnog koraka, odnosno svi izračunati vijenci, odnosno skupovi. Budući da i dalje te vijence možemo smatrati skupovima, i svaki od njih prolazi jednim svojim dijelom preko skupa elipse pogreške čvora A, može se računati njihov presjek. Matematički se to može prikazati ovako:

$$A' = A \cap V_1 \cap V_2 \cap \ldots \cap V_n$$

pri čemu je:

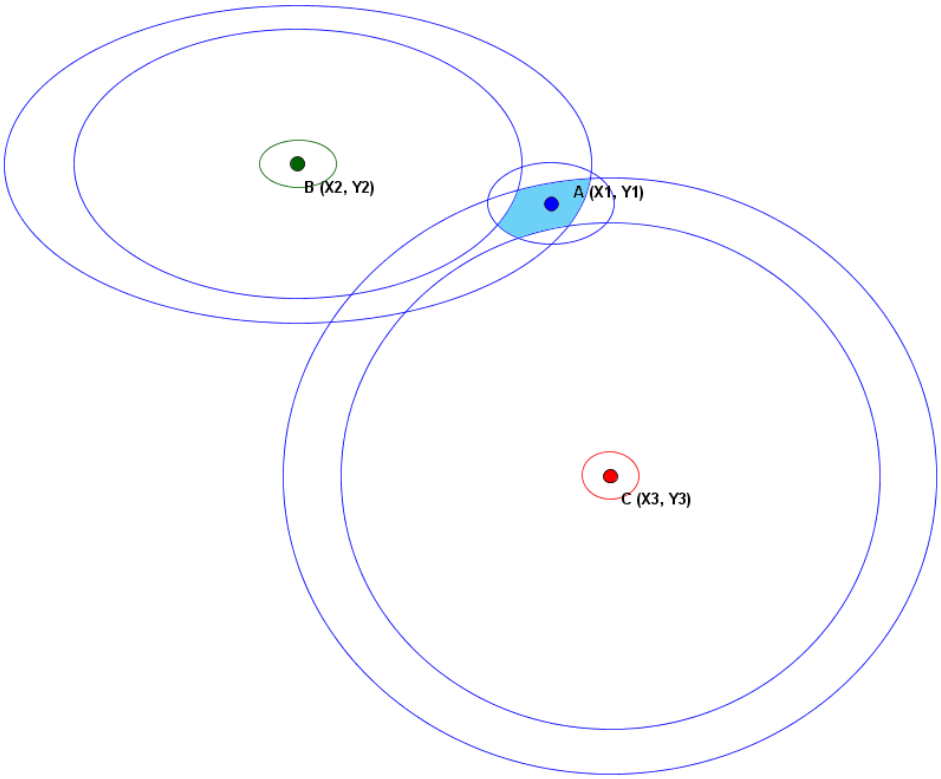
- A' – novi skup koji predstavlja korigiranu pogrešku GPS-a, rezultat algoritma
- A – početni skup GPS pogreške, čini ga skup točaka površine elipse pogreške

- V_1, V_2 ,...V_n – skupovi vijenaca izračunati prethodnim korakom, od svih vidljivih čvorova, gdje n predstavlja broj vidljivih čvorova.

Prema tome, izračunati presjek predstavlja korigiranu pogrešku promatranog čvora A. Na Slici 3. rezultat algoritma, odnosno presjek je osjenčan svijetlo-plavom bojom. U ovom primjeru su dva čvora (B i C) u vidljivom rasponu čvora A, tako da utječu na korekciju njegove pogreške. Naravno, u nastavku algoritma, i čvor A će utjecati na korekciju pogrešaka čvorova B i C, itd...

Zašto je odabran baš ovakav način računanja i zašto bi trebao funkcionirati? GPS lokacija (X,Y) i greška (gX, gY) koju prima promatrani čvor, odnosno elipsa pogreške koja je crtana oko čvora na prethodno objašnjeni način, znači da se promatrani čvor može nalaziti bilo gdje unutar te elipse. Kada se izračuna udaljenost r između dva čvora, i uzme u obzir da se čvor može nalaziti na jednom ili skroz suprotnom kraju elipse, dobivamo upravo ova dva (odnosno četiri) polumjera za izradu velike i male elipse koje će činiti vijenac, odnosno najbližu i najdalju granicu gdje se može nalaziti susjedni čvor.

Iz ovoga se također može zaključiti da algoritam najviše ovisi o GPS pogrešci čvorova i naravno broju samih čvorova (što je više čvorova, lokacija će biti preciznija). Osim toga, najpreciznije određeni čvor (onaj s najmanjom pogreškom) će najviše utjecati na korekciju drugih čvorova u njegovom dometu.



Slika 3. Rezultat algoritma - korigirana GPS pogreška

6. Aplikacija

6.1. Korisnički zahtjevi

Prije razrade samog algoritma, definirat će se korisnički zahtjevi, odnosno specifikacija i domene aplikacije koja će koristiti taj algoritam, kako bi ga mogli testirati.

Aplikacija, koju možemo nazivati i Simulator pozicioniranja dronova, treba ispunjavati sljedeće zahtjeve:

- mogućnost unošenja podataka o svakom dronu: ID, naziv, lokaciju u obliku x i y koordinata, početni smjer kretanja (u stupnjevima) i brzinu
- mogućnost dodavanja unesenog drona na listu dronova koji će biti u simulaciji
- mogućnost brisanja dodanih dronova
- mogućnost pokretanja, zaustavljanja/pauziranja i ponovnog pokretanja (reset) simulacije
- vizualni prikaz kretanja dronova, njihove GPS greške i greške korigirane algoritmom
- mogućnost spremanja dobivenih rezultata simulacije u vanjsku datoteku

6.2. Struktura sustava

U nastavku slijedi opis strukture sustava, odnosno način na koji je osmišljena aplikacija. Za bolji opis, za prikaz će se koristiti UML dijagram klasa, koji sadrži sve klase, atribute, metode i odnose među klasama.

Slika 4. prikazuje navedeni UML dijagram klasa koji opisuje aplikaciju. S obzirom da se radi o jako maloj aplikaciji, dijagram je poprilično jednostavan i sadrži samo šest klasa. Strukturno se sustav može razložiti na dva sloja: klase koje služe za ulaz i izlaz (prikaz) podataka, odnosno komunikaciju s korisnikom (a to su FrmGlavna i FrmIzlaz) te klase koje služe za samu obradu podataka (Program, Dron, KorekcijaPogreske, GeneratorGreske).

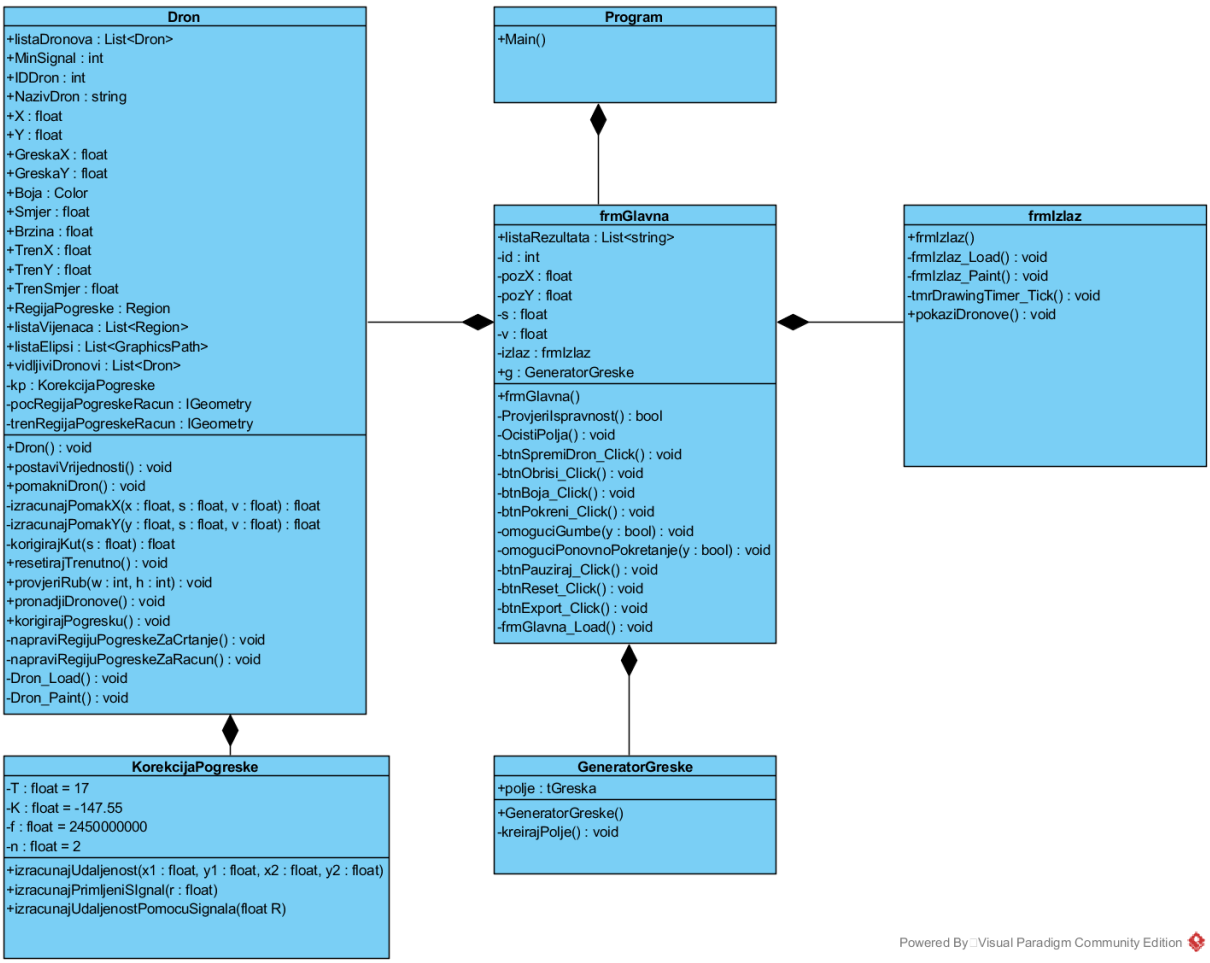
Klasa Program zapravo je početna klasa, iz koje se instancira objekt klase FrmGlavna.

Klasa FrmGlavna je klasa koja služi za prikaz početne forme, koja služi za unos podataka te sadrži glavne kontrole za upravljanje simulacijom. Prema tome, uglavnom sadrži metode za određene događaje (za klik miša na određeni gumb), te metode za uključivanje/isključivanje određenih gumbova, sigurnosnih provjera podataka i sl.

Klasa FrmIzlaz pokreće se odmah nakon glavne forme, a služi samo za prikaz čvorova, odnosno iscrtavanje njihovog kretanja. Sadrži i okidač pomoću kojeg se odvija cijela simulacija kretanja.

Klasa GeneratorGreske još je jedna klasa, čiji se objekt također jedino instancira unutar klase FrmGlavna. Ta klasa služi generiranju vrijendosti GPS pogreške i jednako iz nje čitaju (koriste je) svi objekti klase Dron.

Klasa Dron je klasa koja sadrži sve podatke o čvoru. Osim osnovnih obilježja (atributa), kao što su ID, naziv, lokacija (x,y), greška (x,y), boja (koristi se za prikaz), smjer kretanja i brzina, sadrži još neke atribute. Atributi TrenX i TrenY označavaju trenutnu lokaciju (x,y), atribut MinSignal označava najmanju jačinu signala koju dron može „čuti“ (rečeno je da je to -90dBm, ali ovdje je ostavljeno da se čak može i podesiti za svaki čvor drugačija vrijednost), RegijaPogreske označava GPS regiju pogreške (površinu elipse) koja će se crtati na zaslonu,



Slika 4. Dijagram klasa za aplikaciju Simulator pozicioniranja dronova

atribut kp je objekt klase KorekcijaPogreske (o kojoj će biti riječi kasnije), a služi za pristup metodama za izračun podataka za simulaciju. Tu su još dva atributa, pocRegijaPogreskeRacun

i trenRegijaPogreskeRacun, koja služe za pohranu samog izračuna površine pogreške (početne i trenutne). Osim ovih, tu su još neki atributi, odnosno liste. Lista dronova je statična lista koja predstavlja popis svih unesenih dronova. ListaVijenaca i listaElipsi služe za pohranu međurezultata (vijenaca i elipsi), za svaki pojedini dron posebno, odnosno omogućuju njihovo iscrtavanje na zaslon. Lista vidljiviDronovi je lista koja sadrži sve dronove koji promatrani dron „vidi“ (više o tome bit će rečeno u poglavlju 4.4. Algoritam za korekciju). Ako pogledamo metode, možemo izdvojiti tri skupine. Prva skupina metoda služi za osnovnu manipulaciju podacima, odnosno upravljaju događajima (metode postaviVrijednosti(), resetirajTrenutno(), Dron_Load(), Dron_Paint()). Druga skupina metoda služi za simulaciju kretanja dronova (metode pomakniDron(), izracunajPomakX(), izracunajPomakY(), korigirajKut(), provjeriRub()). Treća skupina metoda koristi se za izračun, tj. korekciju GPS pogreške (metode pronadjiDronove(), korigirajPogresku(), napraviRegijuPogreskeZaCrtanje(), napraviRegijuPogreskeZaRacun()).

Klasa KorekcijaPogreske je klasa koja služi za izračun simuliranih vrijednosti (metode izracunajUdaljenost() i izracunajPrimljeniSignal()) te same udaljenosti između čvorova (metoda izracunajUdaljenostPomocuSignala()) te kao atribute sadrži konstante koje se koriste u formulama.

Naravno, sve ovo će biti puno jasnije u idućim poglavljima, gdje se bude govorilo o samom algoritmu korekcije i računanju simuliranih podataka.

6.3. Simulacija ulaznih parametara

S obzirom na to da je ovaj rad fokusiran na sam algoritam korekcije, aplikacija (tj. algoritam) ne koristi prave ulazne parametre, već ih simulira.

Početne lokacije čvorova korisnik upisuje skupa s ostalim podacima o čvoru (smjer kretanja, brzina, naziv, itd.) Smjer se unosi u stupnjevima (između 0 i 360), a brzina predstavlja broj piksela koji će prijeći čvor u zadanom intervalu, a ono iznosi 30ms. Sve veličine izražene su u pikselima, tako i lokacija, greška, udaljenost i sl. Budući da se radi o grafičkom računalnom koordinatnom sustavu, ishodište (0,0) mu se nalazi u gornjem lijevom kutu i ono ne posjeduje druge kvadrante koji bi imali negativne vrijednosti po ijednoj osi. Za potrebe ovog rada, jedan piksel predstavlja jedan metar.

Definiranje lokacije, odnosno njene promjene, čija posljedica je kretanje samog čvora, simulirano je uz pomoć fiksno određene brzine i smjera koje korisnik unosi na početku. Sam izračun novih lokacija, tj. simulacija kretanja provedena je prema matematičkim zakonima

Pitagorinog poučka, uz pomoć sinusa i kosinusa. Čvorovi se kreću po fiksno određenoj veličini izlazne forme, a kad se dođe do ruba, čvor se odbija od njega pod istim kutem pod kojim je ušao.

Pogreška GPS-a (gX, gY) simulira se pomoću karte veličine izlazne forme, podijeljene na nekoliko nejednakih regija. Svaka regija ima svoje vrijednosti pogreške koje generira, dakle svi čvorovi koji se nalaze u istoj regiji imaju istu pogrešku. Na ovaj način se pokušava simulirati različit teren na kojem se mogu nalaziti čvorovi i njegova povezanost s GPS-om.

Jačina signala (R) simulira se zapravo preko iste formule za gubitke propagacije u neomeđenom prostoru iz udaljenosti čvorova. Udaljenost čvorova za ovu svrhu računa se na temelju lokacija oba čvora, pomoću matematičke formule za udaljenost dvije točke u koordinatnom sustavu.

6.4. Implementacija algoritma

Za implementaciju ovakve aplikacije odabran je jezik C#, odnosno .NET tehnologija i alat Microsoft Visual Studio te je napravljena Windows Forms aplikacija. Osim osnovnih paketa i biblioteka, preuzete su NetTopologySuite i GeoAPI biblioteke za lakši rad s geometrijskim likovima. Za korištenje aplikacije, potrebno je imati Windows operacijski sustav te podršku za .NET 4.5 Framework.

U nastavku je dana implementacija samog algoritm, odnosno njegov kod. Glavna metoda naziva se korigirajPogresku() i nalazi se unutar klase Dron, a poziva još dvije metode, napraviRegijuPogreskeZaCrtanje() i napraviRegijuPogreskeZaRacun(), čiji kod se nalazi ispod koda metode korigirajPogresku(). Naime, zbog prirode tehnologije .NET-a, napravljene su odvojene metode za crtanje samih čvorova, odnosno njihovih vijenaca i korigirane pogreške, i metode za sam izračun površina korigirane pogreške. Osim ove dvije metode, na početku se računaju i simulirani podaci koji su potrebni za algoritam, a to je udaljenost između čvorova, *rSim*, iz kojeg se onda računa i simulirana jačina signala, *R*. Te dvije metode, skupa s metodom za izračun udaljenosti – izracunajUdaljenostPomocuSignala() – nalaze se u klasi KorekcijaPogreske, a ovdje će njihov kod biti prikazan nakon prve tri metode.

```
public void korigirajPogresku()
{
    string zapis = String.Empty;

    napraviRegijuPogreskeZaCrtanje();

    listaElipsi.Clear();
    listaVijenaca.Clear();
```

```
    napraviRegijuPogreskeZaRacun();

    foreach (DronView d in vidljiviDronovi)
    {
        //simulacija podataka
        float rSim = kp.izracunajUdaljenost(this.TrenX, this.TrenY, d.TrenX, d.TrenY);
        float R = kp.izracunajPrimljeniSignal(rSim);

        //izracun udaljenosti izmedju dronova
        float r = kp.izracunajUdaljenostPomocuSignala(R);

        //izracun tocaka (gornja lijeva) i polumjera za elipse
        float maliRY = (r - d.GreskaY);
        float maliRX = (r - d.GreskaX);
        float malaTockaX = d.TrenX - (r - d.GreskaX);
        float malaTockaY = d.TrenY - (r - d.GreskaY);

        float velikiRY = (r + d.GreskaY);
        float velikiRX = (r + d.GreskaX);
        float velikaTockaX = d.TrenX - (r + d.GreskaX);
        float velikaTockaY = d.TrenY - (r + d.GreskaY);

        //mala elipsa
        Region vijenac = new Region();
        GraphicsPath gpeMala = new GraphicsPath();
        gpeMala.AddEllipse(malaTockaX, malaTockaY, maliRX * 2, maliRY * 2);
        listaElipsi.Add(gpeMala);

        GeometricShapeFactory gsf = new GeometricShapeFactory();
        gsf.Centre = new Coordinate(d.TrenX, d.TrenY);
        gsf.Height = maliRY*2;
        gsf.Width = maliRX*2;
        var malaRacun = gsf.CeateEllipse();

        //velika elipsa
        GraphicsPath gpeVelika = new GraphicsPath();
        gpeVelika.AddEllipse(velikaTockaX, velikaTockaY, velikiRX*2, velikiRY*2);
        listaElipsi.Add(gpeVelika);

        gsf.Height = velikiRY*2;
        gsf.Width = velikiRX*2;
        var velikaRacun = gsf.CeateEllipse();

        //vijenac
        vijenac.Intersect(gpeVelika);
        vijenac.Exclude(gpeMala);

        var vijenacRacun = velikaRacun.Difference(malaRacun);

        listaVijenaca.Add(vijenac);

        //intersect
        regijaPogreske.Intersect(vijenac);
        this.trenRegijaPogreskeRacun =
trenRegijaPogreskeRacun.Intersection(vijenacRacun);

    }

    //podaci za zapis
    float postotak = (float)trenRegijaPogreskeRacun.Area /
(float)pocRegijaPogreskeRacun.Area;
    postotak = (float)Math.Round(((1 - postotak)*100),4);
```

```
    float povrsinaPoc = (float)Math.Round(pocRegijaPogreskeRacun.Area, 4);
    float povrsinaZav = (float)Math.Round(trenRegijaPogreskeRacun.Area, 4);
    float XzaIspis = (float)Math.Round(this.TrenX, 4);
    float YzaIspis = (float)Math.Round(this.TrenY, 4);

    zapis = this.IDDron.ToString() + ";" + this.NazivDron.ToString() + ";" +
XzaIspis.ToString() + ";" + YzaIspis.ToString() + ";" + povrsinaPoc.ToString() + ";" +
povrsinaZav.ToString() + ";"+ postotak;
    frmGlavna.listaRezultata.Add(zapis);

       }

private void napraviRegijuPogreskeZaCrtanje()
{
    regijaPogreske.MakeEmpty();
    GraphicsPath gp = new GraphicsPath();
    gp.AddEllipse(this.TrenX - this.GreskaX, this.TrenY - this.GreskaY, this.GreskaX *
2, this.GreskaY * 2);
    regijaPogreske.Union(gp);
}

private void napraviRegijuPogreskeZaRacun()
{
    GeometricShapeFactory gsf = new GeometricShapeFactory();
    gsf.Centre = new Coordinate(this.TrenX, this.TrenY);
    gsf.Height = this.GreskaY * 2;
    gsf.Width = this.GreskaX * 2;
    var pocetnaRegijaRacun = gsf.CeateEllipse();

    GeometryTransformer t = new GeometryTransformer();
    this.pocRegijaPogreskeRacun = t.Transform(pocetnaRegijaRacun);
    this.trenRegijaPogreskeRacun = this.pocRegijaPogreskeRacun;
}

public class KorekcijaPogreske
{
    float T = 17;
    float K = -147.55f;
    float f = 2450000000;
    float n = 2;

    public float izracunajUdaljenost(float x1, float y1, float x2, float y2)
    {
        float r;
        float pom1 = (float)Math.Pow((x2 - x1), 2);
        float pom2 = (float)Math.Pow((y2 - y1), 2);
        r = (float)Math.Sqrt((pom1 + pom2));
        return r;
    }

    public float izracunajPrimljeniSignal(float r)
    {
        float R;
        float pom = ((float)Math.Log10((double)r)*10*n)/(float)Math.Log(10);
        R = T - pom - K - 20 * (float)Math.Log10(f);
        return R;
    }

    public float izracunajUdaljenostPomocuSignala(float R)
    {
        float r;
        float pom = T - R - K - 20 * (float)Math.Log10((double)f);
```

```
            float pot = (float)(Math.Log(10) * pom) / (10 * n);
            r = (float)Math.Pow(10, pot);
            return r;
        }

}
```

Složenost ovog algoritma prilično je jednostavna za odrediti. Vidimo da postoji zapravo samo jedna *for* petlja unutar cijele metode korigirajPogresku(), koju treba uzeti u obzir prilikom računa, a sve ostalo su naredbe konstantne složenosti. U najgorem slučaju, broj dronova *d* u *foreach* petlji jest ukupan broj dronova *n* umanjen za jedan (taj dron koji promatramo), odnosno *n-1*. Tada matematičkim računom dolazimo do sljedećeg:

$$C(n) = \sum_{i=1}^{n-1} 1 = n - 1 - 1 + 1 = n - 1 \quad \in O(n)$$

Prema tome, u najgorem slučaju, složenost jednog poziva ovog algoritma jest O(n). No, budući da se ovaj algoritam poziva za svaki dron unesen u aplikaciju, možemo izračunati i složenost cijelog sustava koji služi za izračun svih dronova u jatu. Dakle, postoji još jedna *foreach* petlja iznad ovog algoritma, koja se u svakom slučaju izvršava *n* puta (gdje *n* predstavlja ukupan broj dronova), pa se dobiva sljedeće:

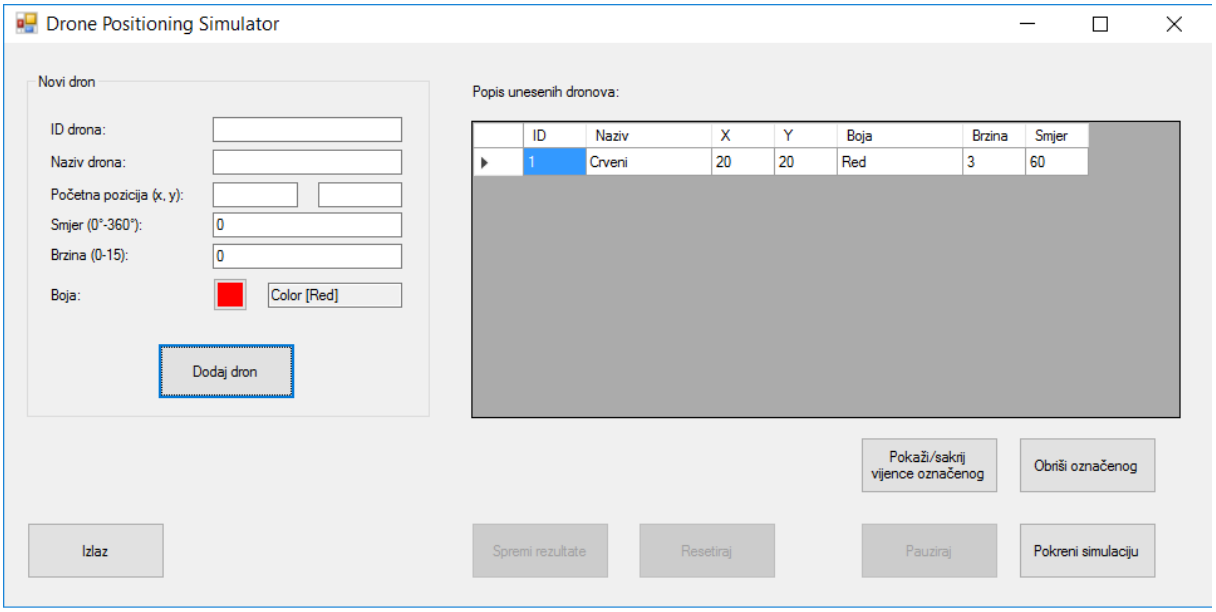
$$C(n) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} 1 = \sum_{i=1}^{n} n - 1 = (n-1) \sum_{i=1}^{n} 1 = (n-1)(n-1+1) = n(n-1) \quad \in O(n^2)$$

Složenost algoritma cijelog sustava za izračun, tj. korekciju pogreške GPS-a svih dronova u jatu jest $O(n^2)$.

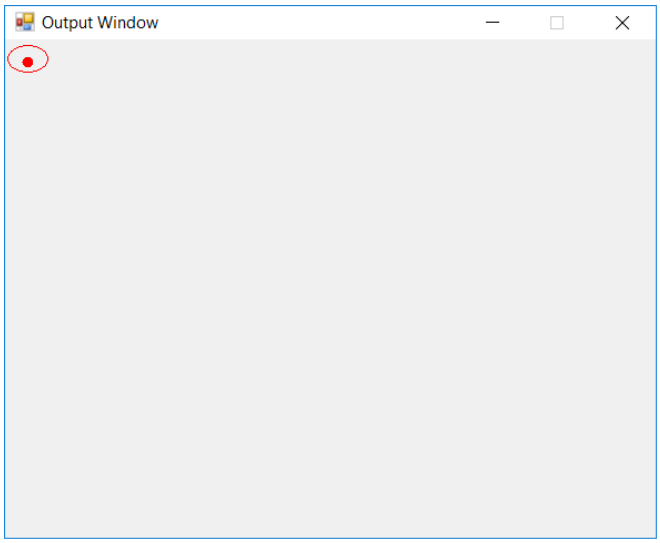
6.5. Korištenje aplikacije

Nakon pokretanja aplikacija otvaraju se dvije jednostavne forme, jedna pored druge. Prva je glavna forma (o kojoj je bilo riječ u poglavlju 4.2. Struktura sustava), a njen izgled može se vidjeti na Slici 5. Kao što možemo vidjeti, s lijeve strane nalazi se prostor za unos podataka o dronu: ID, naziv, početna pozicija, smjer (0-360), brzina (0-15) i boja. Klikom na gumb 'Dodaj dron' dron se dodaje u listu prikazanu s desne strane.

Nakon dodavanja drona, na raspolaganju stoje ostale mogućnosti. Brisanje drona izvršava se označavanjem reda drona kojeg se želi obrisati i klikom na gumb 'Obriši označenog'. Ako treba pokazati ili sakriti vijence tijekom simulacije, potrebno je označiti željeni dron i kliknuti na gumb 'Pokaži/sakrij vijence označenog'. Gumb 'Pokreni simulaciju' pokreće simulaciju kretanja, izračunavaju se pogreške dronova, a prikaz kretanja dronova iscrtava se na drugom prozoru, „Output Window“ (vidi Sliku 6.). Gumb 'Pauziraj' zaustavlja, točnije samo pauzira trenutnu simulaciju, a moguće ju je nastaviti ponovnim klikom na gumb 'Pokreni simulaciju'. Gumbom 'Resetiraj' možemo resetirati cijelu simulaciju i vratiti dronove na početna mjesta. Gumb 'Spremi rezultate' sprema rezultate simulacije u vanjsku .csv datoteku.



Slika 5. Glavna forma aplikacije



Na Slici 6. možemo vidjeti formu za iscrtavanje dronova na kojoj je trenutno prikazano početno stanje unesenog drona, za postavke kakve su navedene u glavnoj formi.

Slika 6. Forma za iscrtavanje kretanja dronova

7. Simulacija

U nastavku rada bit će opisano nekoliko simulacija, prema broju unesenih čvorova, te prikazane potpune postavke i krajnji rezultati simulacija. Karta simulirane GPS greške u svim simulacijama bit će ista, kako ne bi utjecala na rezultate algoritma. Početne lokacije prvih čvorova, kao i njihov smjer i brzina, bit će iste u svim simulacijama, kasnije će se samo dodavati novi čvorovi. Osim toga, definirano je trajanje simulacije na 30s. Mjerenje vremena se vrši ručnom štopericom. Jedan zapis (korekcija) vrši se nakon svakog izvršavanja algoritma, i to za svaki pojedini čvor. Svi zapisi se na pritisak gumba „Spremi rezultate“, spremaju u jednu .csv datoteku, sa sljedećim podacima, odvojeni separatorom točka-zarez (;) : ID drona, naziv drona, trenutna lokacija (x), trenutna lokacija (y), vrijednost GPS površine elipse pogreške, vrijednost površine regije korigirane pogreške (u nastavku: korekcija), postotak poboljšanja.

Postotak poboljšanja površine regije greške računa se prema sljedećoj formuli (koja se vidi i implementirana u algoritmu u prethodnom poglavlju):

$$p = \left(1 - \frac{P_{tren}}{P_{poc}}\right) \times 100$$

pri čemu je:

- p – konačni rezulat poboljšanja, u %
- P_{tren} – trenutna površina regije pogreške
- P_{poc} – početna površina regije pogreške (površina elipse pogreške), izračunata iz GPS podataka o pogrešci

Postotak poboljšanja površine regije greške govori nam za koliko je u postotcima novoizračunata površina pomoću algoritma manja od početne površine greške.

Za praćenje promjena rezultata simulacije, pratit će se uvijek isti kriteriji, a to su:

- Ukupan broj zabilježenih korekcija – označava ukupan broj zapisa (poziva algoritma)
- Najmanja vrijednost poboljšanja površine korekcije
- Najveća vrijednost poboljšanja površine korekcije
- Broj poboljšanja s vrijednošću 0%
- Broj poboljšanja s vrijednošću >20% i broj poboljšanja s vrijednošću >50%
- Prosječni postotak poboljšanja – zbroj svih poboljšanja podijeljen na ukupan broj zapisa

7.1. Scenarij 1

Broj čvorova u ovom scenariju je 2 (minimalan mogući za rad algoritma). U danoj Tablici 1. prikazani su uneseni čvorovi.

Tablica 1. Postavke prve simulacije - scenarij 1

ID	**Naziv**	**X**	**Y**	**Boja**	**Brzina**	**Smjer**
1	Crveni	20	20	Red	4	60
2	Plavi	40	100	Blue	4	100

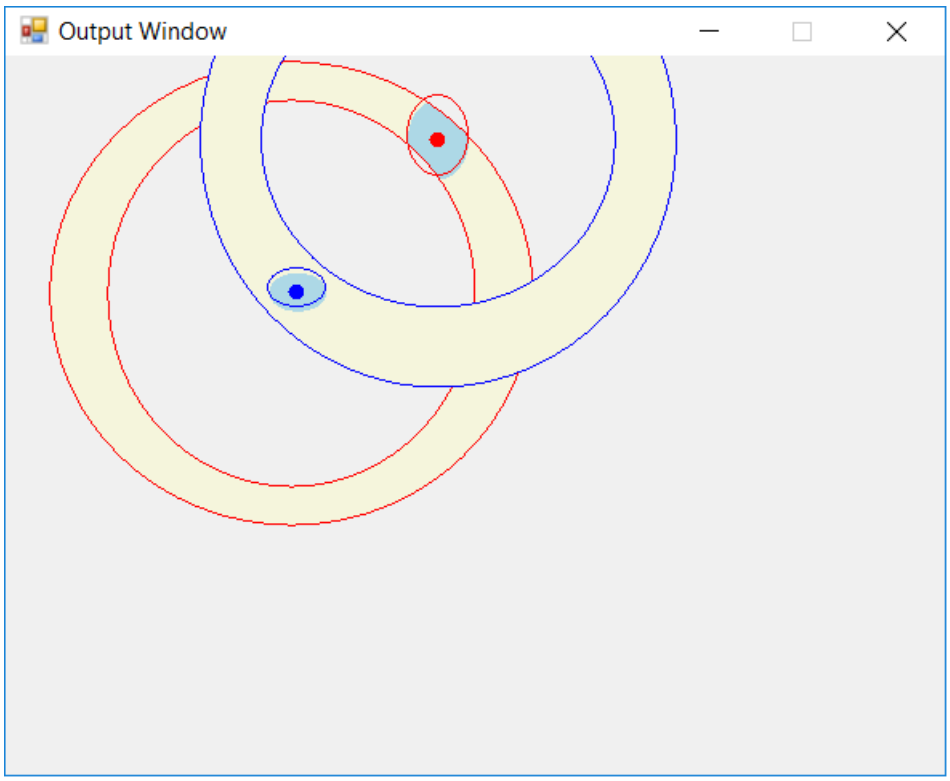
Nakon simulacije, rezultati su spremljeni u csv datoteku i mogu se lako detaljnije analizirati. Analizom csv datoteke dolazi se do sljedećih rezultata prikazanih u Tablici 2.

Tablica 2. Analiza rezultata prve simulacije

Obilježje / Kriterij	**Rezultat**	**Postotak (%)**
Ukupan broj zabilježenih korekcija	1848	-
Najmanja vrijednost poboljšanja površine korekcije	0%	-
Najveća vrijednost poboljšanja površine korekcije	63,6475%	-
Broj poboljšanja s vrijednošću 0%	675	36,53%
Broj poboljšanja s vrijednošću >20%	470	25,43%
Broj poboljšanja s vrijednošću >50%	77	4,17%
Prosječni postotak poboljšanja	11,55%	

Iz tablice se može vidjeti učinak, odnosno uspješnost algoritma. Ukupan broj zapisa (korekcija) bio je 1848. Od toga, 675 zapisa (36,53%) zapravo ima vrijednost 0, odnosno korekcije nije bilo. Do takvog rezultata dolazi kada je pojedini čvor preciznije lociran (ima manju GPS pogrešku) od susjednog čvora i tada zapravo presjek površine njegove elipse pogreške i površine vijenca susjednog čvora rezultira opet njegovom cijelom elipsom pogreške jer je ona manja, odnosno uža od širine vijenca. To se vidi na Slici 7, gdje je plavi čvor preciznije lociran pa algoritam neće moći smanjiti njegovu pogrešku, dok će pak on utjecati na crveni čvor i algoritam će smanjiti njegovu pogrešku. Nadalje, vidi se da je najveće poboljšanje iznosi 63,6475%, što znači da je dobivena površina za čak nešto više od 63% manja od originalne, tj. površine pogreške GPS sustava. Prikazan je i broj poboljšanja čija je vrijednost veća od 20%, kojih je 470, od ukupnih 1848, što u postotku iznosi 25,43%. Također je prikazan broj poboljšanja veći od 50%, takvih je 77/1848, odnosno 4,17%. Dakle, ovo su bile usporedbe

učinjenih korekcija naspram gotovo svih zapisa, uključujući i one koje zapravo iznose 0. Uzimajući u obzir da će takvih vrlo lako uvijek biti, možemo uspoređivati ove brojke i s ukupnom brojkom uspješnih poboljšanja, odnosno onih koje nisu 0, pa se dobiju još rezultati – 470/1173 = 40,07% za broj poboljšanja iznad 20%, te 77/1173 = 6,56% za broj poboljšanja iznad 50%. Prosječni postotak poboljšanja iznosi 11,55% (uključujući i rezultate s vrijednostima jednakim 0), odnosno 18,19% (bez poboljšanja s vrijednostima jednakim 0).



Slika 7. Prikaz Output prozora tijekom simulacije

7.2. Scenarij 2

Broj čvorova u ovom scenariju je 4, odnosno dupli u odnosu na prošlu simulaciju. U danoj Tablici 3. prikazani su uneseni čvorovi.

Tablica 3. Postavke druge simulacije – scenarij 2

ID	Naziv	X	Y	Boja	Brzina	Smjer
1	Crveni	20	20	Red	4	60
2	Plavi	40	100	Blue	4	100
3	Zeleni	30	150	Green	4	200
4	Ljubicasti	40	200	Violet	4	75

Nakon simulacije rezultati su ponovno spremljeni u csv datoteku te je napravljena njihova analiza prema istim kriterijima. Rezultati analize prikazani su u Tablici 4.

Tablica 4. Analiza rezultata druge simulacije

Obilježje / Kriterij	Rezultat	Postotak (%)
Ukupan broj zabilježenih korekcija	3860	-
Najmanja vrijednost poboljšanja površine korekcije	0%	-
Najveća vrijednost poboljšanja površine korekcije	78,7964%	-
Broj poboljšanja s vrijednošću 0%	583	15,10%
Broj poboljšanja s vrijednošću >20%	1500	38,86%
Broj poboljšanja s vrijednošću >50%	501	12,98%
Prosječni postotak poboljšanja	18,87%	

Kao što su pokazali rezultati analize, druga simulacija s četiri čvora pokazala je puno bolje rezultate. Najveći napredak vidi se u smanjenju broja nekorigiranih pogreški, odnosno smanjenje broja nula kao vrijednosti postotka poboljšanja. Unatoč i malo više nego duplom broju zapisa, ovaj put broj nula pao je na 583, odnosno s 36,53% na samo 15,10%. Isto tako značajno su se povećali i postotci broja korekcija s poboljšanjem – tako on za poboljšanja veća od 20% iznosi 38,86%, a za poboljšanja veća od 50% čak 12,98%. Osim toga, povećala se i najveća vrijednost poboljšanja površine korekcije, pa ona sad iznosi čak 78,7964%.

Kada bismo ovaj puta gledali samo poboljšanja koja su veća od 0, dobili bismo prosječno poboljšanje od 22,23%, postotak korekcija s poboljšanjem većim od 20% iznosio bi 45,77%, a postotak korekcija s poboljšanjem većim od 50% iznosio bi čak 15,28%.

7.3. Scenarij 3

Broj čvorova u ovom scenariju postavit će se na 8, odnosno četiri puta više u odnosu na početnu simulaciju. U danoj Tablici 5. navedeni su svi uneseni čvorovi.

Tablica 5. Postavke treće simulacije - scenarij 3

ID	Naziv	X	Y	Boja	Brzina	Smjer
1	Crveni	20	20	Red	4	60
2	Plavi	40	100	Blue	4	100
3	Zeleni	30	150	Green	4	200
4	Ljubicasti	40	200	Violet	4	75
5	Narancasti	50	250	Orange	4	210
6	Smedi	200	250	Brown	4	20
7	SvjZeleni	200	120	Lime	4	20
8	Rozi	300	300	Fuchsia	4	300

Izvršenje ove simulacije teklo je sporije nego prethodne dvije zbog većeg broja čvorova i puno više poziva algoritma, što je izazvalo veće opterećenje računala. Iz tog razloga u istom vremenskom periodu, napravljeno je manje zapisa, odnosno manje korekcija, ali podaci se unatoč tome mogu smatrati validnima jer vremenski raspon uzimanja uzorka, točnije broj uzoraka, može samo u jako malim postotcima poboljšati/pogoršati rezultat. U navedenoj Tablici 6. prikazani su rezultati analize za scenario br. 3.

Tablica 6. Analiza rezultata treće simulacije

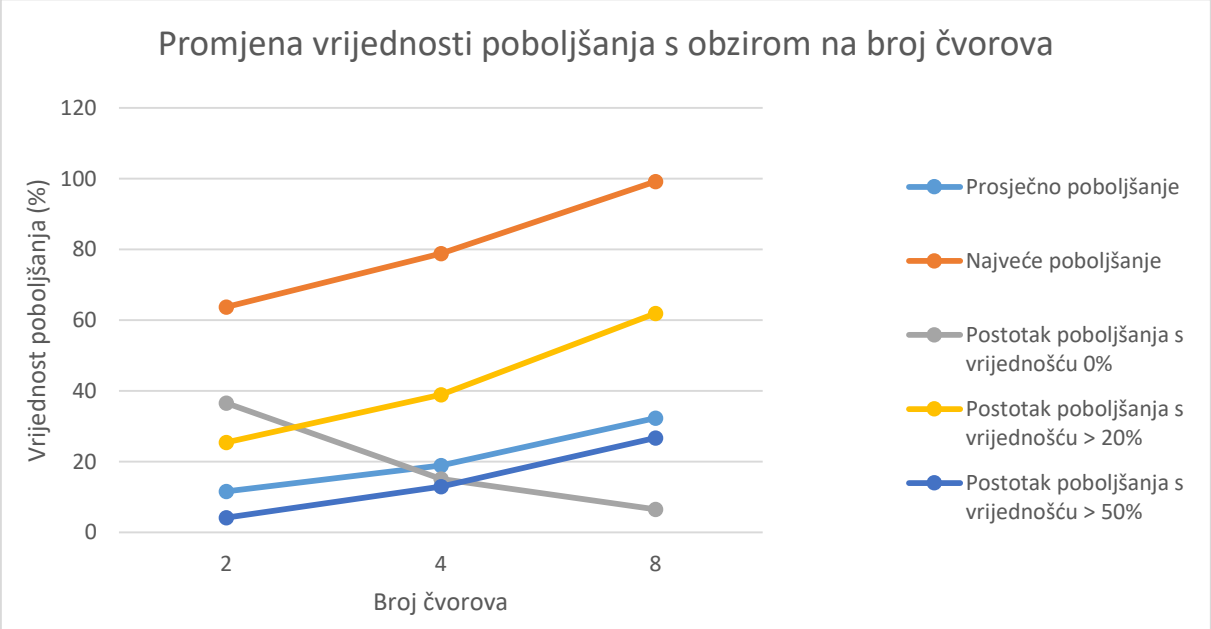
Obilježje / Kriterij	Rezultat	Postotak (%)
Ukupan broj zabilježenih korekcija	2552	-
Najmanja vrijednost poboljšanja površine korekcije	0%	-
Najveća vrijednost poboljšanja površine korekcije	99,1818%	-
Broj poboljšanja s vrijednošću 0%	167	6,54%
Broj poboljšanja s vrijednošću >20%	1500	61,91%
Broj poboljšanja s vrijednošću >50%	501	26,68%
Prosječni postotak poboljšanja	32,34%	

Može se reći da je simulacija i njezina analiza rezultata pokazala ono što se i očekivalo – na veći broj čvorova, algoritam postaje sve precizniji, odnosno dobivamo puno više boljih

korekcija, a puno manje ne-korigiranih pogreški. Kao što je pretpostavljeno, još uvijek se nađu čvorovi kojima algoritam ne uspijeva korigirati pogrešku (gdje je poboljšanje jednako 0%), a to objašnjavamo upravo onom činjenicom da postoje područja na definiranom terenu gdje je definirana GPS pogreška toliko mala da je nijedna kombinacija, odnosno presjeci drugih vijenaca ne može smanjiti. Međutim, takvih slučajeva je sada već jako malo, odnosno 167, od ukupnih 2552, tj. samo 6,54%. S druge strane, algoritam je uspio doći i do 99,1818% poboljšanja u jednom zapisu, što je vrlo dobro. Također, vidi se značajno poboljšanje i ostalih kriterija. Postotak korekcija s poboljšanjem većim od 20% narastao je na čak 61,91%, postotak korekcija s poboljšanjem iznad 50% iznosi 26,68%. Prosječno poboljšanje iznosi 32,34%.

S obzirom da je broj korekcija s vrijednošću 0 jako mali, i razlike u postotcima su jako male kada se računaju isti kriteriji, samo bez tih rezultata. Tako postotak korekcija većih od 20% iznosi 66,24%, a većih od 50% iznosi 28,55%. Prosječno poboljšanje bez korekcija s vrijednošću 0% iznosi 34,61%.

Na Slici 8. prikazan je graf koji prikazuje kretanje, odnosno promjene vrijednosti poboljšanja kroz navedene tri simulacije.



Slika 8. Graf promjena vrijednosti postotka poboljšanja

8. Budući rad

Bez obzira na rezultate dobivene ovom simulacijom, uvijek postoji mogućnost za napredak. Prema tome, već sada postoje ideje za daljnji razvoj navedenog algoritma za korekciju GPS pogreške čvora u crowd sourced sustavu.

U poglavlju 4. Algoritam za korekciju, algoritam je detaljno definiran te su objašnjeni njegovi koraci. Također, na kraju je spomenuta činjenica da čvor koji je najpreciznije određen, najviše i najbolje utječe na korekciju drugih čvorova, tj. njihovih pogreški, što je potvrdila i simulacija. Ta činjenica dovodi nas do logičnog nastavka, a to je ideja o više prolaza u istom trenutku, odnosno na istoj lokaciji. Dakle, nakon prvog prolaza, gdje se u obzir uzimaju samo GPS pogreške, u drugom prolazu bi se kao ulazni parametar uzimala korigirana pogreška, odnosno regija, i pomoću nje računala još preciznija lokacija, budući da bi i vijenci te regije trebali biti manji, odnosno uži. Osim toga, svakako se treba poraditi na optimizaciji algoritma, a možda uvođenju i višedretvenosti kako bi računalo moglo lakše i brže raditi i s većim brojem čvorova. Osim toga, algoritam se može i proširiti, odnosno može se napraviti verzija za one čvorove koji nemaju GPS (ili je iz nekog razloga njegova upotreba onemogućena), a u njihovom dometu postoje barem još dva čvora koji u WiFi dometu promatranog čvora imaju funkcionalan GPS prijemnik - u tom slučaju mogu se koristiti pravila trilateracije. Također, planira se korištenje metode *particle filtering*.

Osim navedene ideje za iduću verziju algoritma, svakako postoje i daljnje ideje za razvoj i napredak samog simulatora. Najprije tu je razvoj u UWP (*Universal Windows Platform*) tehnologiji, ne samo zbog jednostavnosti daljnjeg razvoja, već i zbog sve veće popularizacije aplikacija za Windows 8 i Windows 10 operacijske sustave. Mogućnosti za napredak su brojne: dodavanje mogućnosti spremanja i otvaranja postavki simulacije, prikaz čvorova na pravoj zemljopisnoj karti i unos zemljopisnih koordinata, prikaz rezultata i analiza istih unutar samog simulatora, i u konačnici povezivanje sa stvarnim čvorovima, odnosno bespilotnim letjelicama na terenu, odnosno opcija za nadzor kretanja povezanih letjelica.

9. Zaključak

Ovaj rad bavio se simulacijom pozicioniranja bespilotnih letjelica te razvojem algoritma za preciznije određivanje lokacije istih. Bespilotne (autonomne) letjelice su letjelice koje ne posjeduju posadu, već se upravljaju na daljinu, a često su opremljene mnogim senzorima poput kamera, toplinskih senzora, kamera, GPS prijemnika, itd.

Iznesena je hipoteza H1: Koristeći RF propagacijski model može se minimizirati GPS greška u crowd sourced sustavima. Nakon izrade algoritma, njegove implementacije u C# jeziku, odnosno Microsoftovoj .NET tehnologiji, testiranja i konačno provedenih simulacija te analiza rezultata istih, može se zaključiti da je hipoteza potvrđena jer je u najgorem mogućem slučaju (slučaj sa samo dva čvora), algoritam došao do iznosa površine regije pogreške koja je u prosjeku 11,55% manja od površine regije pogreške GPS sustava. S povećanjem broja čvorova, rezultati su se poboljšavali. Tako je najveći postotak poboljšanja korekcije sa 63,6475% u prvoj simulaciji (s dva čvora) porastao na čak 99,1818% u trećoj simulaciji (s osam čvorova). Postotak poboljšanja većih od 20% porastao je sa 25,43% na 61,91%, dok su one veće od 50% porasle s 4,17% na 26,68%. Prosječno poboljšanje korekcije povećalo se sa 11,55% na 32,34%.

Prema iznesenim rezultatima simulacija, može se zaključiti da je algoritam dobar, svakako postoji prostor za napredak: u obliku optimizacije te korištenja drugih tehnika i metoda kao što su trilateracija i particle filtering, u svrhu još bolje obrade podataka, odnosno još preciznijeg izračuna lokacije. U ovom obliku, algoritam se može koristiti u stvarnim sustavima jata letjelica.

10. Literatura

[1] Experts Exchange (2013) *Industry standard for minimum Wifi signal strength?* Preuzeto 20.08.2016. s https://www.experts-exchange.com/questions/28103112/Industry-standard-for-minimum-Wifi-signal-strength.html

[2] Fang X, Tang J, Xue G, Yang D (*s.a*) *Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing*. Preuzeto 01.09.2016. s http://surface.syr.edu/cgi/viewcontent.cgi?article=1237&context=eecs

[3] Frančula N (2015) Terminologija. Preuzeto 27.08.2016. s https://bib.irb.hr/datoteka/793247.Masovna_podrska.pdf

[4] GPS.GOV (2016) *Official U.S. Government information about the Global Positioning System (GPS) and related topics*. Preuzeto 18.08.2016. s http://www.gps.gov/

[5] Merriam Webster (2016) *Dictionary – Radio Frequency*. Preuzeto 20.08.2016. s http://www.merriam-webster.com/dictionary/radio%20frequency

[6] Sayler K (2015) *A World of Proliferated Drones: A Technology Primer*. Washington, DC: Center for a New American Security. Preuzeto 20.08.2016. s http://www.cnas.org/sites/default/files/publications-pdf/CNAS%20World%20of%20Drones_052115.pdf

[7] TheUAV.com (2016) *The UAV – Unmanned Aerial Vechile*. Preuzeto 18.08.2016. s http://www.theuav.com/index.html

[8] Tomaš B (2013) *WiFi roaming in urban multi-sensor environment.* Sveučilište u Zagrebu, Fakultet organizacije i informatike Varaždin.

[9] Villasensor J (2012) *What Is A Drone, Anyway?* Preuzeto 20.08.2016. s http://blogs.scientificamerican.com/guest-blog/what-is-a-drone-anyway/