Upravljanje bespilotnom letjelicom za pronalazak objekta na temelju semantike prostora

Gusić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 641

UPRAVLJANJE BESPILOTNOM LETJELICOM ZA PRONALAZAK OBJEKTA NA TEMELJU SEMANTIKE PROSTORA

Ivan Gusić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 641

UPRAVLJANJE BESPILOTNOM LETJELICOM ZA PRONALAZAK OBJEKTA NA TEMELJU SEMANTIKE PROSTORA

Ivan Gusić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 641

Pristupnik:	**Ivan Gusić (0036523209)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentorica:	izv. prof. dr. sc. Tamara Petrović

Zadatak: **Upravljanje bespilotnom letjelicom za pronalazak objekta na temelju semantike prostora**

Opis zadatka:

Potrebno je implementirati metodu upravljanja za bespilotnu letjelicu koja će omogućiti da letjelica u nepoznatom prostoru, u što kraćem vremenu, pronađe traženi objekt. Objekt koji je potrebno pronaći zadaje se opisno. Primjeri ulaznih specifikaciju su "plava knjiga" ili "lopta ispod stola". Letjelica je opremljena kamerom, a pretpostavka je da letjelica zna svoju poziciju (što će biti ostvareno pomoću Optitrack Trio sustava) i koristi gotov algoritam za planiranje i izvođenje trajektorija. Potrebno je implementirati metodu za prepoznavanje objekata u prostoru iz slike s kamere (npr. YOLO) te metodu za pretraživanje prostora na temelju prepoznatih objekata tj. semantike prostora. Metodu provjeriti u simulaciji te na stvarnom sustavu za različita okruženja. Usporediti učinkovitost razvijene metode s metodom koja ne uzima u obzir semantiku.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Bespilotna letjelica predstavlja letjelicu bez posade kojom se daljinski upravlja ili radi na različitim stupnjevima autonomije. Prve bespilotne letjelice konstruirane su za vrijeme Prvog svjetskog rata, kada su korištene za obuku protuzračne obrane i kao oružje. Od tada su letjelice prošle kroz značajan razvoj, postajući sofisticirani alati s mnogobrojnim primjenama.

Danas postoji mnogo područja primjene bespilotnih letjelica. Uz već spomenute vojne svrhe, koriste se i u agrikulturi, civilnoj zaštiti, snimanju filmova, industriji:

• U vojne svrhe, bespilotne letjelice koriste se za različite zadatke, uključujući izviđanje, nadzor, prikupljanje obavještajnih podataka i napade. Vojne bespilotne letjelice omogućavaju smanjenje rizika za ljudske živote, jer se mogu koristiti u opasnim misijama bez potrebe za posadom.

• U agrikulturi, bespilotne letjelice igraju ključnu ulogu u modernizaciji poljoprivrednih praksi. Koriste se za nadzor usjeva, identifikaciju bolesti, prskanje pesticida i gnojiva te analizu tla. Pomoću naprednih senzora i kamera, poljoprivrednici mogu dobiti detaljne informacije o stanju svojih usjeva.

• Bespilotne letjelice su postale nezamjenjivi alat za civilnu zaštitu i hitne službe. Koriste se za gašenje požara, nadzor i čuvanje granica, te za operacije pretrage i spašavanja. U slučajevima prirodnih katastrofa, kao što su potresi ili poplave, bespilotne letjelice omogućavaju brzu procjenu štete i identifikaciju preživjelih.

• Snimanje filmova i stvaranje medijskog sadržaja značajno su unaprijeđeni korištenjem bespilotnih letjelica. Bespilotne letjelice omogućavaju snimanje zapanjujućih zračnih snimaka koje bi inače bile nemoguće ili vrlo skupe za realizirati. Koriste se u filmskoj industriji, televizijskoj produkciji, novinarstvu i stvaranju marketinških materijala.

• Industrija također koristi bespilotne letjelice za razne zadatke, uključujući inspekciju infrastrukture, kao što su mostovi, dalekovodi, naftovodi i vjetroturbine. Bespilotne letjelice omogućavaju pregled teško dostupnih ili opasnih mjesta bez potrebe za skelama ili dizalicama, čime se povećava sigurnost i smanjuju troškovi inspekcije. U građevinskoj

industriji, bespilotne letjelice koriste se za praćenje napretka gradnje i izradu preciznih 3D modela gradilišta.

U svim navedenim primjerima poželjno je da letjelica ima implementiranu detekciju objekata kako se slika kasnije ne bi morala aučno obrađivati.

Bespilotne letjelice i dalje se većinom koriste u otvorenim prostorima, no kako letjelice postaju sve dostupnije i manje, njihova se primjena proširuje i na zatvorene prostore. Velike operacije pretrage vanjskih prostora moguće je preslikati i na unutarnje prostore, čime se otvaraju nove mogućnosti za primjenu bespilotnih letjelica u raznim situacijama i industrijama.

U ovome radu cilj je razviti upravljački sustav za bespilotnu letjelicu koji će omogućiti da se u nekom unaprijed zadanom prostoru pomoću letjelice, koja na sebi ima kameru, pronađe zadani objekt.

U prvom poglavlju opisan je problem koji će ovaj rad rješavati. Uz problem koji se rješava, opisano je što su početni resursi i koja su njihova ograničenja.

U drugom poglavlju opisan je odabir letjelice, ali i tehnologije korištene u radu kako bi se razvila metoda koja rješava zadani problem. U trećem poglavlju opisana je metoda koja je razvijena i koja se koristi u konačnom sustavu. Svaki dio metode opisan je u kontekstu svoje primjene unutar sustava, ali se može koristiti i kao zasebni modul koji rješava manje probleme.

U četvrtom poglavlju navedeni su rezultati konačnog sustava. Rezultati su uspoređeni s postojećim metodama pretrage kako bi se ocijenila prednost novog sustava.

1. Opis problema

U ovom radu predstavljen je sustav za autonomnu pretragu u zatvorenom prostoru. Letjelica se nalazi u malom prostoru te je cilj u što kraćem vremenu pronaći zadani predmet. Polazeći od pretpostavke da će se slični objekti nalaziti u blizini, iskoristiti će se semantika prostora za pronalazak predmeta. Primjerice, ako je poznata lokacija stola, a nije poznata lokacija pribora za jelo, letjelica će detaljnije skenirati prostor oko stola jer se može očekivati postavljen stol za jelo.

Semantička mapa ključna je komponenta sustava jer omogućuje letjelici da razumije i pamti položaje različitih objekata u prostoru. Na temelju ove mape, letjelica može planirati svoju putanju i optimizirati proces pretrage, smanjujući vrijeme potrebno za pronalaženje zadanog predmeta. Izlaz algoritma je lokacija predmeta u prostoru.

Sustav je predviđen za pretragu prostora manjih dimenzija poput manjeg stana ili uredskog prostora, gdje su resursi ograničeni. Letjelica koristi minimalne resurse, a opremljena je sustavom lokalizacije, kamerom u boji i senzorima udaljenosti za izbjegavanje prepreka. Ovi senzori omogućuju joj da se sigurno kreće kroz prostor, detektirajući i izbjegavajući prepreke, te identificira ciljani predmet.

Jedan od preduvjeta je *octomapa* prostora. Takvu mapu potrebno je prethodno izgraditi drugim robotom. Ona je ključna za navigaciju prostorom, ali i za estimaciju detektiranih objekata u prostoru.

2. Korištene tehnologije

U ovom poglavlju opisat će se tehnologije i letjelice korištene u radu kako bi se razvila metoda za autonomnu pretragu u zatvorenom prostoru. Prikazana je bespilotna letjelica Crazyflie. Također, opisane su tehnologije kao što su oktalno stablo za prostornu podjelu, YOLOv7 za detekciju objekata, Navigation2 za navigaciju, Clipper za geometrijske operacije te Word2Vec za semantičku analizu.

2.1. Bespilotna letjelica Crazyflie

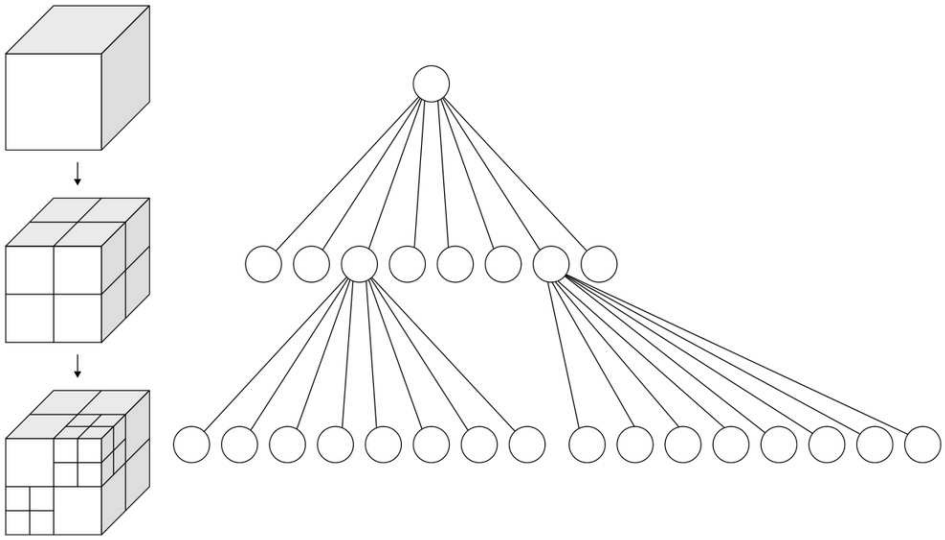
Crazyflie je kvadkopter (letjelica s 4 motora) razvijen od tvrtke Bitcraze. Ova bespilotna letjelica zamišljena je kao eksperimentalna platforma otvorenog koda i hardvera. Upravljanje letjelice moguće je ručno mobilnom aplikacijom te preko računala pomoću tipkovnice ili kontrolera. Upravljati letjelicom moguće je i autonomno uz CFlib API, ROS ili ROS2. Osnovna verzija letjelice ima svega 27 grama i manja je od ljudskog dlana, što omogućuje sigurno letenje i upravljanje u malim prostorima. Mala masa također smanjuje rizik od oštećenja u slučaju sudara, čineći Crazyflie sigurnim izborom za unutarnje letove i eksperimentalne projekte. Crazyflie je dizajniran s modularnošću na umu, omogućujući korisnicima dodavanje različitih modula koji proširuju funkcionalnosti letjelice. Među razvijenim modulima su kamera, senzori udaljenosti, LED svjetla i sustavi za praćenje, koji se jednostavno postavljaju na letjelicu. Ova modularnost omogućava korisnicima prilagodbu letjelice specifičnim potrebama njihovih projekata. Tako je u ovom radu na letjelicu montirana kamera koja služi za detekciju objekata. Isto tako montirani su i senzori udaljenosti kako bi letjelica sigurno letjela prostorom, obilazeći prepreke.

U ovom radu razvoj algoritma napravljen je u simulaciji Webots. Webots simulator je uvrstio Crazyflie letjelicu u svoju standardnu bazu robota. Simulator omogućuje razvoj i testiranje algoritama u kontroliranom virtualnom okruženju, smanjujući rizik od oštećenja stvarne letjelice i omogućujući bržu iteraciju razvoja.

2.2. Oktalno stablo

Oktalno stablo [2] (engl. octree) jest struktura podataka koja se koristi za prostornu podjelu trodimenzionalnih prostora. Ova su stabla posebno korisna u računalnoj grafici, robotici i simulacijama gdje je potrebno efikasno upravljati velikim količinama 3D podataka.

Osnovni princip oktalnih stabala je rekurzivna podjela prostora na manje dijelove [3]. Početni prostor, koji obuhvaća cijelu 3D scenu, predstavlja se kao korijenski čvor stabla. Taj prostor se zatim dijeli na osam manjih dijelova, od kojih svaki predstavlja podčvor korijena. Ova se podjela može nastaviti rekurzivno, pri čemu se svaki podčvor ponovno dijeli na osam manjih dijelova sve dok se ne postigne određena razina detalja ili drugi zadani kriterij zaustavljanja (Slika 2.1).



Slika 2.1 – Lijevo: Rekurzivna podjela prostora kocke. Desno: Odgovarajuće oktalno stablo [4]

Prostori koji su prazni ili homogeni mogu se sažeti u veće čvorove, smanjujući tako količinu potrebne memorije. To je posebno korisno kada se radi s velikim 3D scenama gdje nisu svi dijelovi scene jednako važni ili detaljni. Područja od interesa mogu se prikazati s više detalja, dok se manje važna područja prikazuju manje detaljno.

Struktura stabla omogućuje brzo pretraživanje prostora, što je korisno za zadatke kao što su sudaranje objekata, prepoznavanje objekata i planiranje putanja. Pretrage poput pronalaženja najbližih susjeda ili određivanja sudara mogu se obaviti mnogo brže jer se ne mora pretraživati cijela scena, već se pretraga ograničava na relevantne podčvorove.

U ovom radu oktalno stablo korišteno je za generiraanje karata za navigaciju i za estimaciju detektiranih objekata u prostoru.

2.3. YOLOv7

YOLOv7 [5] je jedna od inačica algoritma za detekciju objekata „You Only Look Once”. Visoka brzina obrade omogućuje letjelicama da u stvarnom vremenu detektiraju i prepoznaju objekte. Isto tako YOLO je učinkovit u prepoznavanju malih predmeta u slici, odnosno onih koji su daleko u prostoru u odnosu na letjelicu.

YOLOv7 detektor, kao i svi njegovi prethodnici, predtreniran je na skupu podataka MS COCO [6], što mu omogućuje prepoznavanje širokog spektra objekata, uključujući ljude, životinje, vozila, namještaj i svakodnevne predmete. Detaljan popis naučenih objekata za prepoznavanje moguće je pogledati na poveznici. Dodatne objekte koje bi YOLO mogao detektirati moguće je nadtrenirati na postojeći model.

U ovom radu YOLO je korišten za detekciju objekata u slici.

2.4. Navigation2

Navigation2 [7], ili skraćeno Nav2, najpoznatiji je ROS2 [8] paket za autonomnu navigaciju robota. Ovaj modularni sustav omogućava planiranje putanja, lokalizaciju i izbjegavanje prepreka u različitim okruženjima. Njegova skalabilnost omogućava podršku različitim tipovima robota.

Navigation2 temelji svoju navigaciju i planiranje putanja na karti tipa *OccupancyGrid*, koja se inicijalno učitava kao slika s postavkama definiranim u *.yaml* datoteci. Ova slika prikazuje stanje popunjenosti svake ćelije u prostoru koristeći boje piksela: bijeli pikseli označavaju slobodne ćelije, crni pikseli zauzete, dok nijanse sive predstavljaju nepoznata područja.

Već postoje kalibracijski parametri za navigaciju letjelice Crazyflie pomoću Navigation2 paketa [9], stoga je Navigation2 paket odabran za 2D navigaciju letjelice. Potrebe za 3D navigacijom u ovom radu nije bilo jer je sav let moguće odraditi na konstantnoj visini iznad namještaja. Ograničavanjem letjelice na konstantnu visinu smanjuje se složenost navigacijskog algoritma. Letjelica se može kretati iznad namještaja i drugih prepreka, zadržavajući konstantnu visinu, što pojednostavljuje proces pretrage i smanjuje rizik od sudara. Ukoliko je potrebna promjena visine, tada se letjelica pomakne na željenu visinu (ako ondje nema prepreke) te se učita nova karta za novu visinu.

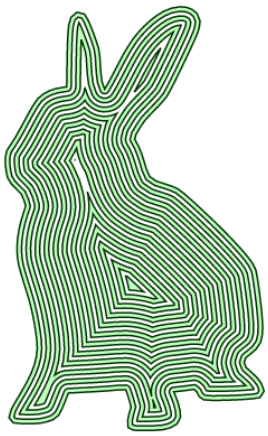
2.5. Clipper

Clipper [10] je besplatna biblioteka otvorenog koda (napisana u C++, C# i Delphi Pascalu) za rad s linijama i poligonima. Clipper omogućava precizne geometrijske operacije kao što su:

- unija (engl. union) – spajanje dvaju ili više poligona u jedan poligon
- razlika (engl. difference) – oduzimanje jednog poligona od drugog
- presjek (engl. intersection) – pronalaženje zajedničkog dijela dvaju poligona
- ekskluzivni ILI (engl. exlusive OR) – kombinacija dijelova poligona koji se ne preklapaju

Isto tako biblioteka Clipper nudi mogućnost *offsettinga* (engl. offset – pomak) koji omogućuje proširivanje ili smanjivanje poligona kao što je u primjeru na Slika 2.2.

Clipper je u ovome radu korišten za računanje presjeka prepreka i generiranih prostora za pretragu, kao i za generiranje *lawnmower* putanje.



Slika 2.2 – Offseting [10]

Zbog svoje učinkovitosti i preciznosti, često se koristi u aplikacijama koje zahtijevaju složene geometrijske izračune. S obzirom da je kod za ovaj rad pisan u Pythonu, korišten je Python omotač naziva Pyclipper [11].

2.6. Word2Vec

Word2Vec [12] je popularna tehnika u obradi prirodnog jezika (NLP) koja pretvara riječi u numeričke vektore, omogućujući strojevima da razumiju kontekstualno značenje riječi.

Radi tako što analizira velike količine teksta i bilježi kontekst u kojem se riječi pojavljuju, koristeći dvije glavne metode: *Continuous Bag of Words* (CBOW) i *Skip-gram* [13]. CBOW predviđa ciljanu riječ na temelju okruženja, dok Skip-gram predviđa okruženje na temelju ciljne riječi. Rezultat su vektori u višedimenzionalnom prostoru gdje slične riječi imaju slične vektorske reprezentacije.

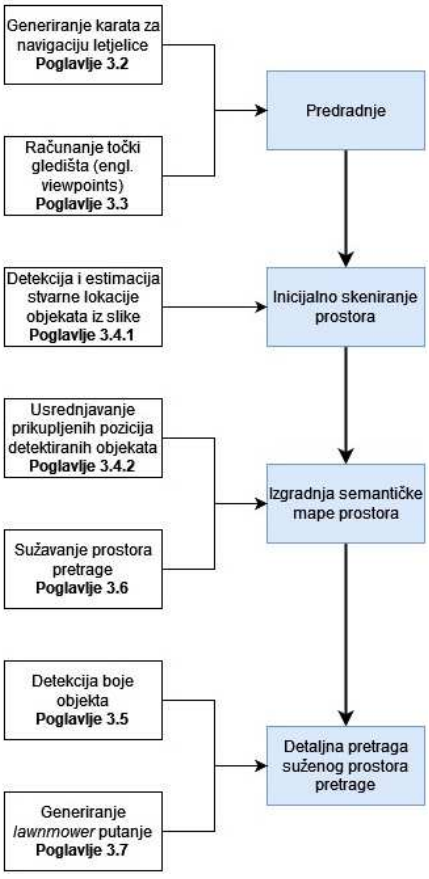
Model za određivanje mjere sličnosti treniran je na skupu podataka Google News [15] koji ima oko 3 milijuna riječi i fraza. U ovom radu Word2Vec korišten je za semantiku prostora, odnosno za određivanje koliko su detektirani objekti slični traženome. Primjer dvije mjere sličnosti:

- *žlica* (engl. spoon) i *hladnjak* (engl. fridge) – mjera sličnosti iznosi 0,4114
- *žlica* (engl.spoon) i *sat (engl.* clock) – mjera sličnosti iznosi 0,1654

3. Metoda za pretraživanje

3.1. Pregled metode

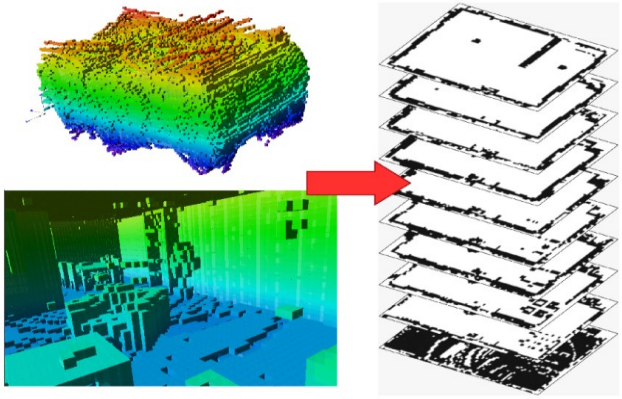
Prije samog leta, potrebno je napraviti nekoliko predradnji. Kao prvi korak, potrebno je generirati 2D kartu prostora iz oktalne mape prostora kako bi se letjelicom moglo upravljati Nav2 paketom. Druga predradnja jest računanje jednog ili više gledišta (engl. *viewpoints*). Gledište se određuje kao lokacija u prostoru koja je značajno udaljena od prepreka. Takva lokacija pogodna je za mjesto gdje se letjelica okrene za 360° i snimi veliku površinu prostora. Za vrijeme skeniranja prostora u točki gledišta, letjelica pomoću kamere detektira objekte te estimira njihove pozicije u prostoru. Postupak se ponavlja dok letjelica ne obiđe sva izračunata gledišta. Idući korak jest filtriranje detektiranih objekata te slanje letjelice na lokacije koje sadrže slične predmete onomu koji se traži. Slika 3.1 prikazuje dijagram razvijene metode.



Slika 3.1 – Pregled metode

3.2. Generiranje karata za navigaciju letjelice

S obzirom na to da se Nav2 paket oslanja na 2D navigaciju unutar 3D okruženja, iz trodimenzionalnog oktalnog stabla, čiji su listovi veličine 10x10x10 cm, izrađene su horizontalne mape prostora za svakih 10 cm visine (Slika 3.2)



Slika 3.2 – Generiranje 2D karata za navigaciju iz oktalnog stabla

Proces generiranja *OccupancyGrida* iz *Octomape* počinje tako da se sortiraju listovi oktalnog stabla po visini, odnosno po z osi. Zatim se za svaku različitu vrijednost z osi generira bijela slika koja predstavlja prazan prostor. Preslikavanjem koordinata listova oktalnog stabla na sliku (postavljanje crnih piksela u slici) generira se mapa prostora za svaku različitu vrijednost z osi. Pseudokod algoritma dan je u Kod 3.1.

```
for each z_level in octree:
    sliced_octree = get_sliced_octree(octree, z_level)
    points = []
    for each node in sliced_octree.begin_leafs():
          points.append([node.y, node.x])
    grid = create_empty_grid(min_point, max_point)
    for each point in points:
          grid_coordinates = convert_to_grid_coordinates(point)
          grid[grid_coordinates] = 1
          save_to_map(grid, z_level)
```

Kod 3.1 – Pseudokod generiranja karata iz oktalnog stabla

Ove horizontalne karte omogućuju planiranje putanja i navigaciju robota unutar složenih trodimenzionalnih okruženja, pružajući jasnu sliku o slobodnim i zauzetim područjima.

3.3. Računanje točki gledišta za inicijalno skeniranje prostora

Prilikom uzleta letjelice, ideja je snimiti prostor na način koji omogućava letjelici da uhvati što više prostora i detektira što više objekata. Inicijalno skeniranje prostora odvija se tako da letjelica dođe na određenu poziciju (točku gledišta) i okrene se za 360° te to ponovi nekoliko puta, svaki put na drugoj unaprijed izračunatoj poziciji. Te pozicije računaju se tako da predstavljaju najudaljenija mjesta od prepreka.

```
Koraci algoritma za određivanje gledišta prikazani su u
nastavku, a algoritam je prikazan i slikovno
```

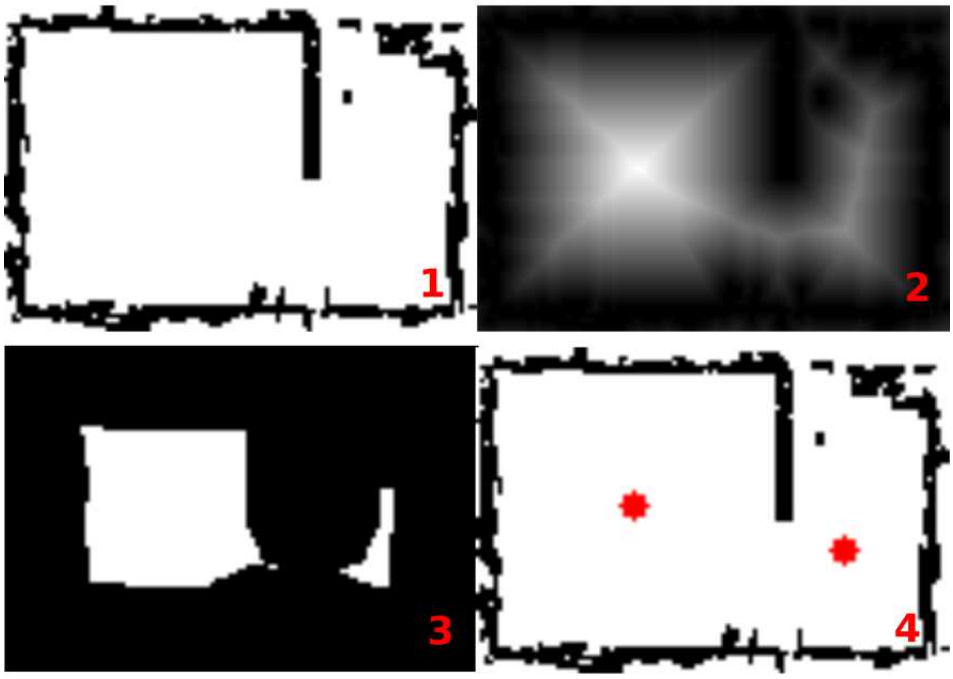
Kod 3.2 i Slika 3.3):

1. Za inicijalnu visinu uzleta letjelice dohvaća se dvodimenzionalna mapa prostora.
2. Pomoću funkcije `distance_transform` za svaki piksel u prostoru koji je različit od 0 (koji ne predstavlja prepreku) računa euklidsku udaljenost od najbližeg piksela koji je 0 (prepreka). Zatim se udaljenosna mapa normalizira u rasponu od 0 do 1 koristeći funkciju `normalize`. Normalizirana slika se pretvara u binarnu sliku, gdje su svi pikseli s vrijednostima većim od zadanog praga (engl. `threshold`) postavljeni na 255, a ostali na 0.
3. Pomoću funkcije `find_contours` pronalaze se konture (rubovi) na binarnoj slici. Konture predstavljaju područja najudaljenija od prepreka. Za svaku konturu izračunavaju se momenti (pomoću funkcije `calculate_moments`) kako bi se našle središnje točke (`centers`) tih područja, a to su ujedno i najudaljenije točke od prepreka.
4. Na kraju se te točke prebace u koordinatni sustav letjelice na koje se zatim letjelica i šalje.

```
map_image = load_map(robot_position.z)
dist_transform = distance_transform(map_image)
adjusted_image = normalize(dist_transform)
binary_image = threshold_image(adjusted_image, threshold)
contours = find_contours(binary_image)
centers = []
for each contour in contours:
moments = calculate_moments(contour)
```

```
if contour area is non-zero:
    center = calcluate_center(moments)
    scaled_center = img_to_real_coordinate(center)
    centers.add(scaled_center)
```

Kod 3.2 – Pseudokod generiranja viewpointa



Slika 3.3 – Algoritam generiranja gledišta

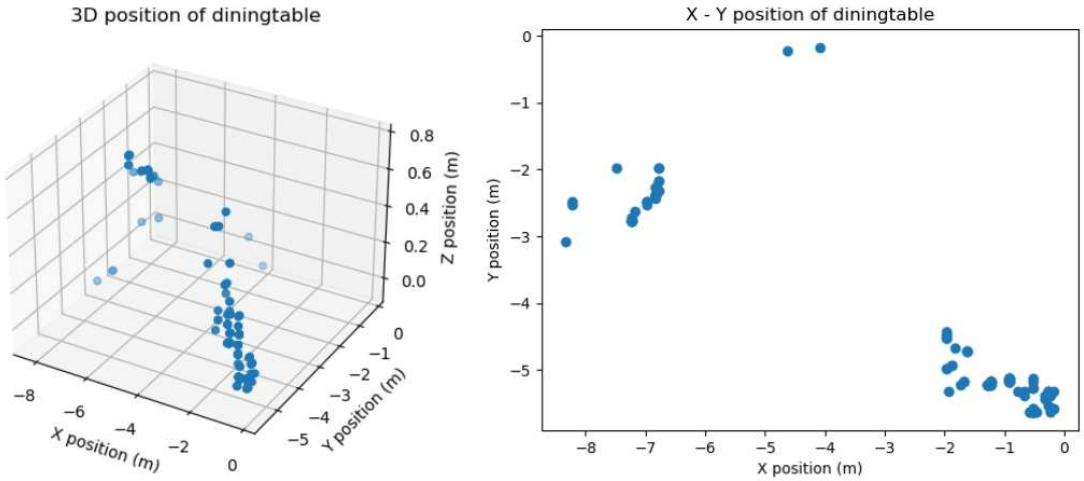
Cijelo vrijeme tijekom letenja letjelica šalje podatke iz kamere te se u pozadini pomoću YOLOv7 detektora detektiraju objekti u prostoru. Kada YOLOv7 detektor prepozna objekt, estimira se i pohrani njegova pozicija u prostoru. Ako se traženi objekt pronađe, algoritam se zaustavlja te nema daljnje pretrage.

3.4. Estimacija lokacije objekta u prostoru

Estimacija lokacije objekata odvija se u dvije faze. U inicijalnom skeniranju prostora, za svaku dohvaćenu sliku s kamere, detektiraju se objekti primjenom Yolo algoritma. Ovim postupkom dobit će se onoliko pozicija istog predmeta koliko je puta bio detektiran u slikama. U estimiranim pozicijama istog objekta postojat će oscilacije. Nakon završetka inicijalnog skeniranja prostora, sve zapisane lokacije o detektiranom objektu sjedinit će se u jednu postupkom usrednjavanja.

3.4.1. Računanje stvarne pozicije detektiranih objekata u slici

Da bi se odredila pozicija detektiranog objekta, potrebno je izvršiti nekoliko koraka. Prvo, detektor na slici daje informacije o objektu: ID objekta, sigurnost detekcije i okvir (engl. *bounding box*) koji opisuje poziciju objekta unutar slike. Zatim se koristeći ove informacije može procijeniti položaj objekta u 3D prostoru. Na Slika 3.4 vidi se primjer estimacije pozicije objekta tipa *blagovaonski stol* (engl. *dining table*) u prostoru. S obzirom da je blagovaonski stol jedini tip stola za koji je YOLO model naučen, on će sve tipove stolova detektirati kao blagovaonske stolove. Na dvodimenzionalnom prikazu može se uočiti kako točke odstupaju (detaljno analizirano u Poglavlju 3.4.3), ali da su i grupirane na tri lokacije jer se u prostoru zaista nalaze tri stola – radni stol, stolić u dnevnom boravku, blagovaonski stol.



Slika 3.4 – Trodimenzionalna i dvodimenzionalna reprezentacija estimiranih lokacija

U nastavku je opisan način estimacije pozicije objekta. Metoda `cast_ray` prima oktalnu mapu prostora `octomap`, ishodište zrake `camera_pose` te smjer zrake `direction`, a vraća prvi okupirani voksel `end` kroz koji ta zraka prolazi. Kao ishodište zrake postavlja se pozicija kamere, a smjer zrake izračunava se kao zbroj orijentacije kamere i detektiranog objekta unutar slike i odnosa centra slike (Kod 3.3).

```
classes, boxes, scores = detect(image)
    object_estimation_dict = {}
    for each cls, box, score in zip(classes, scores):
        diff_pitch, diff_yaw = calc_img_center_dev(box, fov)
        total_yaw_diff = robot_yaw - diff_yaw
        total_pitch = robot_pitch - diff_pitch
```

```
dir_x = cos(total_yaw)
dir_y = sin(total_yaw)
dir_z = -sin(total_pitch / 2)
direction = noramlize(dir_x, dir_y, dir_z)
hit, end = cast_ray(octomap, camera_position, direction)
if hit:
    object_estimation_dict[cls].add(end)
```

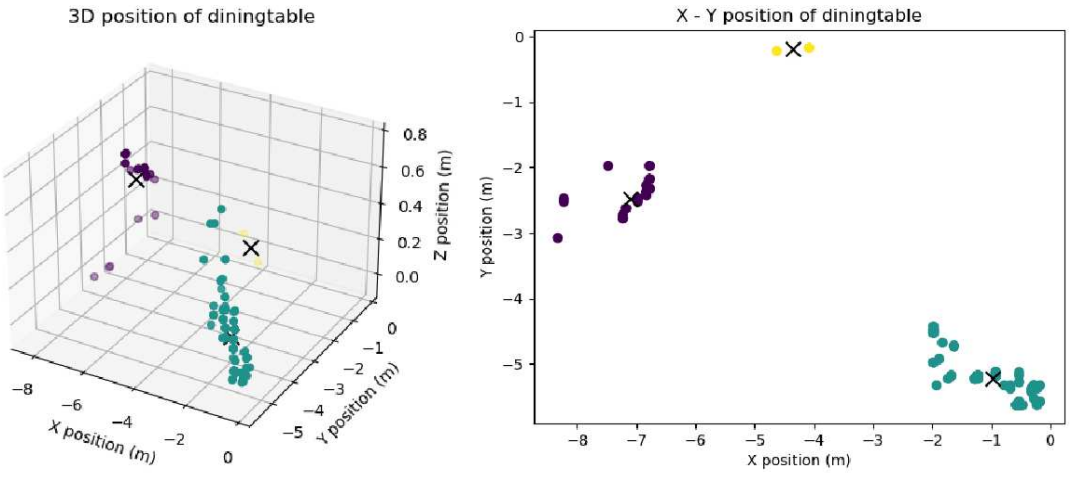
Kod 3.3 – Psedokod estimacije detektiranih objekata u slici

3.4.2. Usrednjavanje prikupljenih pozicija detektiranih objekata

S obzirom da je moguće detektirati nekoliko istih objekata, ali na različitim lokacijama (npr. stol u radnoj sobi i stol u blagovaonici), potrebno je prvo zapisane podatke o objektu grupirati. U ovom radu za to je korišteno DBSCAN grupiranje [14] (Slika 3.5). Zatim se za svaku grupu lokacija računa srednja vrijednost. Tako se nekoliko stotina zapisanih lokacija svede na nekoliko lokacija od kojih svaka predstavlja srednju estimiranu vrijednost lokacije za određeni objekt u prostoru. Pseudokod algoritma moguće je vidjeti u danom Kod 3.4.

```
avg_detected_dict = {}
for each object in object_estimation_dict:
    clusters = calculate_clusters(object_estimation_dict[object])
        for each cluster in clusters:
            avg_detected_dict[object].add(calculate_mean(cluster))
```

Kod 3.4 – Pseudokod grupiranja i usrednjavanja pozicija detektiranih objekata



Slika 3.5 – Grafički prikaz grupiranih podataka

Slika 3.6 prikazuje prostor u kojem su detektirani objekti *blagovaonski stol*. Stvarne pozicije centara stolova su (-1.07, -4.94, 0), (-5.13, -0.52, 0) i (-7.16, -2.56, 0.02) što približno odgovara i estimiranim točkama. Detaljna analiza preciznosti estimacije obrađena je u Poglavlju 3.4.3.

Slika 3.6 – Prikaz stvarnih pozicija stolova (unutar zelenih kružnica)

3.4.3. Analiza

U ovom poglavlju detaljnije će se analizirati preciznost estimacije lokacije detektiranih objekata. Za testiranje odabrana su tri predmeta u prostoru: sat na zidu, vaza s cvijećem i boca koja se nalazi na stolu. Analiza je provedena u simulatoru tako da je letjelica kružila oko objekata i iz različitih kutova detektirala objekte. Analizira se udaljenost estimirane lokacije od centra objekta.

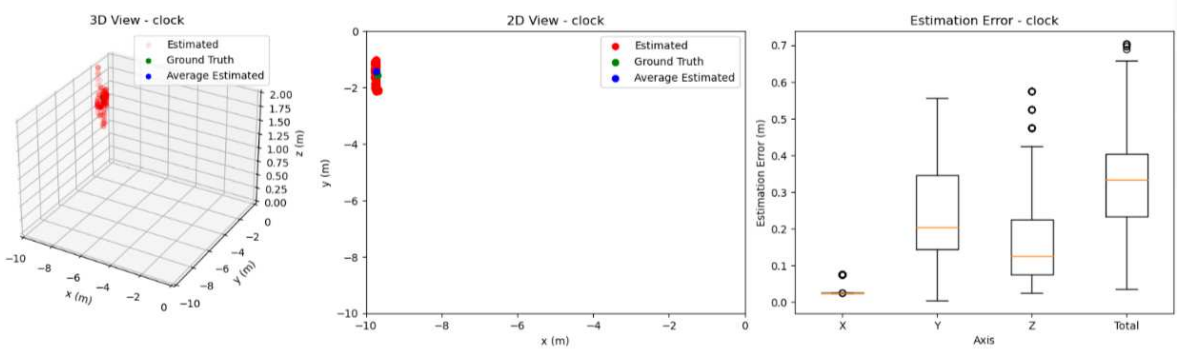
Analiza rezultata za *sat* (Tablica 3.1 i Slika 3.7):

- **Broj mjerenja**: 1107
- **Min vrijednosti**: Najmanje pogreške u x, y i z osi su 0,02 m, 0,01 m i 0,03 m, redom. Ove vrijednosti predstavljaju najbliže detekcije stvarnim pozicijama sata.

- **Max vrijednosti**: Najveće pogreške su znatno veće, posebno u y i z osi (0,56 m i 0,58 m), što ukazuje na potencijalne probleme s detekcijom iz određenih kutova ili udaljenosti.
- **Srednja vrijednost**: Prosječne pogreške za x, y i z osi su 0,03 m, 0,25 m i 0,16 m, dok je ukupna prosječna pogreška 0,33 m. To implicira da su detekcije u prosjeku bile relativno točne.
- **Standardna devijacija (σ)**: Vrijednosti standardne devijacije su 0,01 m za x os, 0,13 m za y os, 0,12 m za z os, i 0,13 m ukupno, što ukazuje na malu varijabilnost mjerenja. Najveća varijabilnost je u y i z osi, dok je x os relativno stabilna.

Tablica 3.1 – Sat

	x os (m)	y os (m)	z os (m)	3D (m)
min	0,02	0,01	0,03	0,04
max	0,07	0,56	0,58	0,70
Srednja vrijednost	0,03	0,25	0,16	0,33
σ	0,01	0,13	0,12	0,13



Slika 3.7 – 3D i 2D prikaz rezultata, grafički prikaz pogreške mjerenja – Sat

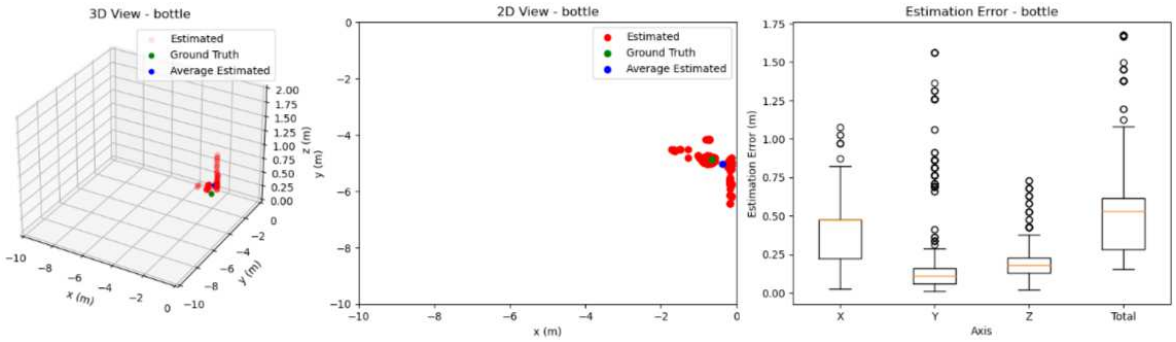
Analiza rezultata za *bocu*:

- **Broj mjerenja**: 869

- **Min vrijednosti**: Najmanje pogreške u x, y i z osi su 0,02 m, 0,01 m i 0,02 m, respektivno. Ove vrijednosti predstavljaju najbliže detekcije stvarnim pozicijama boce.
- **Max vrijednosti**: Najveće pogreške su znatno veće, posebno u y osi (1,56 m) i ukupnoj pogrešci (1,67 m). Ovi rezultati ukazuju na velike varijacije u preciznosti mjerenja.
- **Srednja vrijednost**: Prosječne pogreške za x, y i z osi su 0,04 m, 0,23 m i 0,18 m, dok je ukupna 3D pogreška 0,55 m. Prosječna pogreška je veća nego kod sata, što sugerira da je bocu teže detektirati s visokom preciznošću.
- **Standardna devijacija (σ)**: Vrijednosti standardne devijacije su 0,17 m za x os, 0,30 m za y os, 0,11 m za z os, i 0,27 m kada se gleda ukupno za sve osi. Ove vrijednosti pokazuju značajnu varijabilnost mjerenja, posebno u y osi.

Tablica 3.2 – Boca

	x os (m)	y os (m)	z os (m)	3D (m)
min	0,02	0,01	0,02	0,15
max	1,07	1,56	0,72	1,67
Srednja vrijednost	0,04	0,23	0,18	0,55
σ	0,17	0,30	0,11	0,27



Slika 3.8 – 3D i 2D prikaz rezultata, grafički prikaz pogreške mjerenja – Boca

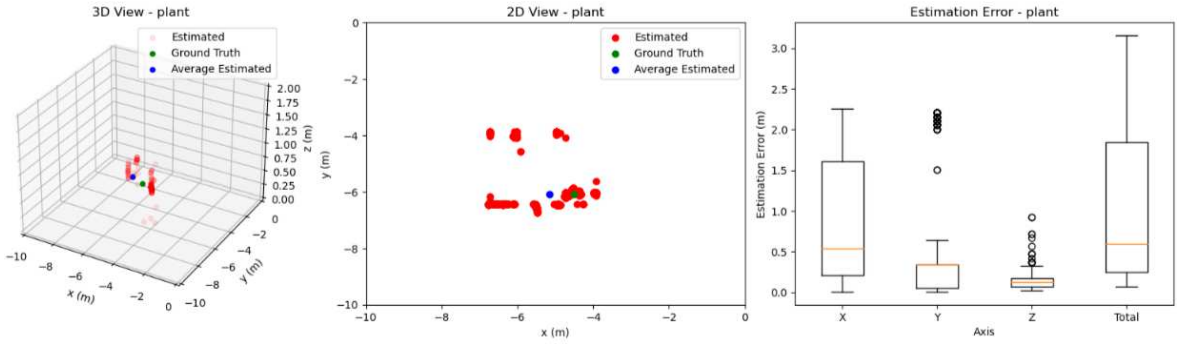
Analiza rezultata za *vazu s cvijećem* (Tablica 3.2 i Slika 3.8):

- **Broj mjerenja**: 998

- **Min vrijednosti**: Najmanje pogreške u x, y i z osi su 0,01 m, 0,01 m i 0,02 m, respektivno. Ove vrijednosti predstavljaju najbliže detekcije stvarnim pozicijama vaze.
- **Max vrijednosti**: Najveće pogreške su vrlo velike, posebno u x i y osi (2,25 m i 2,20 m) te ukupnoj pogrešci (3,15 m). Ovi rezultati ukazuju na vrlo velike varijacije u preciznosti mjerenja.
- **Srednja vrijednost**: Prosječne pogreške za x, y i z osi su 0,84 m, 0,36 m i 0,14 m, dok je ukupna srednja pogreška 0,99 m. Prosječne pogreške su veće nego kod sata i boce, što sugerira da je vazu bilo najteže detektirati s visokom preciznošću.
- **Standardna devijacija (σ)**: Vrijednosti standardne devijacije su 0,75 m za x os, 0,50 m za y os, 0,09 m za z os, i ukupna iznosi 0,83 m. Ove vrijednosti pokazuju izuzetno veliku varijabilnost mjerenja, posebno u x i y osi.

Tablica 3.3 – Vaza s cvijećem

	x os (m)	y os (m)	z os (m)	3D (m)
min	0,01	0,01	0,02	0,07
max	2,25	2,20	0,92	3,15
Srednja vrijednost	0,84	0,36	0,14	0,99
σ	0,75	0,50	0,09	0,83



Slika 3.9 – 3D i 2D prikaz rezultata, grafički prikaz pogreške mjerenja – Vaza s cvijećem

Greške u mjerenjima nastaju zbog nekoliko faktora. Za estimaciju pozicije predmeta u prostoru, uz sliku, potrebna je pozicija letjelice. Pozicija letjelice (koja je isto estimirana)

dohvaća se nevezano uz sliku. Budući da u većini slučaja pozicija letjelice također ima pogrešku, to utječe na pogrešku estimacije objekata. Isto tako, za predmete koji se ne nalaze uz veliku prepreku (poput *sata*) moguće je da lako „promaknu“ i da se detektira veća prepreka koja se nalazi iza predmeta. Isto tako u ovoj analizi uzeta je pogreška od središta predmeta, pa će predmeti većih dimenzija očekivano imati i veću pogrešku u slučajevima kada se detektira rub predmeta (Slika 3.10).

Slika 3.10 – Loše detektirana stolica

3.5. Detekcija boje objekta

Nakon što se objekt detektira, slika se odreže tako da ostane samo slika unutar okvira detektiranog objekta (engl. *bounding box*) objekta. Okvir detektiranog objekta je pravokutnik koji u potpunosti obuhvaća predmet na slici. Izrezivanjem slike unutar okvira detektiranog objekta nastaje nova slika koja prikazuje samo predmet bez okolnog sadržaja.

Nakon izrezivanja slike, generira se histogram boja iz dobivene slike. Histogram boja predstavlja distribuciju boja unutar slike, prikazujući koliko je pojedinih boja prisutno. Analizom histograma možemo odrediti koje su boje najzastupljenije u slici.

Na kraju, najučestalija boja iz histograma odrezane slike koristi se za definiranje boje predmeta. To znači da će boja koja se pojavljuje najviše puta u odrezanoj slici biti određena kao glavna boja objekta.

U Slika 3.11 vidljiv je detektiran objekt tipa *hladnjak* (engl. *refrigerator*) i njegov okvir. Najučestalija boja unutar okvira je crvena, stoga će algoritam za ovaj objekt definirati boju kao *crvena*.

Slika 3.11 – Detekcija boje

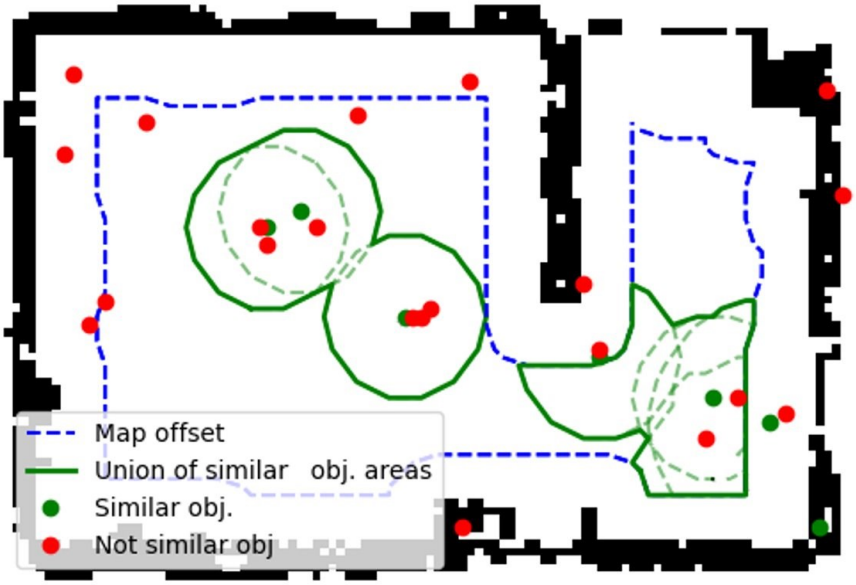
3.6. Sužavanje prostora pretrage

Nakon inicijalnog skeniranja prostora letjelicom (Poglavlje **Error! Reference source not found.**), prikupljene lokacije predmeta se usrednjavaju (Poglavlje 3.4.2) i filtriraju. Filtriranje se provodi tako da se zadrže samo objekti koji su slični traženom objektu. Svaki se detektirani objekt vektorski uspoređuje s traženim objektom. Ako je mjera sličnosti veća od zadanog praga (engl. *threshold*), tada se oko tog objekta izgradi prostor pretrage. Slika 3.12 prikazuje primjer u kojem je vidljivo kako se prvo računaju predmeti slični traženom (na slici prikazani zelenim točkama), dok se crvene točke zanemaruju. Oko predmeta sličnih traženom grade se područja za pretragu (zelena isprekidana crta) koja se presijecaju preprekama (plava isprekidana linija) kako se letjelica ne bi sudarila i snimala preblizu zida. Nakon toga se definiraju unije područja za pretragu (puna zelena linija) koje predstavljaju sužena područja za detaljnu pretragu. Pseudokod algoritma dan je u Kod 3.5.

Kod 3.5 – Pseudokod generiranja suženog prostora pretrage

```
area_list = []
for each object in avg_detected_dict:
    if vec_similarity(object, searched_object) > threshold:
```

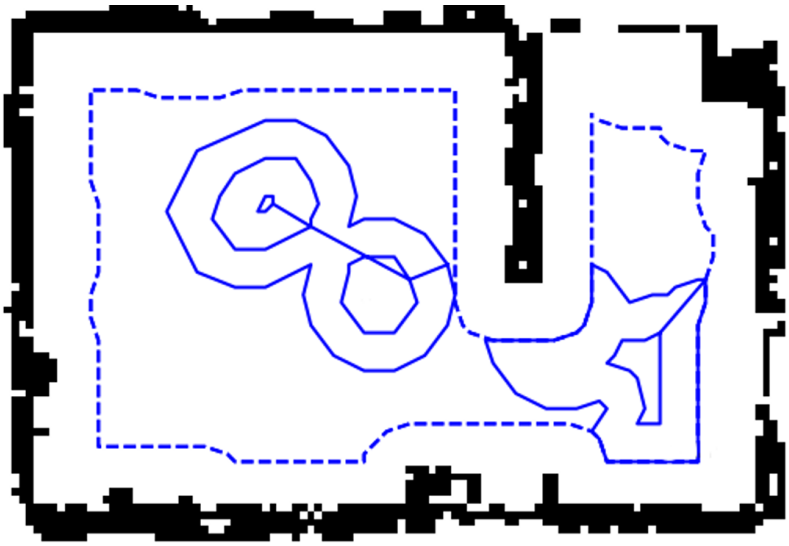
```
            obj_area = create_obj_area(object, map)
            area_list.append(obj_area)
create_unions(area_list)
```



Slika 3.12 – Generiranje suženog prostora pretrage

3.7. Generiranje *lawnmower* putanje

Sljedeći korak je generiranje *lawnmower* putanje unutar suženog prostora pretrage. Ovaj tip putanje osigurava da letjelica sistematski pokrije cijelo područje. Paket *pyclipper* omogućuje računanje novog polinoma na temelju zadanog uz zadani razmak. Tako se za početni polinom koristi polinom suženog prostora, a zatim se generiraju novi polinomi unutar početnog uz definirani razmak. Koordinate vrhova svih generiranih polinoma zapisuju se u listu. Slika 3.13 prikazuje generirane putanje (plava puna linija). Lista vrhova generiranih polinoma se prosljeđuje navigacijskom sustavu letjelice. Navigacija koristi ovu listu kako bi letjelicu vodila od jedne do druge točke i tako detaljno pretražila prostor.



Slika 3.13 – *Lawnmower* putanje

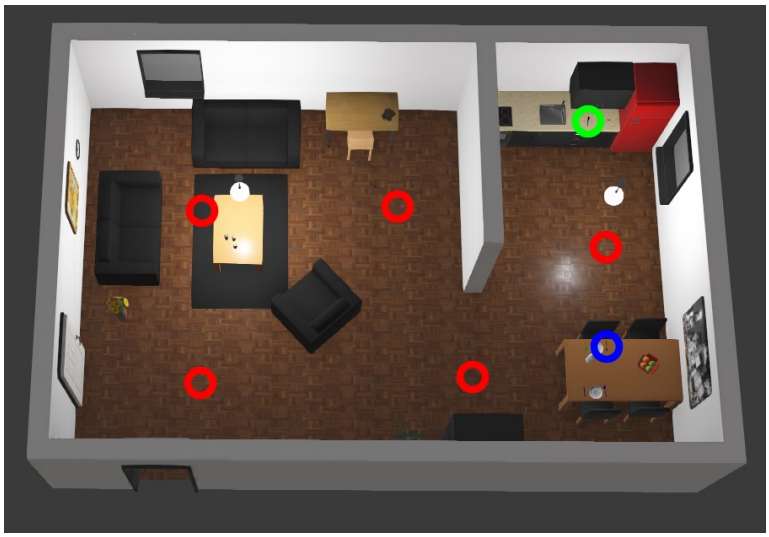
4. Rezultati

Testiranje razvijene metode provedeno je u simulatoru Webots koji ima u svojoj biblioteci letjelicu Crazyflie kao standardnog robota [17]. Razvijena metoda pretrage uspoređivana je s klasičnom *lawnmower* metodom, koja je poznata po svojoj jednostavnosti i učinkovitosti u pokrivanju velikih površina. Cilj usporedbe bio je evaluirati performanse nove metode u odnosu na standardne pristupe te utvrditi njene prednosti i nedostatke u različitim scenarijima pretrage. Izgled letjelice u simulatoru vidljiv je na Slika 4.1.

Slika 4.1 – letjelica Crazyflie

Pribor za jelo, odnosno vilica, žlica i nož, najmanji su predtrenirani objekti za raspoznavanje u detektoru YOLOv7. Stoga je žlica odabrana kao objekt pretrage kako bi zadatak bio što kompleksniji. Raspoznavanje manjih objekata predstavlja veći izazov za sustav pretrage, čime se dodatno testira preciznost i učinkovitost razvijene metode.

U prva dva testiranja korišten je svijet iz standardne biblioteke Webots simulatora [18]. U Slika 4.2 prikazan je izgled apartmana, zajedno s početnim pozicijama letjelice (označene crvenim kružnicama) i pozicijama objekta koji se pretražuje. U **prvom testiranju**, žlica se nalazila na kuhinjskom elementu između sudopera i hladnjaka (označeno zelenom kružnicom), dok je u **drugom testiranju** bila postavljena na blagovaonskom stolu. Pozicije objekta koji se pretražuje označene su zelenom kružnicom za prvo testiranje i plavom kružnicom za drugo testiranje.



Slika 4.2 – Prostor pretrage za 1. i 2. testiranje

Tablica 4.1 prikazuje rezultate prvog testiranja. Prosječno vrijeme za razvijenu metodu iznosi 2 minute i 55 sekundi dok prosječno vrijeme za klasičnu pretragu iznosi 4 minute i 26 sekundi. Razlika iznosi 1 minutu i 31 sekundu u korist razvijene metode. Zamijećeno je da razvijena metoda daje puno bolje rezultate kada je traženi objekt daleko, jer letjelica preskače pretragu dijelova sobe navođena semantikom prostora.

Tablica 4.1 – Rezultati prvog testiranja

Redni broj mjerenja	Razvijena metoda (*min:sek)*	Klasična pretraga (*min:sek*)
1.	3:03	5:17
2.	1:33	4:38
3.	2:08	6:38
4.	4:21	3:25
5.	3:28	2:13

Tablica 4.2 prikazuje rezultate drugog testiranja. U ovom primjeru traženi objekt uspješno se detektira prilikom inicijalnog skeniranja prostora osim u četvrtom mjerenju. Prosječno vrijeme razvijene metode iznosi 1 minutu i 18 sekundi, dok je za klasičnu pretragu prosječno bilo potrebno 3 minute i 9 sekundi. Ovdje se vidi znatni napredak u slučaju kada je predmet vidljiv već u postupku inicijalnog skeniranja prostora.

Tablica 4.2 – Rezultati drugog testiranja

Redni broj mjerenja	Razvijena metoda (*min:sek)*	Klasična pretraga (*min:sek*)
1.	0:54	4:43
2.	1:05	3:38
3.	1:17	5:11
4.	2:31	1:13
5.	0:45	0:59

U **trećem testiranju** korišten je drugi svijet koji predstavlja uredski prostor [19]. Slika 4.3 prikazuje prostor u kojem su dosta jednoliko raspoređeni uredski stolovi, stolice i računala. Na sredini se nalazi jedna čajna kuhinja koja ima sudoper. Žlica je postavljena na radnu ploču čajne kuhinje. Očekuje se da će detekcija objekta *sudoper* (engl. *sink*) i poznavanje njegove lokacije pomoći u potrazi za puno manjim predmetom – *žlicom* (engl. *spoon*).

Slika 4.3 – Prostor pretrage za treće testiranje

Tablica 4.3 prikazuje rezultate trećeg testiranja. Prosječno vrijeme razvijene metode iznosi 4 minute i 16 sekundi dok je za klasičnu pretragu prosječno bilo potrebno 5 minuta i 49 sekundi. Razlika u vremenu pretrage ove dvije metode je 1 minuta i 33 sekunde.

Tablica 4.3 - Rezultati trećeg testiranja

Redni broj mjerenja	Razvijena metoda (min:sek)	Klasična pretraga (min:sek)
1.	4:46	6:29
2.	4:23	7:04
3.	2:01	3:15
4.	5:33	5:41
5.	4:36	6:38

Zaključak

U ovome radu razvijen je sustav pretrage objekata na temelju semantike prostora. U početku pretrage, letjelicom se ugrubo skenira čitav prostor te se tako izgradi semantička mapa prostora. Takva mapa sadrži sve dosad detektirane objekte, uz pomoć kojih se daljnjom obradom lokacija tih objekata dron usmjerava na detaljnu pretragu.

Razvijeni sustav testiran je u simulatoru Webots na već postojećim mapama gdje je razvijeni sustav dao bolje rezultate od klasičnih metoda pretraživanja.

Kako bi se ovaj rad primjenjivao, potrebno je prije pretraživanja prostora izgraditi *octomapu* prostora nekim drugim robotom, jer letjelica Crazyflie u vrijeme pisanja rada nema mogućnost nadogradnje kako bi se njome mogla izgraditi *octomapa.*

U budućem radu ostaje još testirati ovaj sustav u stvarnosti te ga usporediti s nekim od klasičnih metoda pretraživanja.

Literatura

[1] Webots Open-source Mobile Robot Simulation Software. Poveznica: http://www.cyberbotics.com; Pristupljeno: 3. travnja 2024.

[2] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, Wolfram Burgard *OctoMap: An Efficient Probabilistic {3D} Mapping Framework Based on Octrees*, Autonomous Robots, (2013). Poveznica: https://octomap.github.io; pristupljeno 4. travnja 2024.

[3] Goran Obradović *Strukture podataka za CAD objekte – Oktalno stablo*, (2008). Poveznica https://www.zemris.fer.hr/predmeti/rg/diplomski/08Obradovic/index.html; pristupljeno 5. travnja 2024.

[4] Wikimedia Commons *Schematic drawing of an octree, a data structure of computer science* (2010). Poveznica: https://commons.wikimedia.org/wiki/File:Octree2.svg; pristupljeno 10. travnja 2024.

[5] Wang, Chien-Yao and Bochkovskiy, Alexey and Liao, Hong-Yuan Mark *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2023)

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár *Microsoft COCO: Common Objects in Context* (2015)

[7] Steven Macenski, Francisco Martin, Ruffin White, Jonatan Ginés Clavero *The Marathon 2: A Navigation System*, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020)

[8] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, William Woodall *Robot Operating System 2: Design, architecture, and uses in the wild*, Science Robotics, (2022)

[9] Crazyswarm2 *ROS2 Tutorials.* Poveznica: https://imrclab.github.io/crazyswarm2/tutorials.html#navigate-the-crazyflie; pristupljeno 10. ožujka 2024.

[10] Angus Johnson *Clipper - an open source freeware library for clipping and offsetting lines and polygons, (*2014). Poveznica: http://www.angusj.com/clipper2/Docs/Overview.htm; pristuljeno 18. travnja 2024.

[11] Thibault Ladebeck *Pyclipper - A Python wrapper for the Clipper library, (2023)*. Poveznica: https://github.com/greginvm/pyclipper; pristupljeno 18. travnja 2024.

[12] Radim Řehůřek, Petr *Sojka Software Framework for Topic Modelling with Large Corpora*, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (ELRA), (2010)

[13] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean *Efficient Estimation of Word Representations in Vector Space*, arXiv preprint arXiv:1301.3781 (2013)

[14] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD), (1996)

[15] Google *Google News Dataset* Google Research. Poveznica: https://code.google.com/archive/p/word2vec/; pristupljeno 37. travnja. 2024.

[16] Wikipedia, The Free Encyclopedia *Color difference* (2024, srpanj). Poveznica: https://en.wikipedia.org/wiki/Color_difference; pristupljeno 14. srpnja 2024.

[17] Webots Cloud *Crazyflie Proto*. Poveznica: https://github.com/cyberbotics/webots/blob/released/projects/robots/bitcraze/crazyflie/protos/Crazyflie.proto; Pristupljeno 3. travnja 2024.

[18] Webots Cloud *Apartment Environment*. Poveznica: https://github.com/cyberbotics/webots/blob/released/projects/samples/environments/indoor/worlds/apartment.wbt; Pristupljeno 3. travnja 2024.

[19] Webots Cloud *Break Room Environment*. Poveznica: https://github.com/cyberbotics/webots/blob/released/projects/samples/environments/indoor/worlds/break_room.wbt; Pristupljeno 23. travnja 2024.

Sažetak

U ovom radu razvijena je metoda koja uz pomoć autonomne bespilotne letjelice pretražuje zatvoreni prostor manje površine, poput stana ili ureda s ciljem pronalaska zadanog predmeta. Autonomna bespilotna letjelica, opremljena kamerom, pomoću YOLO algoritma detektira objekte u prostoru. Detektiranim objektima estimira se lokacija te se na temelju detektiranih objekata gradi semantička mapa prostora. U daljnjem detaljnom pretraživanju u obzir se uzima semantika prepoznatih objekata kako bi se skratilo vrijeme pretrage. Metoda je testirana u simulatoru i njena je učinkovitost pokazana na nekoliko scenarija.

Ključne riječi: autonomna bespilotna letjelica, YOLO, pretraga prostora, oktalno stablo, Crazyflie, ROS2, Webots

Summary

In this paper, a method using an autonomous drone was developed to search a small, enclosed space with the aim of finding a given object. The drone is equipped with a camera to detect objects using the YOLO algorithm in a space such as an office or apartment. The location of the detected objects is estimated to construct a semantic map of the space. The recognized objects' semantics can then shorten search time. This method was tested in a simulator with effectiveness demonstrated in several scenarios

Keywords: autonomous unmanned aerial vehicle, YOLO, space search, octree, Crazyflie, ROS2, Webots

Skraćenice

ROS	*Robot Operating System*	robotski operacijski sustav
YOLO	*You Only Look Once*	algoritam za detekciju objekata u slici
NLP	*neurolingvističko programiranje*	grana umjetne inteligencije