LATVIJAS UNIVERSITĀTE
EKSAKTO ZINĀTŅU UN TEHNOLOĢIJU FAKULTĀTE
DATORIKAS NODAĻA

LIDOJUMĀ IEGŪTU DATU APVIENOŠANA AR VOKSEĻU SLAM SISTĒMU

BAKALAURA DARBS

Autors: **Ārija Kalniņa**
Studenta apliecības Nr.: ak21374
Darba vadītājs: Mg. sc. comp. Pēteris Račinskis

RĪGA 2025

ANOTĀCIJA

Darba ietvarā tiek izstrādāta metode lidojumā iegūtu datu apvienošanai ar vokseļu SLAM sistēmu. SLAM sistēmas ir robotikā un autonomajā navigācijā plaši izmantotas sistēmas, kas nodrošina iespēju aģentam vienlaikus veikt apkārtējās vides kartēšanu un sevis lokalizēšanu izveidotajā kartē. Darbā tiek aplūkoti datu apvienošanas metodes teorētiskie aspekti, projektējums, metodes novērtēšanai nepieciešamo datu ieguves metodes, kā arī metodes novērtēšanas rezultāti. Metodes novērtēšanas rezultāti norāda uz precīzu, retinātu lidojumā iegūtu datu integrāciju SLAM sistēmā.

Atslēgvārdi: SLAM, vokselis, punktu mākonis, datu apvienošana, segmentācija

ABSTRACT

AERIAL DATA FUSION IN VOXEL-BASED SLAM SYSTEMS

In this work a method is developed for fusing aerial in-flight data with a voxel-based SLAM system. SLAM systems are widely used in robotics and autonomous navigation, enabling the agent to simultaneously map the environment and localize itself in the created map. The work discusses the theoretical aspects of the data fusion method, design of the method, data acquisition methods necessary for method evaluation, and the results of method evaluation. The results indicate accurate and sparse integration of aerial data into the SLAM system.

Keywords: SLAM, voxel, pointcloud, data fusion, segmentation

SATURS

APZĪMĒJUMU SARAKSTS

- BRIEF - iezīmju deskriptoru iegūšanas algoritms (angliski: *binary robust independent elementary features*).
- FAST - iezīmju izgūšanas algoritms (angliski: *features from accelerated segment test*).
- Hz - hercs
- IoU - šķēlums pār apvienojumu (angliski: *intersection-over-union*).
- LIDAR - attāluma lāzermērīšana (angliski: *light detection and ranging*).
- lpp. - lappuse.
- Nr. - numurs
- ORB - iezīmju izgūšanas algoritms (angliski: *oriented FAST and rotated BRIEF*).
- RGB-D - attēlveidošanas sistēma, kas vienlaikus uztver gan krāsu informāciju (RGB - sarkans, zaļš un zils), gan dziļumu (D - dziļums).
- ROS – robotu operētājsistēma (angļu: *Robot Operating System*) – satvars robotu programmatūras izstrādei.
- SIFT - iezīmju izgūšanas algoritms (angliski: *scale-invariant feature transform*).
- SLAM – vienlaicīga lokalizācija un kartēšana (angliski: *simultaneous localization and mapping*).
- SURF - iezīmju izgūšanas algoritms (angliski: *speeded up robust features*).

IEVADS

Pārvietojoties telpā, apkārtējās vides izkārtojums un sava atrašanās vieta šajā izkārtojumā ne vienmēr ir iepriekš zināmi. Spilgts piemērs minētajai situācijai būtu nokļūšana svešā pilsētā bez kartes, kompasa, globālās satelītnavigācijas sistēmas vai citiem rīkiem, kas ļautu noskaidrot savu atrašanās vietu. Kā jau noprotams, navigācijai nezināmajā apkārtnē būtu galvenokārt jāpaļaujas uz maņām: redzi, dzirdi, tausti, ožu vai pat garšu. Līdzīga situācija piemeklētu arī nedzīvus aģentus[1], tikai nedzīvu aģentu gadījumā maņu uzdevumu veiktu sensori: kameras, radari, LIDAR sensori, attāluma sensori un cita veida sensori.

Problēmu, kurā nedzīvs aģents pārvietojas telpā, kartējot apkārtni un vienlaikus nosakot savu atrašanās vietu šajā apkārtnē, dēvē par SLAM jeb vienlaicīgu lokalizāciju un kartēšanu. Klasiskas SLAM sistēmas darbības pamatideju var aprakstīt kā četru secīgu posmu izpildi: iezīmju izgūšanu, datu asociāciju, sistēmas stāvokļa novērtēšanu un kartes atjaunināšanu [1]. Iezīmju izgūšanas posmā no aģenta sensoru mērījumiem tiek izgūtas iezīmes - labi identificējami un invarianti telpas elementi, piemēram, objektu stūri vai šķautnes. Datu asociācijas solī tiek meklētas asociācijas starp jauniegūtajām iezīmēm un jau iepriekš izgūtajām iezīmēm, piemēram, pārejas matricas starp iepriekšējā sensoru mērījuma un tagadējā sensoru mērījuma iezīmēm. Sistēmas stāvokļa novērtēšanas posmā, izmantojot iegūtās asociācijas, tiek novērtēta aģenta atrašanās vieta kartē. Pēdējā (kartes atjaunināšanas) posmā tiek atjaunināta karte, lai tā atbilstu jauniegūtajiem sistēmas stāvokļa novērtējumiem.

SLAM sistēmām ir plašs pielietojums autonomu aģentu, konkrētāk, pašbraucošu automašīnu un autonomu robotu [2, 3], izstrādē. No SLAM sistēmas iegūtie dati - apkārtējās vides karte un aģenta atrašanās vieta šajā kartē - tiek izmantoti kā ievade turpmākajos plānošanas algoritmos, atrodot optimālu maršrutu līdz vēlamajam galamērķim un izvairoties no kartē redzamajiem šķēršļiem bez cilvēka manuālas iesaistes [4]. SLAM sistēmas tiek pielietotas arī medicīnas nozarē, piemēram, endoskopisko izmeklējumu laikā pacienta iekšējo orgānu kartēšanai [5], kā arī navigācijas palīgsistēmu izstrādē vājredzīgām personām [6].

Neatkarīgi no tā, kādam nolūkam tiek izmantota SLAM sistēma, tās darbības rezultātā iegūto datu kvalitāte ir lielā mērā atkarīga ne tikai no pašas SLAM sistēmas implementācijas, bet arī no izmantotajiem sensoriem. Integrējot SLAM sistēmā datus no dažādu veidu sensoriem, tiek kompensēti katra individuālā sensora ierobežojumi un trūkumi, piemēram, kameru jutīgums pret apgaismojuma izmaiņām, kas nav tik sastopams LIDAR sensoros. Savukārt izmantojot sensorus ar atšķirīgu izvietojumu telpā, ir iespējams vienā sensoru datu ieguves reizē iegūt informāciju par plašāku aģenta apkārtnes teritoriju. Attiecīgi, integrējot SLAM sistēmā lidojumā iegūtus datus, ir iespējams iegūt detalizētāku informāciju par aģenta apkārtni tajās vie-

[1] Šajā kontekstā jēdziens "nedzīvs aģents" apzīmē jebkuru nedzīvu būtni, piemēram, robotu, pašbraucošu mašīnu vai klēpjdatoru, kas spēj veikt kartēšanu un lokalizāciju telpā. Turpmākajās nodaļās jēdziena "nedzīvs aģents" vietā tiks izmantots saīsināts jēdziens "aģents".

tās, kas ir grūti pārskatāmas vai sasniedzamas citādu sensoru izvietojumu gadījumā, piemēram, vietās starp sienām vai uz kāpnēm.

Problēmas definīcija un darba mērķis

Bakalaura darba ietvaros pētāmā problēma ir lidojumā iegūtu datu apvienošana ar vokseļu SLAM sistēmu, respektīvi, lidojumā iegūto datu integrācija SLAM sistēmā, papildinot SLAM sistēmas izveidoto vokseļu karti un punktu mākoni ar lidojumā iegūtajiem datiem.

Lidojumā iegūtie dati tiek iegūti no lidojoša bezpilota lidaparāta. Bezpilota lidaparāts pārvietojas telpā ar uz leju vērstiem sensoriem, iegūstot sensoru mērījumus par apkārtni zem tā. Bezpilota lidaparāta poza jeb atrašanās vieta telpā ir zināma, un ir pieejama lidojumā iegūtajos datos. Poza ir izteikta relatīvi lidojuma uzsākšanas sākumpunktam. Tiek pieņemts, ka SLAM sistēmas ievadē padotā bezpilota lidaparāta poza ir patiesa, tas ir, atbilst patiesajai bezpilota lidaparāta atrašanās vietai telpā.

Izņemot bezpilota lidaparāta pozu, lidojumā iegūtie dati ietver sevī arī bezpilota lidaparāta kameras attēlus un apkārtējās vides absolūtā augstuma punktus ar tiem atbilstošajiem klašu vektoriem - vektoriem, kas norāda katra punkta piederību konkrētām virsmas klasēm. Virsmas klases ir iepriekš definētas kategorijas, kurās tiek kategorizēta bezpilota lidaparāta sensoru novērotā teritorija jeb, tā kā bezpilota lidaparāta sensori ir vērsti uz leju, zemes virsma.

Tā kā bakalaura darba izstrādes laikā vēl nebija pieejami lidojumā iegūti dati no reāla bezpilota lidaparāta, vēlamie dati tika ģenerēti imitētā vidē, rezultātā iegūstot identiska formāta datus. Ņemot vērā, ka bakalaura darbā pētāmā problēma ir nevis pašu datu iegūšana, bet gan to integrācija SLAM sistēmā, attiecīgais ierobežojums neietekmēja problēmas atrisinājuma izvēli. Ģenerētie dati tika izmantoti tikai izvēlētas metodes novērtēšanai.

SLAM sistēma, kurā tiek integrēti lidojumā iegūtie dati, ir vokseļu, respektīvi, apkārtējās vides kartēšanas rezultātā tiek iegūta vokseļu karte, kur katrā vokselī tiek glabāti ieraksti par vokselim atbilstošo apkārtējās vides apgabalu. Ieraksti satur informāciju par apgabala semantiku, tas ir, tie satur attiecīgā apgabala semantikas vektoru. Informācija par apkārtējās vides struktūru (telpas izkārtojumu) tiek glabāta atsevišķā punktu mākonī.

Atbilstoši pētāmajai problēmai darba mērķis ir apvienot lidojumā iegūtus datus ar vokseļu SLAM sistēmu, paplašinot SLAM sistēmas funkcionalitāti ar lidojumā iegūtu datu apstrādi. Mērķa sasniegšanai ir izvirzīti sekojošie uzdevumi:

- Teorētisko pamatu izklāstīšana, aprakstot darba ietvaros izmantotos jēdzienus. Teorētisko pamatu izklāstīšanai par pamatu tika ņemta kursa darbā [7] veiktā literatūras analīze.
- SLAM sistēmas modificēšana, paplašinot sistēmas funkcionalitāti ar lidojumā iegūtu datu apstrādi, rezultātā ievietojot apstrādātos datus SLAM sistēmas vokseļu kartē un punktu mākonī. SLAM sistēmas modificēšana iekļauj sevī jaunu sistēmas vienību izstrādi un esošo sistēmas vienību pielāgošanu darba mērķa sasniegšanai, nesamazinot iepriekšējo SLAM sistēmas funkcionalitāti.

- Modificētās SLAM sistēmas veiktspējas novērtēšana, salīdzinot SLAM sistēmas rezultātus pēc lidojumā iegūtu datu ievades ar patiesajiem (angliski: *ground truth*) datiem par apkārtējo vidi un novērtējot to sakritību.

Darba struktūra

Atbilstoši darbā izvirzītajiem mērķiem darbs sastāv no teorijas nodaļas, metodes nodaļas, kā arī rezultātu un diskusijas un secinājumu nodaļām.

Teorijas nodaļā tiek veikta turpmākajās darba nodaļās izmantoto jēdzienu un ar tiem saistīto teorētisko pamatu izklāstīšana. Teorijas nodaļa tiek iedalīta trīs apakšnodaļās. Pirmā apakšnodaļa ir veltīta jēdziena "poza" aprakstīšanai, detalizēti izklāstot iespējamos pozas reprezentācijas veidus. Otrā apakšnodaļa koncentrējas uz SLAM sistēmu uzbūves pamatprincipiem un klasiskajiem SLAM kategorizācijas variantiem. Trešā nodaļa apraksta kameras kalibrēšanas rezultātā iegūtos parametrus.

Metodes nodaļā tiek aprakstīta implementētā SLAM sistēmas modificēšanas metode, kā arī metodes rezultātu novērtēšanas metode. Metodes nodaļa sastāv no sešām apakšnodaļām. Pirmajā apakšnodaļā tiek sniegts augsta līmeņa metodes projektējuma apraksts. Turpmākajās četrās nodaļās metodes projektējums tiek izklāstīts detalizētāk, katrā nodaļā aprakstot konkrētu, secīgu metodes posmu. Pēdējā nodaļā tiek aprakstīta rezultātu novērtēšanas metode, konkrētāk, datu ieguves metode un rezultātu novērtēšanas metrikas.

Rezultātu un diskusijas nodaļā tiek aprakstīti rezultātu novērtēšanā iegūtie rezultāti. Savukārt secinājumos tiek sniegti vispārīgi secinājumi par paveikto darbu.

Matemātiskie apzīmējumi

Lai darba lasītājiem būtu vieglāk orientēties darbā aprakstītajos algoritmos un vienādojumos, attiecīgajā nodaļā tiek sniegts biežāk izmantoto matemātisko apzīmējumu skaidrojums.

Viens no pirmajiem izmantotajiem apzīmējumiem darba ietvaros ir p jeb **poza**. Poza sastāv no pozīcijas un orientācijas. Šī darba ietvaros **pozīcija** tiek apzīmēta ar t, savukārt orientācijas apzīmējums ir atkarīgs no reprezentācijas. **Kvaternionu** gadījumā tas ir q, **rotāciju matricu** gadījumā tas ir R. Jāpiemin, ka kvaternioni teorijas nodaļā tiek apzīmēti arī ar Q, taču tikai kvaternionu reprezentācijas apakšnodaļas ietvaros.

Pārejas matricu apzīmēšanai tiek izmantots apzīmējums T. Pāreja no atskaites sistēmas A uz atskaites sistēmu B tiek apzīmēta kā T_{BA}. Ja pārejas matricas apzīmējumam ir norāde tikai uz vienu atskaites sistēmu, piemēram, T_L, tas nozīmē, ka atskaites sistēma, no kuras tiek veikta pāreja, nav specifiski definēta. Attiecīgajos gadījumos minētās atskaites sistēmas definēšana nav nepieciešama, tā kā parejas matrica netiek izmantota pārejai no vienas atskaites sistēmas uz citu, bet gan kā pozas reprezentācija.

Lai norādītu uz konkrēta apzīmējuma piederību kādai atskaites sistēmai, tiek lietots for-

māts apzīmējums$_X$, kur X ir atskaites sistēma. Respektīvi, dotā apzīmējuma vērtība ir izteikta relatīvi atskaites sistēmai X. Atskaites sistēmas darba ietvaros tiek apzīmētas ar lielajiem burtiem.

Turpmāk tiek dots algoritmos izmantoto apzīmējumu skaidrojums:

- r - izšķirtspēja;
- o_s_arr - augstuma mērījumi;
- $submap$, s - lokālā karte;
- e – augstuma punkts un tā virsmas klašu vektors;
- i – attēls;
- p - poza;
- x_buf – x buferis;
- $global\_mapper$ - globālais kartētājs;
- c - klašu vektors;
- key, k – pozai atbilstošā atslēga (viendimensiju vērtība, pēc kuras meklēt pozu);
- $voxel\_map$ - globālā vokseļu karte;
- $pointcloud$ - globālais punktu mākonis;
- $publisher$ – globālā kartētāja ROS publicētājs;
- v – vokselis.

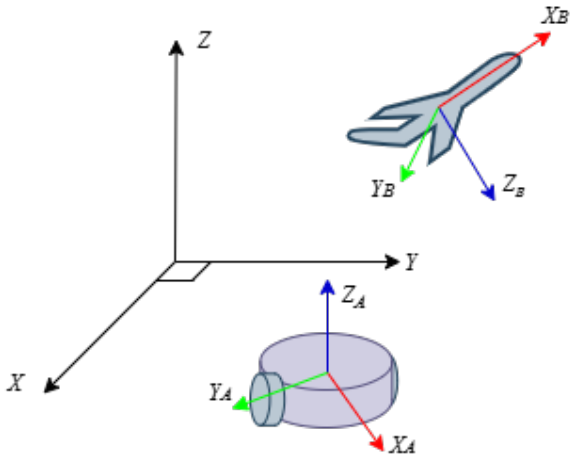
1. TEORIJA

1.1. Poza un koordinātu transformācijas

Robotikas un datorredzes nozarē fundamentāls termins, kas tiek izmantots noteikta objekta atrašanās vietas apzīmēšanai, ir poza. Poza ir objektam atbilstošās atskaites sistēmas pozīcija un orientācija. Tipveida gadījumos attiecīgā atskaites sistēma ir centrēta objekta atskaites punktā, ar X asi vērstu uz objekta priekšpusi jeb kustības virzienu (skatīt 1.1 attēlu). Jāņem vērā, ka objekta pozu ir iespējams izteikt tikai relatīvi kādam citam objektam, konkrētāk, cita objekta atskaites sistēmai, piemēram, orientiera atskaites sistēmai [8]. Ja objektu ir iespējams modelēt kā sastāvošu no vairākiem cietiem ķermeņiem (angliski: *rigid bodies*) - ķermeņiem, kurus rotējot vai pārvietojot, attālums starp jebkurām divām ķermeni veidojošajām daļiņām nemainās, neveidojoties objekta "spoguļattēlam" [9, 92 lpp.] -, pozu mēdz definēt katram objekta cietajam ķermenim atsevišķi.

Plaknes gadījumā pozu p var pilnībā reprezentēt ar trīs dimensiju vektoru $p = (x, y, \theta)$, kur x un y atbilst objekta pozīcijai (punkta koordinātas), un θ atbilst objekta orientācijai plaknē, parasti izteiktai grādos vai radiānos.

Telpas gadījumā objekta pozu p var pilnībā reprezentēt ar sešu dimensiju vektoru $p = (x, y, z, \alpha, \beta, \gamma)$, kur x, y un z atbilst objekta pozīcijai (punkta koordinātas $(x; y; z)$ trīs koordinātu sistēmā X, Y un Z), un α, β, γ atbilst objekta orientācijai telpā, kur katrs leņķis ir rotācija ap konkrētu koordinātu sistēmas X, Y, Z asi. Lai gan objekta orientāciju ir iespējams reprezentēt, izmantojot trīs leņķus, praktiski objekta orientācijas reprezentācijai tiek izmantotas dažādas metodes, piemēram, rotāciju matricas, Eilera leņķi, kvaternioni un ass-leņķa (angliski: *axis-angle*) reprezentācija.



Att. 1.1: **Objekti un tiem atbilstošās atskaites sistēmas telpā**

1.1.1. Rotāciju matricas

Orientāciju ir iespējams aprakstīt, izmantojot rotāciju matricas. Pieņem, ka atskaites sistēmas A poza ir izteikta relatīvi atskaites sistēmai B, kuru veido trīs ortogonāli bāzes vektori jeb asis X, Y un Z. Atskaites sistēmas A rotāciju leņķī θ ap atskaites sistēmas B X asi var izteikt kā:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{1.1}$$

Atskaites sistēmas A rotāciju ap Y asi var izteikt kā:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{1.2}$$

Rotāciju ap Z asi var izteikt kā:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

Izmantojot iepriekšminētās trīs rotāciju matricas, ir iespējams iegūt jebkuru atskaites sistēmas A rotāciju relatīvi atskaites sistēmai B. Attiecīgi, apvienojot minētās matricas, ir iespējams iegūt jebkuru objekta orientāciju telpā. Matricu apvienošana tiek veikta, veicot matricu reizināšanu, piemēram, $R = R_z(\alpha)R_y(\beta)R_x(\gamma)$ (skatīt 1.4 vienādojumu). Šajā piemērā objekts vispirms tiek rotēts ap X asi, pēc tam ap Y asi un beidzot ap Z asi [9].

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\sin\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{1.4}$$

Rotāciju matricas izmantošanas gadījumā objekta pozu ir iespējams reprezentēt vienā 4x4 pārejas matricā T, kas sastāv no objekta jeb tā atskaites sistēmas orientācijas R un pozīcijas t.

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{1.5}$$

Rotāciju matricai $R = R_z(\alpha)R_y(\beta)R_x(\gamma)$ atbilstošā pārejas matrica ir redzama 1.6 izteiksmē. Šajā gadījumā objekts vispirms tiek rotēts ap X asi leņķī γ, pēc tam ap Y asi leņķī β, pēc tam ap Z asi leņķī α un beigās tiek veikta paralēlā pārnese par vektoru $t = (x_t, y_t, z_t)$.

$$T = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \sin\alpha\sin\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.6)$$

Pārejas matricas robotikā un datorredzē tiek izmantotas vektoru koordinātu pārveidei no vienas atskaites sistēmas citā, piemēram, pārveidojot attēla pikseļu koordinātas no kustīgai kamerai atbilstošās atskaites sistēmas uz fiksētu atskaites sistēmu kādā noteiktā telpas punktā. Vektora v koordinātu $v = (x; y; z)$ pārveidi no atskaites sistēmas A uz atskaites sistēmu B var izteikt kā vienādojumu $v_B = T_{BA}v_A$, kur T_{BA} ir pārejas matrica no atskaites sistēmas A uz atskaites sistēmu B, v_A ir paplašinātais vektors v atskaites sistēmā A, un v_B ir rezultātā iegūtais paplašinātais vektors atskaites sistēmā B. Vektora paplašināšana notiek, pievienojot tā galā vieninieku, respektīvi, $v_A = (x; y; z; 1)^T$.

$$v_B = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_{BA}v_A \quad (1.7)$$

1.1.2. Eilera leņķi

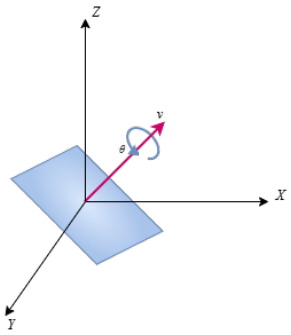
Eilera leņķu [10, 11] gadījumā objekta orientācija tiek reprezentēta, izmantojot trīs secīgus leņķus α, β, γ, kur katrs leņķis atbilst rotācijai ap konkrētu objekta koordinātu sistēmas asi. Tā kā koordinātu sistēmas asu izvietojums katrai nākamajai rotācijai ir atkarīgs no iepriekšējām rotācijām, lai spētu precīzi definēt orientāciju, izņemot leņķu vērtības, ir nepieciešams norādīt arī attiecīgo rotāciju veidus (ap kuru asi tiek veikta rotācija katram leņķim). Ņemot vērā ierobežojumu, ka divas secīgas rotācijas nevar tikt veiktas ap vienu un to pašu asi, kopumā ir iespējams iegūt 12 dažādas Eilera leņķu rotāciju secības. No 12 iespējamajām rotāciju secībām, populāras ir rotācijas Z-X-Z, Z-Y-Z un Z-Y-X. Z-Y-X Eilera leņķu gadījumā rotācija vispirms notiek ap Z asi, pēc tam ap rotēto Y asi un pēc tam ap divreiz rotēto X asi.

Alternatīvi Eilera leņķi var tikt izmantoti objekta orientācijas reprezentācijai attiecībā pret fiksētu koordinātu sistēmu. Šādā gadījumā koordinātu sistēmas asu izvietojums pēc rotācijām nemainās, taču precīzai orientācijas definīcijai joprojām ir nepieciešams norādīt, ap kuru asi tiek veikta attiecīgā rotācija katram leņķim. Piemēram, X-Y-Z fiksētas koordinātu sistēmas Eilera leņķu gadījumā rotācija vispirms notiek ap fiksētu X asi, pēc tam ap fiksētu Y asi un pēc tam ap fiksētu Z asi. Jebkurai rotāciju secībai attiecībā pret fiksētu koordinātu sistēmu atbilst pretējas secības rotāciju secība paša objekta koordinātu sistēmā. Piemēram, X-Y-Z fiksētas koordinātu sistēmas Eilera leņķi definē tādu pašu orientāciju kā Z-Y-X objekta koordinātu sistēmas Eilera leņķi.

Eilera leņķu trūkums ir iespējamā singularitāte. Singularitāte rodas gadījumā, kad rotācijas asis sakrīt pirmajā un pēdējā (trešajā) rotācijā, padarot pirmās un trešās rotācijas leņķus neizšķiramus.

1.1.3. Ass-leņķa reprezentācija

Ass-leņķa reprezentācijas gadījumā orientācija tiek aprakstīta, izmantojot vienības vektoru v un leņķi θ. Vienības vektors $v = (x, y, z)$ atbilst asij, ap kuru tiek veikta rotācija leņķī θ (skatīt 1.2 attēlu).



Att. 1.2: **Ass-leņķa reprezentācija**

1.1.4. Kvaternioni

Atšķirībā no Eilera leņķiem kvaternionu reprezentācijai nepiemīt singularitātes problēma. Kvaternionu [12] var definēt kā izteiksmi $Q = w + ix + jy + kz$, kur w, x, y, z ir reāli skaitļi un kur i, j, k apzīmē trīs imaginārās vienības, kas apmierina sekojošās izteiksmes:

$$\begin{aligned} &i^2 = j^2 = k^2 = -1 \\ &ij = k, \quad jk = i, \quad ki = j \\ &ji = -k, \quad kj = -i, \quad ik = -j \end{aligned} \tag{1.8}$$

Kvaternionu saskaitīšanai piemīt asociativitāte, komutativitāte un distributivitāte. Kvaternionu saskaitīšanu vai atņemšanu var izteikt kā vienādojumu:

$$Q \pm Q' = w \pm w' + i(x \pm x') + j(y \pm y') + k(z \pm z') \tag{1.9}$$

Savukārt kvaternionu reizināšanu var izteikt kā 1.10 vienādojumu. Kvaternionu reizināšana ir asociatīva un distributīva, bet nav komutatīva.

$$\begin{aligned} QQ' = {} & ww' + iwx' + jwy' + kwz' \\ & + ixw' + i^2xx' + ijxy' + ikxz' \\ & + jyw' + jiyx' + j^2yy' + jkyz' \\ & + kzw' + kizx' + kjzy' + k^2zz' \end{aligned} \tag{1.10}$$

Izmantojot 1.8 izteiksmes, 1.10 vienādojumu var vienkāršot:

$$\begin{aligned} QQ' = {} & ww' - xx' - yy' - zz' \\ & + (wx' + xw' + yz' - zy')i \\ & + (wy' + yw' - xz' + zx')j \\ & + (wz' + zw' + xy' - yx')k \end{aligned} \tag{1.11}$$

Objekta orientācijas reprezentācijai telpā tiek izmantoti vienības kvaternioni. Pieņem, ka objekta pozīcija ir definēta, izmantojot vektoru $v = (v_x, v_y, v_z)$, kas atbilst kvaternionam $v = iv_x + jv_y + kv_z$. Vektora v rotācija ap vienības kvaterniona $Q = w + ix + jy + kz$ definētu asi (ar virzienu $(x; y; z)$) ir izsakāma kā operācija $Qv\overline{Q}$, kur $\overline{Q}$ ir vienības kvaternionam Q pretējs kvaternions (angliski: *conjugate*) $\overline{Q} = w - ix - jy - kz$. Vienības kvaterniona gadījumā $Q\overline{Q} = \overline{Q}Q = w^2 + x^2 + y^2 + z^2 = 1$ [10].

1.2. SLAM sistēmas

Vienlaicīga lokalizācija un kartēšana jeb SLAM [7] ir fundamentāla robotikas problēma, kurā aģentam, kas veic apkārtējās vides kartēšanu, vienlaicīgi tiek noteikta atrašanās vieta šajā izveidotajā kartē. Kopš 1986. gada jeb klasiskā SLAM pirmsākumiem SLAM sistēmas ir ievērojami attīstījušās, implementējot savos risinājumos dažādas datorredzes [13, 14], mašīnmācīšanās [15, 16] un citu jomu inovācijas. Turpmākajās apakšnodaļās tiks aplūkoti SLAM sistēmu uzbūves pamatprincipi un iespējamā kategorizācija.

1.2.1. SLAM sistēmu uzbūves pamatprincipi

Klasisku SLAM sistēmu ir iespējams iedalīt četros moduļos:

- priekšgala modulī, kurā tiek apstrādāti sensoru dati, izgūtas iezīmes un veikta datu asociācija (sensoru mērījumu saistīšana ar konkrētu apkārtējās vides orientieri, piemēram, trīs dimensiju punktu [17]),
- aizmugures modulī, kurā tiek veikta datu optimizācija, sistēmas stāvokļa novērtēšana un kartes atjaunināšana,
- cilpu noslēgšanas (angliski: *loop closure*) modulī, kurā tiek identificētas iepriekš kartētas vietas telpā un samazināta uzkrātā novirze,
- kartēšanas modulī, kurā tiek izveidota pati karte.

Alternatīvi SLAM sistēmu ir iespējams sadalīt tikai divos moduļos: priekšgala un aizmugures. Šajā gadījumā cilpu noslēgšana un kartes izveidošana tiek iekļautas priekšgala modulī.

Priekšgala modulis

Priekšgala moduļa implementācija SLAM sistēmās ir atkarīga no ievadē saņemto sensoru datu formāta. Izteikti ir iespējams atdalīt priekšgala moduļa implementācijas variantus gadījumiem, kad ievadē tiek saņemti punktu mākoņi, un gadījumiem, kad ievadē tiek saņemti kameras attēli.

Kameras attēlu apstrādes gadījumā priekšgala modulī iezīmes tiek izgūtas no attēliem. Vispirms attēlos tiek identificēti intereses punkti un iegūta to atrašanās vieta. Intereses punkti ir stingri noteicami pikseļi, kuriem ir liels informācijas saturs. Par stingri noteicamiem pikseļiem dēvē tādus pikseļus, kurus ir iespējams identificēt dažādos attēlos neatkarīgi no izmaiņām attēlu rotācijā, apgaismojumā vai citos parametros. Attiecīgi stingri noteicamie pikseļi ir invarianti. Tālāk no intereses punktiem tiek iegūti iezīmju deskriptori, kuros ir iekodēta informācija, kas ļauj atšķirt vienu iezīmi no citas. Citiem vārdiem sakot, vispirms tiek pielietots iezīmju detektors, lai noteiktu iezīmju atrašanās vietu, un pēc tam tiek pielietots iezīmju deskriptors, lai iekodētu informāciju par konkrētajām iezīmēm. Iezīmju izgūšanai ir pieejami tādi algoritmi (un to paplašinājumi) kā SIFT, SURF un ORB.

SIFT algoritma gadījumā iezīmes tiek izgūtas četros soļos. Vispirms tiek izveidota Gausa starpību mērogu telpa (angliski: *difference of Gaussians (DoG) scale-space*) ($I(x, y)$ ir sākotnējais attēls):

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{1.12}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2)} \tag{1.13}$$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1.14}$$

Pēc tam tiek iegūti intereses punkti, kas atrodas vietās, kur Gausa starpības funkcija mērogu telpā sasniedz maksimumu vai minimumu. Intereses punktiem tiek piešķirta orientācija un gradienta modulis:

$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \tag{1.15}$$

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{1.16}$$

Pēdējā posmā no iegūtajām vērtībām tiek izveidots SIFT iezīmju deskriptors.

SURF algoritma gadījumā iezīmju izguve tiek bāzēta uz Hesa matricas (angliski: *Hessian matri*x), kas, ja ir dots punkts $z = (x, y)$, ir definēta šādi:

$$X(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{1.17}$$

$L_{xx}(x, \sigma)$ - attēla I konvolūcija punktā x, izmantojot Gausa funkcijas otrās pakāpes atvasinājumu $\frac{\delta^2}{\delta x^2} g(\sigma)$

ORB algoritms ir bāzēts uz FAST un BRIEF algoritmiem. Intereses punkti tiek izgūti, izmantojot FAST algoritmu, bet deskriptori tiek iegūti, izmantojot BRIEF algoritmu. FAST algoritmā, lai attēla pikselis P būtu intereses punkts, n no 16 tam blakus esošajiem pikseļiem ir jābūt par T lielākai vai zemākai intensitātei I_p nekā attiecīgajam pikselim P.

Punktu mākoņu apstrādes gadījumā priekšgala modulī tiek veikta skenējumu salīdzināšana (angliski: *scan matching*). Skenējumu salīdzināšanā, lai novērtētu pašreizējā skenējuma pozu, tiek salīdzināti blakusesošie skenējumi. Par skenējumu tiek dēvēts punktu mākonis, kas iegūts viena sensoru mērījuma laikā.

Viens no izplatītiem veidiem, kā veikt skenējumu salīdzināšanu, ir izmantojot ICP jeb iteratīvo tuvāko punktu (angliski: *iterative closes point*) algoritmu. ICP algoritmu var definēt kā tādas transformācijas (R, t) atrašanu (R ir rotācijas matrica un t ir translācijas vektors), kas, ja ir dotas divas neatkarīgi iegūtas 3D punktu kopas M ($m_i \in M$) un D ($d_i \in D$), minimizē sekojošo izmaksu funkciju:

$$E(R, t) = \frac{1}{N} \sum_{i=1}^{N} ||m_i - (Rd_i + t)||^2 \tag{1.18}$$

Aizmugures modulis

SLAM sistēmas aizmugures modulī tiek veikta aģenta trajektorijas un kartes optimizēšana, apstrādājot priekšgala modulī iegūtos datus. Optimizēšanas rezultātā tiek precizēta aģenta pārvietošanās trajektorija un orientieru atrašanās vieta telpā, tādā veidā samazinot trajektorijas novirzi no īstās aģenta pārvietošanās trajektorijas un saglabājot kartēšanas uzticamību. Konsekvences nolūkos jāpiemin, ka SLAM sistēmās par orientieriem tiek dēvētas apkārtējā vidē novērotās un izgūtās noturīgās iezīmes, kas turpmāk tiek izmantotas kartēšanai un aģenta lokalizācijai izveidotajā kartē.

Aizmugures moduļa implementācijas metodes ir iespējams iedalīt divās dažādās kategorijās: filtrēšanā bāzētās metodēs un optimizācijā bāzētās metodēs. Pie filtrēšanā bāzētām metodēm pieder tādas metodes kā paplašinātais Kalmana filtrs, daļiņu filtrs (angliski: *particle filter*) vai informācijas filtrs. Šajās metodēs tiek pakāpeniski atjaunināts pēdējā jeb tagadējā sistēmas stāvokļa - tagadējās aģenta pozas un tagadējo kartes orientieru - novērtējums, balstoties uz priekšgala modulī iegūtajiem datiem. Atšķirībā no filtrēšanā bāzētām metodēm optimizācijā bāzētās metodes ir globālās optimizācijas metodes, kas optimizē visus iepriekšējos sistēmas stāvokļus - visas aģenta pozas jeb visu aģenta pārvietošanās trajektoriju un, iespējams, arī vi-

sus kartes orientierus -, nodrošinot precīzāku kartēšanu. Jāpiemin, ka abu kategoriju metodes ir iespējams vienlaikus izmantot vienā SLAM sistēmā, piemēram, koriģējot filtrēšanas metodes uzkrāto novirzi ar optimizācijā bāzētu metodi, atrodot un integrējot aģenta trajektorijas novērtējumā cilpu noslēgšanas [18].

Detalizētāk aplūkojot vienu no filtrēšanā bāzētām metodēm - daļiņu filtru -, svarīgs ir jēdziens posteriors (angliski: *posterior*). Posteriors ir varbūtības sadalījums, kas attēlo pašreizējo sistēmas stāvokļa novērtējumu. Posteriors tiek attēlots kā daļiņu kopa, kur katra daļiņa tiek uztverta kā sistēmas patiesā stāvokļa konkrēts minējums. Apvienojot attiecīgos minējumus kopās, daļiņu filtrs novērtē posteriora sadalījumu.

Cita filtrēšanā bāzētā metode - paplašinātais Kalmana filtrs - ir Kalmana filtra paveids, kas var tikt pielietots nelineārām sistēmām. Sistēmas stāvokli var attēlot kā vektoru (stāvokļa vektoru) $\overline{a}_k$, kur k ir laika posms attiecīgajam stāvoklim:

$$\overline{a}_k = f_{k-1}(\overline{a}_{k-1}) + w_{k-1} \tag{1.19}$$

$f_{k-1}(\overline{a}_{k-1})$ - stāvokļa propagators (angliski: *state propagator)*, kas atbilst vienmērīgai deterministiskai kustībai, kas būtu sagaidāma, ja nepastāvētu w_{k-1} radītie nejaušie traucējumi.

$w+k-1$ - procesa trokšņi. Procesa troķšņiem nav nobīdes (angliski: *bias*) $< w_{k-1} >= 0$ un ir kovariācija $Q_{k-1} \equiv < w_{k-1} w_{k-1}^T >$.

Attiecīgās sistēmas novērojumi (novērojumi par sistēmas stāvokli) laika posmā k tiek attēloti kā vektors m_k (mērījumu vektors):

$$m_k = h_k(\overline{a}_k) + \epsilon_k \tag{1.20}$$

$h_k(\overline{a}_k)$ - mērījumu vektors, kas tiktu iegūts, ja ϵ_k radītās nejaušās kļūdas nepastāvētu.

ϵ_k - mērījumu trokšņi. Mērījumu trokšņiem nav nobīdes $< \epsilon_k >= 0$ un ir kovariācija $V_k \equiv < \epsilon_k \epsilon_k^T >$.

Paplašinātā Kalmana filtra algoritmam ir divi posmi: prognozēšana un atjaunināšana. Prognozēšanas posmā tiek prognozēts stāvokļa vektors laika posmā k, izmantojot mērījumus (novērojumus) līdz laika posmam $k-1$:

$$a_k^{k-1} = f_{k-1}(a_{k-1}^{k-1}) = f_{k-1}(a_{k-1}) \tag{1.21}$$

Kovariācijas matrica (kļūdas matrica):

$$C_k^{k-1} = F_{k-1} C_{k-1} F_{k-1}^T + Q_{k-1} \tag{1.22}$$

$$F_{k-1} = \left(\frac{\delta f_{k-1}}{\delta a_{k-1}} \right) \tag{1.23}$$

Atjaunināšanas jeb filtrēšanas posmā tiek atjaunots prognozētais stāvokļa vektors, iekļaujot novērojumu laika posmā k, lai iegūtu optimālu novērtējumu stāvokļu vektoram laika posmā k. Atjaunoto stāvokļa vektoru var iegūt, izmantojot sekojošos vienādojumus:

$$a_k = a_k^{k-1} + K_k \left(m_k - h_k \left(a_k^{k-1} \right) \right) \tag{1.24}$$

$$K_k = C_k^{k-1} H_k^T \left(V_k + H_k C_k^{k-1} H_k^T \right)^{-1} \tag{1.25}$$

$$H_k = \left(\frac{\delta h_k}{\delta a_k^{k-1}} \right) \tag{1.26}$$

Kovariācijas matrica:

$$C_k = \left[\left(C_k^{k-1} \right)^{-1} + H_k^T G_k H_k \right]^{-1} \tag{1.27}$$

Optimizācijā bāzētu metožu gadījumā sistēmas stāvokļi tiek saglabāti stāvokļu grafā (vai, ja sistēmas stāvokļos neiekļauj orientierus, pozu grafā). Optimizācijas process sākas ar ierobežojumu identificēšanu starp jaunajiem novērojumiem un stāvokļu grafu. Iegūtie ierobežojumi pēc tam tiek izmantoti, lai precizētu sistēmas stāvokļus.

Šo metožu principu var aprakstīt sekojoši. Pieņem, ka $x = (x_1, ..., x_T)^T$ ir vektors, kur x_i apzīmē sistēmas stāvokli grafa mezglā i. z_{ij} ir mezglu i un j virtuālā novērojuma vidējais, bet Ω_{ij} - attiecīgā virtuālā novērojuma informācijas matrica. Virtuālais novērojums ir transformācija, kas maksimāli pārklāj i novērojumus ar j novērojumu. $\hat{z}_{ij}(x_i, x_j)$ ir virtuālā novērojuma prognoze, zinot mezglu x_i un x_j konfigurāciju (parasti virtuālā novērojuma prognoze ir relatīvā transformācija starp abiem mezgliem). Mērījuma z_{ij} logaritmisko iespējamību l_{ij} var izteikt kā:

$$l_{ij} \propto [z_{ij} - \hat{z}_{ij}(x_i, x_j)]^T \Omega_{ij} [z_{ij} - \hat{z}_{ij}(x_i, x_j)] \tag{1.28}$$

Funkciju, kas aprēķina atšķirību starp paredzēto novērojumu $\hat{z}_{ij}$ un patieso novērojumu z_{ij} izsaka kā:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j) \tag{1.29}$$

Optimizācijas mērķis ir atrast tādu mezglu x^* konfigurāciju, kas minimizē negatīvo logaritmisko iespējamību $F(x)$ visiem novērojumiem:

$$x^* = \arg\min_x F(x) \tag{1.30}$$

$$F(x) = \sum_{(i,j) \in C} e_{ij}^T \Omega_{ij} e_{ij} \tag{1.31}$$

Cilpu noslēgšanas modulis

Cilpu noslēgšanas modulim ir būtiska nozīme SLAM sistēmas uzkrātās novirzes samazināšanā. Sensoru mērījumu neprecizitātes dēļ SLAM sistēmas novērtētā aģenta trajektorija var novirzīties no patiesās, kā rezultātā aģents var atgriezties iepriekš kartētā vietā telpā, taču pēc

trajektorijas novērtējuma būt pavisam citā atrašanās vietā. Cilpu noslēgšanas moduļa uzdevums ir identificēt šādus iepriekš kartētus telpas apgabalus, konkrētāk, cilpu noslēgšanas modulim ir jāsasaista divi SLAM sistēmas kartes apgabali, kas atbilst vienam un tam pašam telpas apgabalam, bet kuru atrašanās vieta kartē nesakrīt jeb tie nepārklājas.

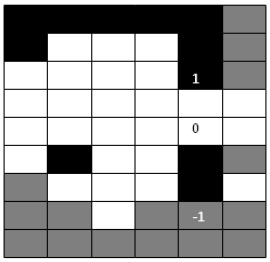
Neatkarīgi no SLAM sistēmā iegūto sensoru mērījumu veida cilpu noslēgšanas moduļa pamatideja ir optimāli salīdzināt pēdējos jeb tagadējos SLAM sistēmas novērojumus ar visiem iepriekšējiem SLAM sistēmas novērojumiem. Tiešā pieeja - iegūto sensoru mērījumu salīdzināšana ar iepriekš iegūtajiem sensoru mērījumiem - ir skaitļošanas ziņā neizdevīga, tāpēc cilpu noslēgšanas moduļa implementācijai tiek piedāvātas optimālākas metodes, piemēram, no sensoru mērījumiem izgūto iezīmju salīdzināšana starp tagadējo novērojumu un jau izveidoto karti. Alternatīva metode ir attēlu izgūšanas metode, kur atbilstības tiek meklētas starp tagadējo kameras attēlu un iepriekš uzņemtajiem kameras attēliem, salīdzinot no attēliem izgūtās iezīmes.

Diemžēl, salīdzinot divus līdzīgus telpas apgabalus, cilpu noslēgšanas modulis var saskarties ar tādu parādību kā uztveres kropļojums (angliski: *perceptual aliasing*). Uztveres kropļojuma gadījumā no sensoriem iegūtie dati (dēvēti par uztveres nospiedumu (angliski: *perceptual footprint*)) divos dažādos telpas apgabalos ir tik līdzīgi, ka cilpu noslēgšanas algoritms attiecīgos apgabalus nespēj atšķirt, nepareizi atzīmējot abus telpas apgabalus kā vienu un to pašu apgabalu. Rezultātā SLAM sistēmas aizmugures modulim tiek padota kļūdaina informācija par cilpu slēgumiem, kas var rezultēties ar SLAM sistēmas izveidotās kartes datu bojāšanu. Lai novērstu attiecīgo problēmu, tiek veikta cilpu slēgumu verifikācija [19].

Kartēšanas modulis

Kartēšanas moduļa funkcionalitāte ir kartes izveidošana. Vienas SLAM sistēmas ietvaros var tikt izveidotas arī vairākas kartes. Atkarībā no plānotā kartes pielietojuma izveidotā karte var būt retināta, daļēji retināta vai arī blīva. Atšķirība starp blīvu un retinātu karti ir tās detalizācijas pakāpē. Retinātas kartes satur tikai svarīgāko kartētās telpas informāciju, piemēram, orientierus, savukārt blīvas kartes ietver sevī detalizētu informāciju par telpas uzbūvi, respektīvi, blīvas kartes ir atmiņas ziņā apjomīgākas kartes.

Tipisks blīvas kartes piemērs ir režģkarte. Režģkaršu gadījumā telpiskie dati tiek attēloti režģī. Režģis telpu sadala tīklojuma šūnās - ģeometriskās figūrās, piemēram, kubos, starp kurām nav atstarpju vai pārklāšanās. Plaknes kartēšanas gadījumā tipisks režģkartes piemērs ir aizpildījuma režģkarte (angliski: *occupancy grid map*). Aizpildījuma režģkartē kartētā plakne ir vienmērīgi sadalīta kongruentos jeb vienādos kvadrātos. Šajā gadījumā karti var uztvert kā attēlu vai matricu, kur katrs matricas elements jeb katra tīklojuma šūna (x, y) satur sevī informāciju par plaknes attiecīgās teritorijas aizpildījuma varbūtību. Jo lielāka ir aizpildījuma varbūtība, jo lielāka ir iespēja, ka attiecīgajā plaknes teritorijā ir šķērslis, kuru aģents nespēj pārvarēt un kuru ir nepieciešams apiet, piemēram, siena vai upe. Aizpildījuma varbūtības vērtība parasti ir daļskaitlis robežās no 0 (šūna ir brīva) līdz 1 (šūna ir aizpildīta), bet vērtību -1 piešķir šūnām, kuru aizpildījuma varbūtība nav zināma.



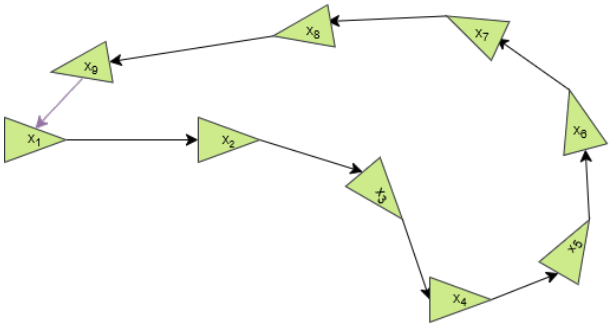
Att. 1.3: **Aizpildījuma režģkarte**

Vēl viens režģkaršu piemērs ir pacēluma kartes (angliski: *elevation maps*). Šajā gadījumā katra tīklojuma šūna satur vērtību, kas apraksta attiecīgās plaknes teritorijas pacēlumu attiecībā pret kādu atskaites punktu.

Telpas kartēšanas gadījumā rezultātā iegūtā režģkarte tiek saukta par vokseļu režģi (angliski: *voxel grid*). Vokselis ir trīsdimensionāls pikseļa analogs, attiecīgi telpa tiek vienmērīgi sadalīta kongruentos kubos. Arī vokseļu režģa tīklojuma šūnas var saturēt informāciju par aizpildījuma varbūtībām attiecīgajās telpas teritorijas, kā arī citu konkrētajai SLAM sistēmai būtisku informāciju.

Kā retinātas kartes piemēru var minēt iezīmju kartes. Iezīmju kartēs telpiskie dati tiek attēloti kā iezīmju kopa, kur katra iezīme satur sevī konkrētajai SLAM sistēmai būtisku informāciju. Šādas iezīmes var būt punkti, līnijas, plaknes vai objekti. Tā kā iezīmju kartes ir retinātas, salīdzinot ar režģkartēm, tām ir nepieciešami mazāki atmiņas resursi.

Vēl viens retinātas kartes piemērs ir stāvokļu grafi. Stāvokļu grafos tiek kartēti visi iepriekšējie un pašreizējie SLAM sistēmas stāvokļi, veidojot ķēdei līdzīgu datu struktūru, kurā sistēmas stāvokļi - aģenta pozas un orientieri - ir savstarpēji saistīti jeb savienoti ar dažādiem ierobežojumiem. Ja sistēmas stāvokļos netiek iekļauti orientieri, bet gan tikai aģenta pozas, stāvokļu grafu dēvē par pozu grafu. Stāvokļu grafā un pozu grafā aģenta pozas ir savstarpēji savienotas ar nobrauktā attāluma mērījumiem un papildus ierobežojumiem starp patvaļīgām pozām cilpu slēgumu gadījumā.

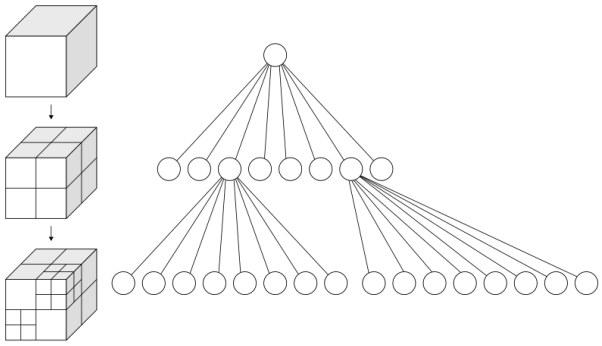


Att. 1.4: **Pozu grafs**

Lai gan stāvokļu grafi nesatur telpiskos datus, tie tiek pielietoti SLAM sistēmas aizmugures modulī SLAM sistēmas stāvokļa novērtēšanai. Optimizējot stāvokļu grafus, piemēram, atrodot tādu aģenta pozu (mezglu) konfigurāciju, kas minimizē mazāko kvadrātu kļūdu (angliski: *least squares error*) pāri visiem ierobežojumiem (šķautnēm) attiecīgajā stāvokļu grafā, tiek

iegūts precīzāks aģenta pārvietošanās trajektorijas novērtējums.

Kā piemēru daļēji retinātām kartēm var minēt hierarhiskās datu struktūras. Hierarhisku datu struktūru gadījumā telpiskie dati tiek attēloti kokiem līdzīgās struktūrās. Populārs šādas datu struktūras piemērs ir oktantkoki (angliski: *octree*) - koki, kur katram iekšējam mezglam ir tieši astoņi bērni. Oktantkoku gadījumā telpa tiek rekursīvi sadalīta astoņās oktantēs. Kartējot plakni, oktantkokiem analoga divdimensionāla datu struktūra ir kvadrantkoki (angliski: *quadtree*), kur katram iekšējam mezglam ir tieši četri bērni. Oktantkokos un kvadrantkokos glabātās informācijas apstrādi, piemēram, lapas atrašanu, kas atbilst noteiktam punktam (x, y), ir iespējams veikt, izmantojot pamata koku operācijas.



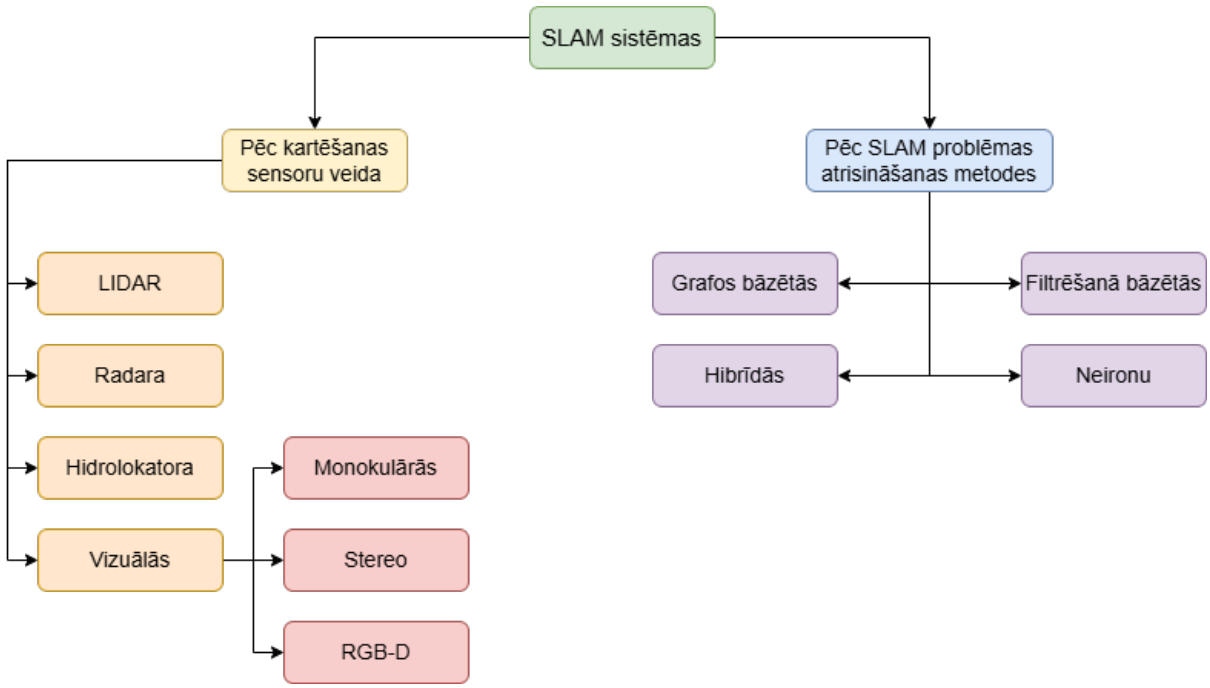
Att. 1.5: **Oktantkoks [20]**

1.2.2. SLAM sistēmu kategorizācija

SLAM sistēmu daudzveidīgo klāstu ir iespējams kategorizēt gan pēc izmantoto kartēšanas sensoru veida, gan pēc SLAM problēmas atrisināšanas metodes. Abas kategorizācijas ir apskatāmas 1.6 attēlā. Protams, eksistē arī citas kategorizācijas metodes, piemēram, pēc kartējamās telpas veida vai aģentu skaita, taču šajā nodaļā tās netiks aplūkotas, tā kā attiecīgās nodaļas mērķis ir dot vispārīgu ieskatu SLAM sistēmu daudzveidībā, izklāstot klasiskās SLAM sistēmu kategorizācijas metodes.

SLAM sistēmas pēc izmantoto kartēšanas sensoru veida

SLAM sistēmu kategorizācijā pēc izmantoto kartēšanas sensoru veida par kartēšanas sensoru tiek dēvēts sensors, kas iegūst datus par kartējamo apkārtni, respektīvi, šī sensora dati tiek izmantoti par pamatu kartes veidošanai. Kartēšanas sensors var būt, piemēram, kamera, LIDAR sensors, radars vai hidrolokators. Galvenā atšķirība starp dažādu kategoriju SLAM sistēmām šajā iedalījumā ir ievades datu formātā un to sākotnējā apstrādē. Radara SLAM sistēmu gadījumā ievadē tiek saņemti radara dati, piemēram, četru dimensiju attēlveidošanas radara punktu mākoņi, kas ietver sevī mērījumus četrās dimensijās: attāluma, relatīvā ātruma, azimuta un pacēluma dimensijā [21]. LIDAR SLAM sistēmu gadījumā kartēšana un lokalizācija notiek, ievadē saņemot LIDAR sensora iegūtos punktu mākoņus un veicot skenējumu salīdzināšanu. Savukārt vizuālās SLAM sistēmās ievadē tiek saņemta vizuālā informācija jeb kameras attēli, no kuriem



Att. 1.6: **SLAM sistēmu kategorizācija**

tiek iegūtas iezīmes apkārtējās vides kartēšanai un aģenta kustības novērtēšanai.

Vizuālās SLAM sistēmas ir iespējams detalizētāk iedalīt monokulārās, stereo un RGB-D SLAM sistēmās. Monokulāro SLAM sistēmu gadījumā attēli tiek iegūti no vienas kameras. Tā kā kameras attēli neietver sevī dziļuma informāciju, šai pieejai ir būtisks trūkums trīs dimensiju SLAM sistēmās, kurās tiek kartēta telpa. Dziļuma informācijas iegūšanai, protams, ir iespējams izmantot tādas metodes kā "forma pēc ēnojuma" (angliski: *shape from shading* jeb SFS) vai "forma pēc tekstūras" (angliski: *shape from texture* jeb SFT), kas balstās uz attēla analīzi pēc kāda konkrēta parametra (ēnojuma vai spilgtuma, tekstūras, kontūras, kustības vai kāda cita parametra), taču tām ir savi ierobežojumi. Piemēram, "forma pēc ēnojuma" metode ir skaitļošanas sarežģītības ziņā salīdzinoši izdevīga metode, taču, lai iegūtu precīzus dziļuma novērtējumus, tai ir nepieciešams nemainīgs apgaismojums un fiksēta kameras pozīcija, kas tipiskos SLAM sistēmu pielietojumos nav nodrošināms. "Forma pēc kustības" (angliski: *shape from motion* jeb SFM) metodes gadījumā dziļums tiek novērtēts, izmantojot virkni attēlu, kas uzņemti no dažādiem leņķiem. Attiecīgi kamerai ir nepieciešams pārvietoties telpā, kas tipiskos SLAM sistēmu pielietojumos arī notiek, taču šīs metodes trūkums ir nespēja novērtēt dziļumu pēc viena attēla uzņemšanas vai miera stāvoklī, kad uzņemtajos attēlos nav novērojamas izmaiņas. Alternatīvi dziļuma noteikšanai ir iespējams izmantot mašīnmācīšanās vai dziļās mašīnmācīšanās metodes, kas spēj sniegt precīzākus dziļuma novērtējumus dažādos attēlu uzņemšanas apstākļos, taču kurām ir nepieciešamas lielas datu kopas modeļu apmācībai un kuru skaitļošanas sarežģītība var būt ievērojami augstāka, salīdzinot ar tradicionālajām dziļuma novērtēšanas metodēm [22].

Stereo SLAM sistēmu gadījumā ievadē tiek saņemti attēli no divām dažādās pozīcijās vai leņķos izvietotām kamerām. Izmantojot iegūtos kameru attēlus, var noteikt telpas dziļumu gan kustībā, gan miera stāvoklī, gan no viena attēlu pāra. Telpas dziļums tiek aprēķināts, balstoties uz attālumu starp abām kamerām un nevienādību starp abiem uzņemtajiem attēliem, atdarinot

to, kā cilvēka redze uztver telpas dziļumu. Jāņem vērā, ka stereo SLAM gadījumā skaitļošanas sarežģītība ir ievērojami augsta, kas ir trūkums zema enerģijas patēriņa sistēmām.

RGB-D SLAM sistēmu gadījumā vienlaicīgi tiek iegūts atsevišķs krāsains attēls un attiecīgā attēla katra pikseļa dziļums. Krāsainais attēls tiek iegūts, izmantojot kameru, bet katra pikseļa dziļums tiek iegūts, izmantojot dziļuma sensoru. Dziļuma sensora dati var tikt iegūti, piemēram, balstoties uz infrasarkanās strukturētās gaismas struktūru vai izstarotās gaismas lidojuma laiku. RGB-D SLAM ir augsta precizitāte iekštelpās, taču āra apstākļos tās precizitāti būtiski ietekmē izmaiņas apgaismojumā un kustības izplūdums.

SLAM sistēmas pēc SLAM problēmas atrisināšanas metodes

Kategorizējot SLAM sistēmas pēc SLAM problēmas atrisināšanas metodes, galvenā atšķirība starp dažādām kategorijām ir SLAM sistēmas aizmugures moduļa implementācijā. Grafos bāzētu SLAM sistēmu gadījumā aizmugures modulis tiek implementēts, izmantojot optimizācijā bāzētās metodes un stāvokļu vai pozu grafus. Savukārt filtrēšanā bāzētās SLAM sistēmās aizmugures moduļa implementācijā tiek izmantotas filtrēšanā bāzētās metodes. Neironu SLAM sistēmās SLAM problēmas atrisināšanai tiek izmantoti neironu tīkli. Neironu tīklu izmantošana šādās sistēmās neierobežojas tikai ar aizmugures moduļa implementāciju, tas ir, neironu tīkli var tikt izmantoti arī jebkura cita SLAM sistēmas moduļa implementācijai. SLAM sistēmas, kuras vienlaikus izmanto vairākas iepriekšminētās metodes dēvē par hibrīdām SLAM sistēmām.

1.3. Kameras kalibrēšana

Gadījumos, kad SLAM sistēmas ievadē tiek saņemti kameras attēli, būtisks solis sekmīgas SLAM sistēmas darbības realizēšanai ir kameras kalibrēšanai [23, 24]. Kameras kalibrēšana ir kameras parametru - kropļojuma koeficientu (angliski: *distortion coefficients*), kameras raksturlielumu (angliski: *camera intrinsics*) un kameras ārējo parametru (angliski: *camera extrinsics*) - noteikšana. Nosakot kameras parametrus, SLAM sistēmai ir iespējams koriģēt kameras objektīva kropļojumus un noteikt kameras pozu, nodrošinot korektu attēla punktu projekciju SLAM sistēmas kartē.

Attiecību starp kameras attēla pikseli p un telpas punktu t_W var izteikt kā vienādojumu $sp = AT_{CW}t_W$, kur s ir mērogošanas koeficients (angliski: *scaling factor*), A ir kameras raksturlielumu matrica, bet T_{CW} ir pārejas matrica (ārējie parametri) no telpas atskaites sistēmas W uz kameras atskaites sistēmu C:

$$s \cdot \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s' & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{CW} & t_{CW} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{t_W} \\ y_{t_W} \\ z_{t_W} \\ 1 \end{bmatrix} \tag{1.32}$$

1.3.1. Kropļojuma koeficienti

Kameras kropļojuma koeficientus ir iespējams reprezentēt kā piecu vērtību vektoru $(k_1, k_2, p_1, p_2, k_3)$, kur k_1, k_2, k_3 ir radiālā kropļojuma koeficienti, bet p_1, p_2 ir tangenciālā kropļojuma koeficienti. Gadījumā, kad radiālais kropļojums nav ievērojami liels, koeficientu k_3 mēdz neiekļaut.

Radiālā kropļojuma koeficienti reprezentē radiālo kropļojumu, kas rodas tad, kad gaismas stari objektīva malu tuvumā ir ar lielāku lieci nekā objektīva optiskajā centrā. Radiālā kropļojuma kropļotos attēla pikseļus (x_d, y_d) var izteikt sekojoši:

$$x_d = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{1.33}$$

$$y_d = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{1.34}$$

$$r^2 = x^2 + y^2$$

(x, y) - attēla pikseļi bez kropļojuma.

Tangenciālā kropļojuma koeficienti reprezentē tangenciālo kropļojumu, kas rodas tad, kad kameras objektīvs nav paralēls attēla plaknei. Tangenciālā kropļojuma kropļotos attēla pikseļus (x_d, y_d) var izteikt kā:

$$x_d = x + (2p_1xy + p_2(r^2 + 2x^2)) \tag{1.35}$$

$$y_d = y + (p_1(r^2 + 2y^2) + 2p_2xy) \tag{1.36}$$

$$r^2 = x^2 + y^2$$

(x, y) - attēla pikseļi bez kropļojuma.

1.3.2. Kameras raksturlielumi

Kameras raksturlielumus A var reprezentēt kā matricu:

$$A = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.37}$$

(f_x, f_y) - fokusa attālums (angliski: *focal length*). Raksturo kameras attālumu no attēla plaknes.

(c_x, c_y) - optiskais centrs (angliski: *optical center*).

s' – slīpuma koeficients.

2. METODE

2.1. Datu apvienošanas metodes projektējums

Darba ietvaros implementēto datu apvienošanas metodi var iedalīt vairākos secīgos posmos:

1. Datu priekšapstrādē. Šajā posmā lidojumā iegūtie dati tiek pārveidoti, lai atbilstu SLAM sistēmas ievadei.
2. Datu buferizācijā. Pārveidotie dati tiek saglabāti buferī pirms to tālākās apstrādes SLAM sistēmā. Buferizācijas nolūks ir nodrošināt SLAM sistēmas darbības neatkarību no lidojumā iegūtu datu saņemšanas ātruma.
3. Lokālajā kartēšanā. Lokālā kartētāja ietvaros tiek izveidotas tuvākās apkārtnes kartes, kas vēlāk tiek integrētas globālajā (visas kartētās teritorijas) kartē. Lokālajā kartētājā tuvākās apkārtnes kartes tiek izveidotas tikai no lidojumā iegūtajiem datiem.
4. Globālajā kartēšanā. Globālā kartētāja ietvaros lokālās (lokālā kartētāja izveidotās) kartes tiek integrētas globālajā kartē, konkrētāk, globālā vokseļu kartē un punktu mākonī.

Vienlaicīgās lokalizācijas un kartēšanas process ir ciklisks, respektīvi, minētie posmi, izņemot datu priekšapstrādi, tiek atkārtoti līdz programmas darbības nobeidzei (parasti lietotāja ierosinātai). Turpmākajās nodaļās tiek sniegts detalizētāks katra posma implementācijas apraksts, savukārt 2.1.1 apakšnodaļā tiek aprakstīts satvars, uz kura pamata tika veidota SLAM sistēma.

2.1.1. ROS 2 integrācija

Savstarpējās komunikācijas nodrošināšanai starp atsevišķajiem SLAM sistēmas procesiem un lidojumā iegūtiem datiem tiek izmantots ROS 2 [25] satvars. ROS 2 ir atvērtā pirmkoda satvars, kas iekļauj sevī dažādas programmatūras bibliotēkas un rīkus robotu programmatūru izstrādei. Numurs "2" apzīmē satvara jaunāko versiju, kas aizstāj ROS satvara iepriekšējo versiju ROS 1.

ROS gadījumā sistēma sastāv no mezgliem (angliski: *nodes*), kur katrs mezgls ir atbildīgs par kādu noteiktu sistēmas moduli, piemēram, robota motoru kontroli. Datu pārraide starp mezgliem notiek, izmantojot tematus (angliski: *topics*), servisus (angliski: *services*), darbības (angliski: *actions*) vai parametrus. Darbā aprakstītās metodes ietvaros datu pārraidei tiek izmantoti ROS temati un servisi.

ROS temati ir vienkāršākais datu pārraides variants starp mezgliem. ROS tematu gadījumā pārraidītās datu vienības tiek dēvētas par ziņojumiem. Lai mezgls spētu nosūtīt jeb publicēt

ziņojumu pārējiem mezgliem caur kādu konkrētu tematu, tam ir nepieciešams izveidot attiecīgajam tematam publicētāju (angliski: *publisher*). Savukārt, lai mezgls spētu saņemt ziņojumu no kāda temata, tam ir nepieciešams izveidot attiecīgajam tematam abonentu (angliski: *subscriber*). Ziņojumi, kas tiek publicēti konkrētajā tematā, ir pieejami visiem temata abonentiem. Metodes ietvaros ROS temati tiek izmantoti lidojumā iegūtu datu ievadei SLAM sistēmā.

ROS servisu gadījumā, lai mezgls spētu saņemt datus jeb atbildes ziņojumus caur kādu konkrētu servisu, tam ir nepieciešams izveidot attiecīgajam servisam klientu. Savukārt mezglam, kas veic atbildes ziņojumu sūtīšanu caur servisu, ir nepieciešams izveidot attiecīgā servisa serveri. Atšķirībā no ROS tematiem ROS servisu gadījumā dati tiek pārraidīti tikai tad, kad servisa klients nosūta pieprasījuma ziņojumu. ROS servisiem var būt vairāki klienti, taču tikai viens serveris. Metodes ietvaros ROS servisi tiek izmantoti SLAM sistēmas vokseļu kartes ieguvei turpmākajai SLAM sistēmas veiktspējas novērtēšanai.

2.2. Datu priekšapstrāde

Pirms lidojumā iegūtie dati tiek padoti SLAM sistēmas ievadei, ir nepieciešams veikt to priekšapstrādi, lai nodrošinātu datu atbilstību noteiktajam ievades datu formātam.

Lidojumā iegūtie dati sastāv no trīs atsevišķām komponentēm: bezpilota lidaparāta pozas, kameras attēliem un apkārtējās vides absolūtā augstuma punktiem ar tiem atbilstošajiem klašu vektoriem.

2.2.1. Poza

Bezpilota lidaparāta atrašanās vieta telpā jeb poza tiek reprezentēta ar vektoru $p_L = (x_L, y_L, z_L, q_{x_L}, q_{y_L}, q_{z_L}, q_{w_L})$, kur $t_L = (x_L, y_L, z_L)$ ir bezpilota lidaparāta pozīcija un kur $q_L = (q_{x_L}, q_{y_L}, q_{z_L}, q_{w_L})$ ir bezpilota lidaparāta orientācija telpā, izteikta kvaternionos. Bezpilota lidaparāta poza ir izteikta relatīvi lidojuma sākumpunktam. Tā kā darba ietvaros lidojuma atskaites sistēma sakrita ar SLAM sistēmas kartēšanas atskaites sistēmu, pozas priekšapstrāde nebija nepieciešama. Ja atskaites sistēmas nesakristu, pozas priekšapstrādē poza tiktu pārveidota no lidojuma atskaites sistēmas uz kartēšanas atskaites sistēmu, izmantojot 1.7 vienādojumu.

Attiecīgi bezpilota lidaparāta orientācija q_L tiktu pārveidota no kvaterniona reprezentācijas uz rotāciju matricas reprezentāciju [26]:

$$R(q) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{2.1}$$

Vienības kvaternionu gadījumā $q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1$. Attiecīgi reprezentācijas pārveidojumu var izteikt arī sekojoši:

$$R(q) = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{2.2}$$

No iegūtās rotāciju matricas $R(q_L)$ un bezpilota lidaparāta pozīcijas t_L tiktu iegūta pārejas matrica T_L. Lidojuma sākumpunkta poza (lidojuma atskaites sistēmas L centrs) kartēšanas atskaites sistēmā K tiktu reprezentēta ar pārejas matricu T_{KL}. Bezpilota lidaparāta poza p_K kartēšanas atskaites sistēmā būtu iegūstama ar vienādojumu $p_K = T_{KL}T_L$.

Respektīvi, bezpilota lidaparāta orientācija būtu reprezentēta kā rotāciju matrica $R(q_K)$:

$$R(q_K) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{2.3}$$

Lai iegūtu kvaterniona reprezentāciju bezpilota lidaparāta orientācijai, būtu jāpielieto sekojošie pārveidojumi [26]:

$$\begin{cases} q_x = \frac{R_{32} - R_{23}}{4q_w} \\ q_y = \frac{R_{13} - R_{31}}{4q_w} \\ q_z = \frac{R_{21} - R_{12}}{4q_w} \\ q_w = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \end{cases} \tag{2.4}$$

Bezpilota lidaparāta poza SLAM sistēmai tiek publicēta caur */mapper_node_elevation/pose* tematu. Temata nosaukums ir modificējams SLAM sistēmas konfigurācijas datnē. Publicētā ziņojuma veids ir *geometry_msgs/msg/TransformStamped* [27].

2.2.2. Kameras attēli

No bezpilota lidaparāta kameras tiek iegūti 192x128 krāsaini attēli. Kameras attēliem priekšapstrāde nav nepieciešama. Tie tiek publicēti SLAM sistēmai caur */camera/image_raw* tematu. Temata nosaukums ir modificējams SLAM sistēmas konfigurācijas datnē. Publicētā ziņojuma veids ir *sensor_msgs/msg/Image* [28].

2.2.3. Augstuma punkti

Absolūtā augstuma punkti ar tiem atbilstošajiem klašu vektoriem no bezpilota lidaparāta tiek saņemti kā 2x15 matrica. Pirmie 15 matricas elementi ir absolūtā augstuma mērījumi. Mērījumu rezolūcija ir 0,2 metri mērāmās telpas platumā un garumā. Otrie 15 matricas elementi ir mērījumiem atbilstošie klašu vektori. Klašu vektori ir 5x1 vektori, kuri ietver sevī vienizcēluma kodējumā iekodētus datus par atbilstošo mērījumu piederību šīm piecām virsmas klasēm,

konkrētāk, asfaltam, cietiem dubļiem, zālei, ūdenim un objektam.

Lai lidojumā iegūtie dati atbilstu SLAM sistēmas noteiktajam ievades datu formātam, iegūtie absolūtā augstuma mērījumi tiek pārveidoti par absolūtā augstuma punktiem (algoritms Nr. 1). Klašu vektoriem priekšapstrāde nav nepieciešama.

Algoritms Nr. 1 Augstuma mērījumu pārveidošana par augstuma punktiem

Require: $r = 0.2,\ t_{DS} = (x_{DS}, y_{DS}, z_{DS}),\ R_{DS},\ o_S\_arr$
1: $x\_arr = \{x_i \mid x_i = -1.4 + ri; \quad i \in \{0, 1, \ldots, 14\}\}$
2: $t_D\_arr \leftarrow [\,]$
3: **for** $i = 0$ **to** 14 **do**
4: $\quad t \leftarrow [x_{DS} + o_S\_arr[i],\ y_{DS} + x\_arr[i],\ z_{DS}]$
5: $\quad t_D \leftarrow R_{DS}t$
6: $\quad t_D\_arr \leftarrow t_D\_arr \cup [t_D]$
7: **end for**
8: **return** t_D_arr

Absolūtā augstuma punkti ar tiem atbilstošajiem klašu vektoriem SLAM sistēmai tiek publicēti caur */elevation* tematu. Tematam ir individuāls ziņojuma veids - *ElevationMessage*. Ziņojums sastāv no sekojošām sastāvdaļām:

- *std_msgs/Header* header - ziņojuma galvenes;
- uint32 size - augstuma punktu skaita (šajā gadījumā 15);
- uint32 step - klašu vektora klašu skaita (šajā gadījumā 5);
- *geometry_msgs/Point[]* points - augstuma punktiem;
- uint8[] segmentation - klašu vektoriem.

2.3. Datu buferizācija

Tā kā SLAM sistēmai ir specifiska datu apstrādes frekvence, ievades datu saglabāšanai pirms to apstrādes ir nepieciešams datu buferis. Datu buferis saglabā lidojumā iegūtos datus, novēršot datu zudumu un nodrošinot sistēmas neatkarību no iegūstamo datu frekvences.

Tā kā SLAM sistēmā jau bija pieejamas klases pozu un attēlu buferizācijai, bezpilota lidaparāta pozas un kameras attēli tiek saglabāti attiecīgo klašu objektos. Attēlu buferizācijas klases objektu parametru nolasīšana tika vispārināta, turpmāk veicot parametru nolasīšanu no SLAM sistēmas konfigurācijas datnes. SLAM sistēmas konfigurācijas datne tika papildināta ar parametriem attiecīgajiem klašu objektiem.

Absolūtā augstuma punktu buferizācijai tika izveidota atsevišķa klase ElevationBuffer. Tāpat kā citas buferizācijas klases, arī *ElevationBuffer* klase ir ROS mezgls. Attiecīgajam ROS mezglam ir izveidots abonents tematam */elevation*. Iegūtie absolūtā augstuma punkti ar tiem atbilstošajiem klašu vektoriem tiek saglabāti kā divu virkņu pāri. Pirmā virkne satur augstuma punktus, savukārt otrā virkne satur klašu vektorus.

2.4. Lokālā kartēšana

Lokālajā kartētājā tiek izveidotas apkārtējās vides lokālās kartes, kas vēlāk tiek apvienotas kopā globālajā kartētājā. Lokālā kartēšana noris ar noteiktu frekvenci – 0,5 Hz jeb reizi 2 sekundēs. Vispirms no datu buferiem tiek iegūti kameras attēli un absolūtā augstuma punkti ar tiem atbilstošajiem klašu vektoriem. Augstuma punkti tiek ievietoti lokālās kartes oktantkokā. Pēc tam tiek apstrādāti kameras attēli (ne vairāk par 10 attēliem), projicējot attēlu semantikas vektorus uz lokālās kartes punktiem. Projicēšanas rezultāti kopā ar augstuma punktu klašu vektoriem tiek saglabāti lokālās kartes ierakstos, kas vēlāk tiek izmantoti vokseļu kartes izveidošanai. Lokālās kartes ierakstu datu struktūra tika modificēta, iekļaujot klašu vektorus. Tāpat tika modificēta ierakstu ievietošanas metode, iekļaujot klašu vektoru ievietošanu. Kad bezpilota lidaparāta nolidotās trajektorijas garums pārsniedz 30 metrus, tagadējā lokālā karte tiek padota globālajam kartētājam, savukārt lokālajā kartētājā tiek izveidota jauna lokālā karte.

Algoritms Nr. 2 Lokālais kartētājs

Require: $submap,\ e\_buf,\ i\_buf, p\_buf, global\_mapper$

$last\_e \leftarrow$ pēdējā apstrādājamā e_buf elementa laika zīmogs nanosekundēs
$last\_i \leftarrow$ pēdējā apstrādājamā i_buf attēla laika zīmogs nanosekundēs
$c\_map \leftarrow \{\}$
$e\_elem \leftarrow \{e \mid e \in e\_buf; \quad e.stamp \leq last\_e\}$
$i\_elem \leftarrow \{i \mid i \in i\_buf; \quad i.stamp \leq last\_i\}$
for e **in** e_elem **do**
  $T_e \leftarrow p\_buf.at(e.stamp)$
  $submap.octree.insert(e.data.first,\ T_e)$
  for $i = 0$ **to** $\mid e.data.second \mid$ **do**
    $T_{Le} \leftarrow submap.T^{-1} \cdot T_e$
    $e_L \leftarrow T_{Le}.R \cdot e.data.first[i] + T_{Le}.t$
    $k \leftarrow submap.octree.T\_to\_key(e_L)$
    $c\_map \leftarrow c\_map \cup \{< k,\ e.data.second[i] >\}$
  end for
end for
$req \leftarrow RenderingRequest()$
$req.points \leftarrow submap.octree.points$
for $i = 0$ **to** 10 **do**
  if $i\_buf[i] = \emptyset$ **then**
    break
  end if
  $i \leftarrow i\_elem[i]$
  if $p\_buf.at(i.stamp) \neq \emptyset$ **then**
    $T_i \leftarrow p\_buf.at(i.stamp)$

$req.T_c \leftarrow req.T_c \cup [submap.T^{-1} \cdot T_i]$
$req.images \leftarrow req.images \cup [i.image]$
end if
end for
if $\mid req.T_c \mid > 0$ **then**
$req\_result \leftarrow submap.projector.render(req)$
$records \leftarrow [\,]$
$k\_arr \leftarrow [\,]$
for $i = 0$ **to** $\mid req\_result \mid$ **do**
$k \leftarrow req\_result[i].key$
if $c\_map[k] \neq \emptyset$ **then**
$records \leftarrow records \cup [Record(req\_result[i], c\_map[k])]$
$k\_arr \leftarrow k\_arr \cup [k]$
else
$records \leftarrow records \cup [Record(req\_result[i])]$
end if
end for
for $< k, c >$ **in** c_map **do**
if $k\_arr.find(k) = \emptyset$ **then**
$records \leftarrow records \cup [Record(c)]$
end if
end for
$submap.records.insert(records)$
end if
if $submap.path > 30$ **then**
$s \leftarrow submap$
$submap \leftarrow Submap()$
$global\_mapper.push(s)$
end if

Alternatīvā pieeja lokālā kartētāja implementācijai SLAM sistēmā būtu lokālo karšu izveidošana ārpus SLAM sistēmas. Šādā gadījumā SLAM sistēmā nebūtu jāiekļauj arī lidojumā iegūto datu buferizācija, attiecīgi pirmie trīs metodes posmi tiktu veikti ārpus SLAM sistēmas. SLAM sistēma ievadē saņemtu lokālās kartes, kuras uzreiz tiktu apstrādātas globālajā kartētājā un integrētas globālajā kartē. Šāda pieeja samazina SLAM sistēmas noslodzi, kā arī sniedz iespēju SLAM sistēmas lietotājam izvēlēties savam datu formātam[1] pielāgotu lokālo kartētāju, taču šādā gadījumā, ja nu vienīgi SLAM sistēma netiek piedāvāta kopā ar jau implementētu (atsevišķu) lokālo kartētāju, lokālā kartētāja implementēšana kļūst par lietotāja atbildību. Respektīvi, lietotājam ir jānodrošina lokālā kartētāja lokālo karšu atbilstība SLAM sistēmas lokālo

[1] Ārpus šī darba problēmas definīcijas.

karšu datu tipam un formātam. Ja lokālās kartes tiek pārraidītas caur ROS tematu, visticamāk būs nepieciešams izveidot pielāgotu ziņojumu karšu datu pārraidei. Pielāgota ziņojuma izveidei nepieciešams modificēt SLAM sistēmas programmatūru, konkrētāk, to pakotni, kas satur pielāgotos ROS ziņojumus un servisus. Tāpat globālajā kartētājā jāimplementē lokālo karšu apstrāde, tas ir, pārveidošana par SLAM sistēmas lokālajām kartēm pirms to turpmākās integrācijas globālajā kartē.

Lokālā kartētāja algoritms redzams agloritmā Nr. 2.

2.5. Globālā kartēšana

Globālajā kartētājā tiek izveidota globālā karte: globāla vokseļu karte un punktu mākonis. Lai integrētu lidojumā iegūtos datus SLAM sistēmas globālajā kartē, vispirms no lokālā kartētāja tiek iegūta lokālā karte. Kad lokālā karte ir iegūta, tiek pārbaudīts, vai lokālajā kartē kartētā teritorija jau nav iekļauta globālajā kartētāja, citas lokālās kartes integrācijas laikā. Ja tā ir, tagadējo lokālo karti apvieno ar iepriekšējo lokālo karti. Tagadējo lokālo karti ievieto lokālo karšu virknē turpmākajai salīdzināšanai. Tagadējās lokālās kartes ievietošanu globālajā kartē veic SLAM sistēmā jau eksistējoša metode.

Globālā kartētāja saņemtās lokālās kartes apstrādes algoritms redzams algoritmā Nr. 3.

Algoritms Nr. 3 Globālais kartētājs

Require: $submap$, $submaps\_by\_key$, $voxel\_map$, $pointcloud$, $publisher$
  $k \leftarrow T\_to\_key(submap.T)$
  if $submaps\_by\_key[k] \neq \emptyset$ **then**
    $s \leftarrow submaps\_by\_key[k]$
    $v\_arr \leftarrow \{v \mid v \in voxel\_map \ \wedge v \in s\}$
    for v **in** v_arr **do**
      $v.erase(s)$
    end for
    $pointcloud.erase(s)$
    $submaps\_by\_key.erase(s)$
    $T_{K_{submap}K_s} \leftarrow submap.T^{-1} \cdot s.T$
    $s.T \leftarrow submap.T \cdot T_{K_{submap}K_s}$
    $submap.merge(s)$
  end if
  $submaps\_by\_key[k] \leftarrow submap$
  $render\_submap(submap)$
  $publisher.publish(pointcloud)$

2.6. Rezultātu novērtēšana

2.6.1. Datu ieguve

Metodes novērtēšanai nepieciešamie dati tika iegūti no imitētas vides jeb simulācijas, kas tika izveidota, izmantojot Webots [29] simulatoru. Tā kā Webots simulators nodrošina simulāciju integrācijas iespējas ar ROS 2, no simulācijas iegūto datu priekšapstrādi un ievietošanu ROS ziņojumos (implementētās metodes pirmo posmu) bija iespējams veikt jau simulācijas darbības laikā, uzreiz pēc datu vienību saņemšanas, rezultātā iegūstot SLAM sistēmas ievadei derīgus lidojumā iegūtus datus. Ievades datu formāta aprakstu skatīt 2.2 nodaļā "Datu priekšapstrāde".

Bezpilota lidaparāta poza relatīvi lidojuma atskaites sistēmai tika iegūta, izmantojot simulācijas *GPS* [30] un *InertialUnit* [31] mezglus. No *GPS* mezgla tika iegūta lidaparāta pozīcija (x, y, z), savukārt no *InertialUnit* mezgla tika iegūta lidaparāta orientācija kvaternionos $q = (q_x, q_y, q_z, q_w)$. Tā kā mezglu konfigurācijā abiem mezgliem netika nodefinēts iegūstamo vērtību troksnis (angliski: *noise*), iegūtās vērtības atbilda patiesajai bezpilota lidaparāta atrašanās vietai telpā. Iegūtā pozīcija un orientācija tika apvienotas vienā ROS ziņojumā un publicētas atbilstošajam ROS tematam ar frekvenci 62,5 Hz jeb vienu ziņojumu 16 milisekundēs.

Kameras attēli no simulācijas tika iegūti tieši, respektīvi, no simulācijas tie jau tika saņemti ROS ziņojumu veidā. Lai nodrošinātu fiksētu kameras attēlu ziņojumu plūsmu, tie tika publicēti ar fiksētu frekvenci - 20 Hz -, kas atbilstu vienam kameras attēlam 0,05 sekundēs.

Absolūtā augstuma punkti ar tiem atbilstošajiem klašu vektoriem tika iegūti no simulācijas, izmantojot *RangeFinder* [32] un *Camera* [33] mezglus. Absolūtā augstuma mērījumi tika iegūti no *RangeFinder* mezgla. Absolūtā augstuma mērījumi tika pārveidoti par absolūtā augstuma punktiem, izmantojot algoritmā Nr. 1 aprakstīto datu priekšapstrādes metodi.

No *Camera* mezgla tika iegūti atbilstošie segmentācijas attēli. Segmentācijas attēli satur katram augstuma punktam atbilstošās segmentācijas klases krāsu. Krāsu kodēšanai tiek izmantots RGB (R – sarkans, G - zaļš, B – zils) krāsu modelis. Viens segmentācijas attēla pikselis atbilst vienam augstuma punktam. Lai no segmentācijas attēliem iegūtu virsmas klašu vektorus, katra segmentācijas attēla pikseļa krāsa tika salīdzināta ar virsmas klasēm atbilstošajām segmentācijas krāsām, konkrētāk:

- (0, 0, 0) - asfalta virsmas klasei - [1, 0, 0, 0, 0];
- (99, 69, 44) - cieto dubļu virsmas klasei - [0, 1, 0, 0, 0];
- (46, 194, 126) - zāles virsmas klasei - [0, 0, 1, 0, 0];
- (98, 160, 234) - ūdens virsmas klasei - [0, 0, 0, 1, 0];

Aiz virsmas klašu nosaukumiem ir dots attiecīgo klašu vienizcēluma kodējums. Jebkura cita krāsa, kas nesakrita ar nevienu no definētajām virsmas klases krāsām, tika pārveidota par objektu jeb virsmas klasi ar vienizcēluma kodējumu [0, 0, 0, 0, 1].

Iegūtie absolūtā augstuma punkti un tiem atbilstošie klašu vektori tika apvienoti vienā ROS ziņojumā un publicēti ar frekvenci 5 Hz jeb vienu ziņojumu 0,2 sekundēs.

Rezultātā iegūtie dati tika ierakstīti ROS multikopā (angliski: *bag*). Multikopas saturēja 28283 ziņojumus: 6670 kameras attēlu ziņojumus, 1610 absolūtā augstuma punktu ziņojumus un 20003 bezpilota lidaparāta pozas ziņojumus.

2.6.2. SLAM sistēmas rezultātu ieguve

SLAM sistēmas rezultātu ieguvei SLAM sistēmas ievadē tika izmantoti ROS multikopā ierakstītie dati. Rezultātā kartētā globālā vokseļu karte un punktu mākonis tika iegūti, izmantojot ROS servisus un tematus.

Globālās vokseļu kartes ieguvei tika izveidots ROS serviss */global_elevation_flush*. Servisa pieprasījuma ziņojums ir simbolu virkne, kurā ir norādīts ceļš līdz datnei, kurā tiks saglabāta vokseļu karte. Servisa atbildes ziņojums ir simbolu virkne ar paziņojumu par datu ieguves sekmību. Saņemot pieprasījuma ziņojumu, serveris, kas šajā gadījumā ir globālais kartētājs, iegūst globālās vokseļu kartes datus. Iegūtie globālās vokseļu kartes dati sastāv no visu vokseļu koordinātām un visu vokseļu ierakstu klašu vektoriem (ja vienam vokselim ir vairāki ieraksti, katram vokseļa ierakstam tiek saglabātas atsevišķas vokseļa koordinātas), kā arī no vokseļu kartes izšķirtspējas (0,25). Semantikas vektori no ierakstiem netika izgūti, tā kā to izveidošana un ievietošana ierakstos tika implementēta ārpus šī darba ietvariem. Iegūtie dati tika saglabāti pieprasījuma ziņojuma norādītajā datnē - binārajā datnē.

Globālais punktu mākonis tika iegūts no ROS temata, kurā tas tika publicēts - */global_mapper_occ*. Iegūtais *PointCloud2* [34] ziņojums tika pārveidots par NumPy [35] masīviem: punktu mākoņa punktu masīvu un punktu aizpildījuma vērtību masīvu. Izveidotie masīvi tika saglabāti *.npz* datnē. Jāpiemin, ka, lai gan punktu aizpildījuma vērtības tika saglabātas, rezultātu novērtēšanā tās netika izmantotas, tāpēc ka visiem punktiem tās bija identiskas (vērtība 0).

2.6.3. Patiesuma vērtību ieguve

Patiesuma vērtību (angliski: *ground truth*) ieguve, tāpat kā priekšapstrādātu datu ieguve, tika veikta simulācijas darbības laikā. Tas nodrošināja to, ka patiesuma vērtības globālā vokseļu karte un punktu mākonis tika izveidoti no tiem pašiem lidojumā iegūtajiem datiem, kas bija pieejami SLAM sistēmai ievadē no ROS multikopas.

Patiesuma vērtības globālās vokseļu kartes dati tika iegūti, apvienojot visus lidojumā iegūtos klašu vektorus. Atšķirībā no SLAM sistēmas globālās vokseļu kartes, patiesuma vērtības globālajā vokseļu kartē klašu vektori netika ievietoti ierakstos, kas atbilstu kādam konkrētam vokselim. Respektīvi, patiesuma vērtības globālā vokseļu karte ir blīvāka. Iegūtie patiesuma vērtības globālās vokseļu kartes dati sastāv no visiem klašu vektoriem un tiem atbilstošajiem augstuma punktiem (to koordinātām), kā arī no augstuma punktu izšķirtspējas. Iegūtie dati tika saglabāti identiskā formātā kā SLAM sistēmas rezultāti, arī binārajā datnē.

Patiesuma vērtības globālais punktu mākonis tika iegūts, apvienojot visus lidojumā iegūtos unikālos absolūtā augstuma punktus. Pirms punktu apvienošanas, tie tika pārveidoti SLAM sistēmas kartēšanas atskaites sistēmā. Punktu pārveidošana tika veikta, izmantojot vienādojumu $t_K = R_{KD}t_D + t_{KD}$, kur t_D ir augstuma punktu koordinātas bezpilota lidaparāta atskaites sistēmā D. Patiesuma vērtības globālais punktu mākonis tika saglabāts identiskā formātā kā SLAM sistēmas rezultāti, arī *.npz* datnē.

2.6.4. Rezultātu novērtēšanas metode

Lai novērtētu SLAM sistēmas rezultātus, binārās patiesuma vērtību un SLAM sistēmas rezultātu datnes tika pārveidotas par *.npz* datnēm. Respektīvi, vokseļu vai punktu koordinātas tika ievietotas NumPy masīvā. Klašu vektoru vērtības tika sadalītas pēc klasēm, tas ir, katras virsmas klases vērtības tika ievietotas atsevišķā masīvā.

Iegūtās četras (2 punktu mākoņi un 2 vokseļu kartes) *.npz* datnes tika turpmāk izmantotas rezultātu novērtēšanai.

Rezultātu novērtēšanai gan punktu mākoņu, gan vokseļu karšu gadījumā tiek izmantotas trīs metrikas: precizitāte, pārklājums, šķēlums pār apvienojumu (IoU). Vokseļu karšu gadījumā, izņemot minētās metrikas, rezultātu novērtēšanai tiek veikta arī SLAM sistēmas rezultātos iegūto klašu vektoru salīdzināšana ar patiesuma vērtības klašu vektoriem. Klašu vektoru kartēšanas precizitāte tiek novērtēta procentuāli, tāpat kā pārējās metrikas.

Precizitātes metrikas ietvaros tiek novērtēts, cik daudz SLAM sistēmas kartēšanas rezultātā iegūto punktu mākoņa un vokseļa kartes koordinātu sakrīt ar patiesma vērtības punktu mākoņa un vokseļu kartes koordinātām. Jo lielāka ir precizitāte, jo lielāka ir sakritība starp patiesuma vērtībām un kartēšanas rezultātiem. Kartēšanas rezultātu precizitāti var izteikt kā:

$$\text{Precizitāte} = \frac{\text{pareizas atbilsmes}}{\text{rezultāts}} \tag{2.5}$$

Ar šo pašu vienādojumu var izteikt arī klašu vektoru kartēšanas precizitāti.

Pārklājuma metrikas ietvaros tiek novērtēts, cik liels patiesuma vērtības apgabals tika kartēts. Jo lielāks ir pārklājums, jo lielāks patiesuma vērtības apgabals ir iekļauts izveidotajā kartē. Pārklājumu var izteikt kā:

$$\text{Pārklājums} = \frac{\text{pareizas atbilsmes}}{\text{patiesuma vērtība}} \tag{2.6}$$

IoU metrikas ietvaros tiek novērtēta patiesuma vērtības un SLAM sistēmas kartēšanas rezultātu pārklāšanās. Atšķirībā no pārklājuma metrikas IoU gadījumā pārklāšanās tiek izteikta attiecībā pret kartēšanas rezultātu un patiesuma vērtību koordinātu apvienojumu, nevis tikai patiesuma vērtību koordinātām. IoU var izteikt kā:

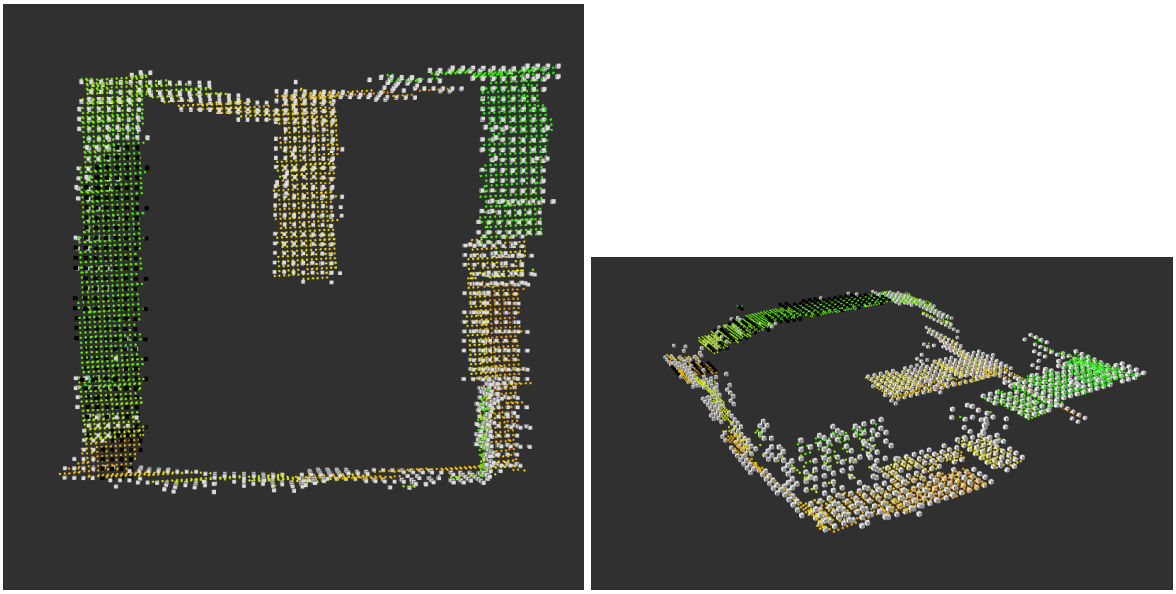
$$IoU = \frac{\text{šķēlums}}{\text{apvienojums}} \tag{2.7}$$

Pirms rezultātu novērtēšanas, gan vokseļu karšu koordinātām, gan punktu mākoņu punktiem tiek veikta vokselizācija. Vokselizācijas rezultātā tiek normalizēta SLAM sistēmas rezultātu un patiesuma vērtību izšķirtspēja (uz 0,5). Tā kā izšķirtspēja tiek normalizēta uz zemāku izšķirtspēju, izdzēšot no vokselizācijas rezultātā iegūtajiem SLAM sistēmas rezultātiem un patiesuma vērtībām vērtības, kas atkārtojas, tiek nodrošināts, ka abas datu kopas ir identiska blīvuma un izšķirtspējas. Tas sniedz iespēju objektīvāk novērtēt iegūtos rezultātus, vienādojot iegūto rezultātu un patiesuma vērtību datu formātu.

Klašu vektoru salīdzināšanas ietvaros identiskās koordinātu vērtības pēc vokselizācijas uzreiz netiek izdzēstas. Vispirms koordinātu vērtībām tiek pievienoti klašu vektori. Koordinātu vērtībām, kuras ir identiskas, klašu vektori tiek saskaitīti kopā. Rezultātā koordināta pieder tai virsmas klasei, kurai pēc klašu vektoru saskaitīšanas bija vislielākā vērtība. Pēc klašu vektoru saskaitīšanas, identiskās koordinātu vērtības tiek izdzēstas, lai pēc tam veiktu klašu vektoru salīdzināšanu. Klašu vektoru salīdzināšana tiek veikta tikai uz pareizas atbilsmes (angliski: *true positive*) koordinātām, tas ir, koordinātām, kas bija sastopamas gan SLAM sistēmas rezultātā, gan patiesuma vērtībā.

3. REZULTĀTI UN DISKUSIJA

SLAM sistēmas kartēšanas rezultātā tika iegūts globāls punktu mākonis ar kartētajiem punktiem un globāla vokseļu karte ar kartētajiem vokseļiem. Kartēšanas rezultātus ir iespējams redzēt 3.1 attēlā. Attēlos ir redzami gan globālais punktu mākonis, gan globālā vokseļu karte. Krāsainie punkti pieder globālajam punktu mākonim, savukārt melnbaltie kubi ir globālās vokseļu kartes vokseļi.



(a) *(b)*

Att. 3.1: **Kartēšanas rezultāti no augšas (a) un no sāniem (b)**

Iegūto rezultātu novērtējumi ir aplūkojami 3.1 tabulā.

Tabula 3.1: **Iegūtie SLAM sistēmas rezultāti, novērtēti procentuāli**

Rezultāti, %	**Precizitāte**	**Pārklājums**	**IoU**	**Klašu vektoru precizitāte**
Punktu mākonis	87,24	57,48	53,02	-
Vokseļu karte	89,96	68,45	63,59	19,38

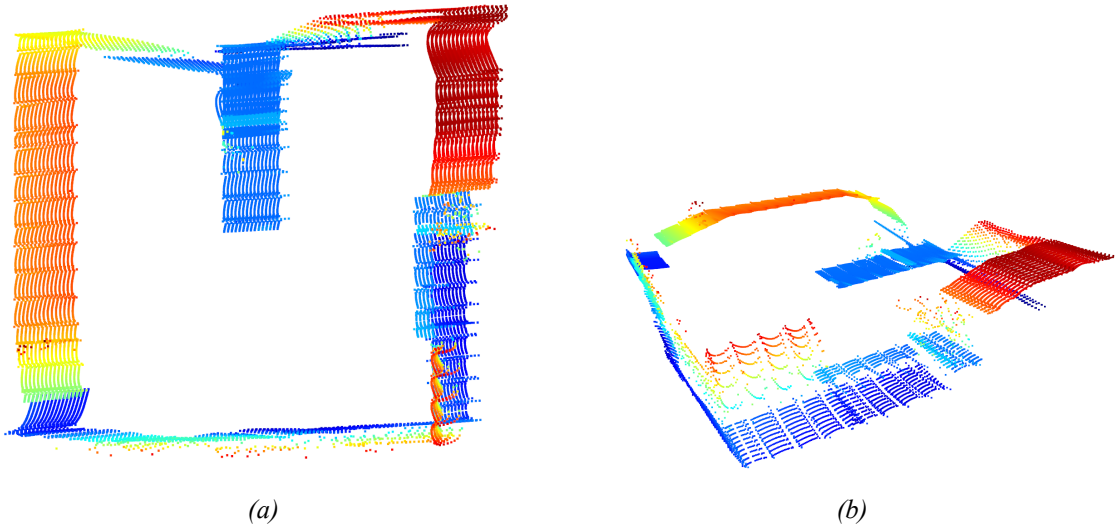
Pēc iegūto rezultātu novērtējuma ir redzams, ka SLAM sistēmas kartēšanas rezultātā iegūtā vokseļu karte atbilst patiesuma vērtībai tuvāk nekā iegūtais punktu mākonis. Iegūtā atšķirība ir izskaidrojama ar atšķirīgas kartēšanas metodes pielietojumu attiecīgajām globālās kartes sastāvdaļām. Lokālā kartētāja gadījumā punktu mākonis tiek saglabāts oktantkokā, savukārt vokseļu kartes ieraksti tiek glabāti atsevišķā virknē. Globālajā kartētājā oktantkoka vērtības tiek apvienotas ar globālo punktu mākoni, savukārt ierakstu virknes ieraksti tiek ievietoti atbilstošajos vokseļu kartes vokseļos. Abām globālās kartes sastāvdaļām - punktu mākonim un vokseļu kartei – ir augsta precizitāte, tās pārklāj vairāk nekā pusi no patiesuma vērtībām. Globālās kartes sastāvdaļu un patiesuma vērtību pārklājumi (IoU) ir virs 50%, respektīvi, pareizas atbilsmes

vērtības sastāda vairāk nekā 50% no visām rezultātos iegūtajām vērtībām (patiesuma vērtībām un SLAM sistēmas rezultātiem) kopā.

Turpmākajās apakšnodaļās globālā punktu mākoņa novērtēšana tiks aplūkota atsevišķi no globālās vokseļu kartes novērtēšanas, vispirms aplūkojot punktu mākoņa novērtēšanu.

3.1. Globālais punktu mākonis

Globālā punktu mākoņa patiesuma vērtība ir redzama 3.2 attēlā. Zilie punkti punktu mākonī ir ar zemāku augstumu nekā sarkanie punkti.

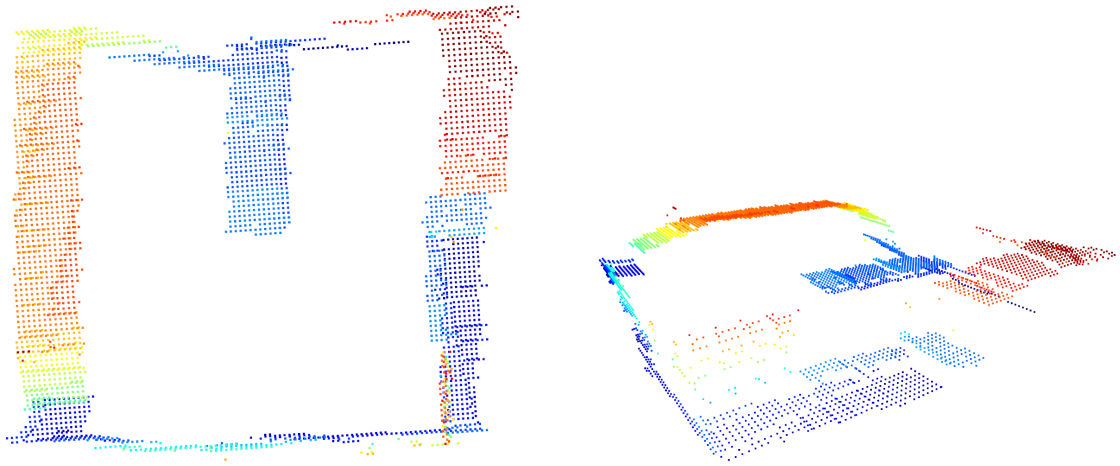


(a) *(b)*

Att. 3.2: **Patiesuma vērtības punktu mākonis no augšas (a) un no sāniem (b)**

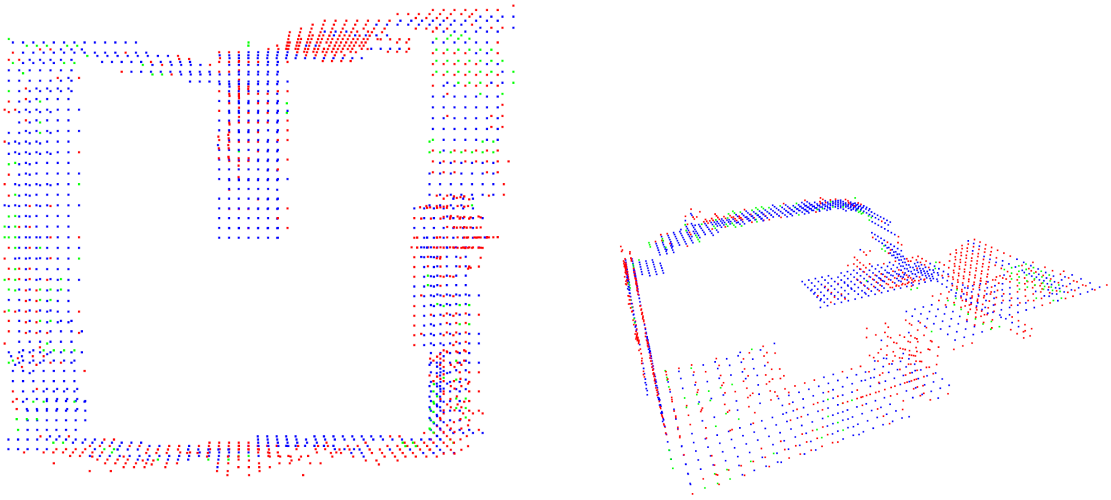
SLAM sistēmas globālais punktu mākonis, kas tika salīdzināts ar punktu mākoņa patiesuma vērtību, ir redzams 3.3 attēlā. Kā redzams, SLAM sistēmas globālais punktu mākonis ir retināts, respektīvi, tas ir mazāks blīvs nekā patiesuma vērtības punktu mākonis.

Globālā punktu mākoņa sakritības ar patiesuma vērtību vizualizāciju var redzēt 3.4 attēlā. Sarkanie punkti attēlā atbilst patiesuma vērtības punktu mākoņa punktiem, kas netika kartēti (kļūdaina neatbilsme (angliski: *false negative*)). Zaļie punkti atbilst SLAM sistēmas punktu mākoņa punktiem, kas nebija atrodami patiesuma vērtības punktu mākonī (kļūdaina atbilsme). Zilie punkti atbilst SLAM sistēmas punktu mākoņa punktiem, kas bija atrodami patiesuma vērtības punktu mākonī (pareiza atbilsme). Kā redzams attēlā, lielākā neatbilstība starp SLAM sistēmas punktu mākoni un patiesuma vērtības punktu mākoni bija tajos apgabalos, kuros strauji mainījās augstums, vai arī tajos apgabalos, kuros tika kartēts šaurs, bet augstuma ziņā daudzveidīgs apgabals.



(a) *(b)*

Att. 3.3: **SLAM sistēmas punktu mākonis no augšas (a) un no sāniem (b)**



(a) *(b)*

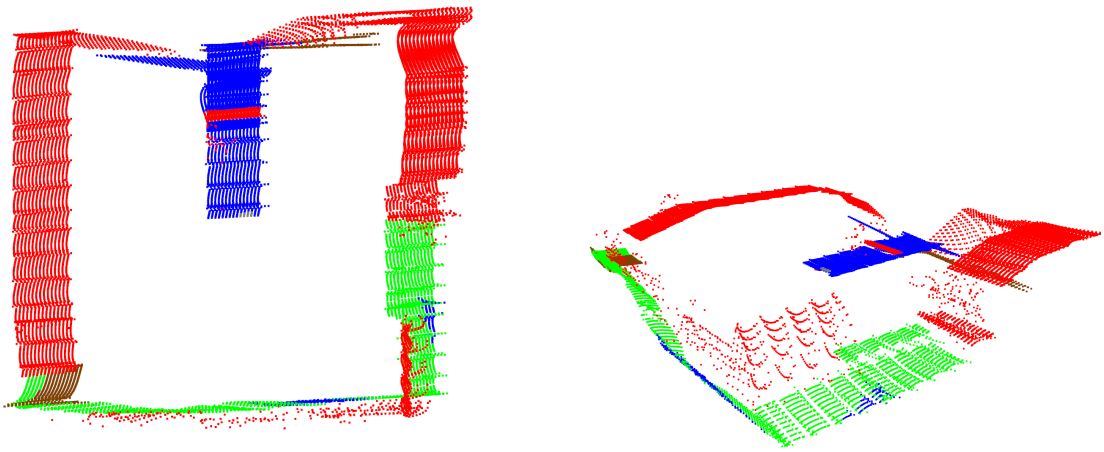
Att. 3.4: **Punktu mākoņa novērtējuma vizualizācija no augšas (a) un no sāniem (b)**

3.2. Globālā vokseļu karte

Globālās vokseļu kartes patiesuma vērtība ir redzama 3.5 attēlā. Attēlā redzamo vokseļu krāsa atbilst attiecīgo vokseļu virsmas klasei, konkrētāk:

- pelēka - asfalta virsmas klasei - [1, 0, 0, 0, 0];
- brūna - cieto dubļu virsmas klasei - [0, 1, 0, 0, 0];
- zaļa- zāles virsmas klasei - [0, 0, 1, 0, 0];
- zila - ūdens virsmas klasei - [0, 0, 0, 1, 0];
- sarkana – objekta virsmas klasei - [0, 0, 0, 0, 1].

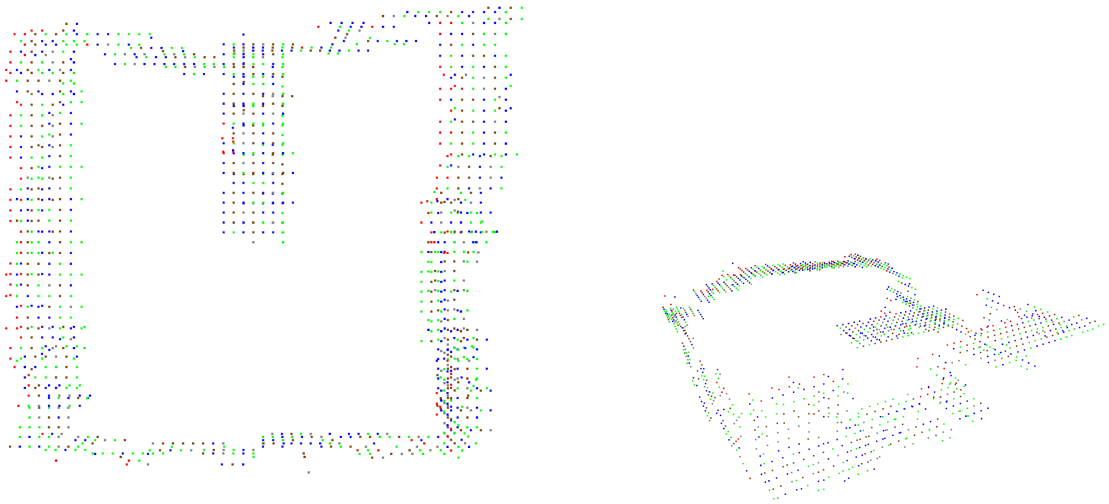
SLAM sistēmas globālā vokseļu karte, kas tika salīdzināta ar vokseļa kartes patiesuma vērtību, ir redzama 3.6 attēlā. Kā redzams, SLAM sistēmas globālā vokseļu karte ir retināta.



(a) *(b)*

Att. 3.5: **Patiesuma vērtības vokseļu karte no augšas (a) un no sāniem (b)**

Tāpat ir redzams, ka SLAM sistēmas kartēšanas rezultātā iegūtās vokseļu kartes klašu vektori ievērojami neatbilst patiesuma vērtības klašu vektoriem. Tā kā klašu vektoru vērtības netika mainītas datu buferizācijas posmā, pieļaujams, ka klašu vektoru nesakritību varēja izraisīt modificētā ierakstu ievietošanas funkcija. Respektīvi, ierakstu apvienošanas posmā, apvienojot ierakstu klašu vektorus, rezultātā iegūtais klašu vektors varēja tikt izrēķināts nepareizi, kas rezultējās ar zemu klašu vektoru sakritību rezultātu novērtējumā.

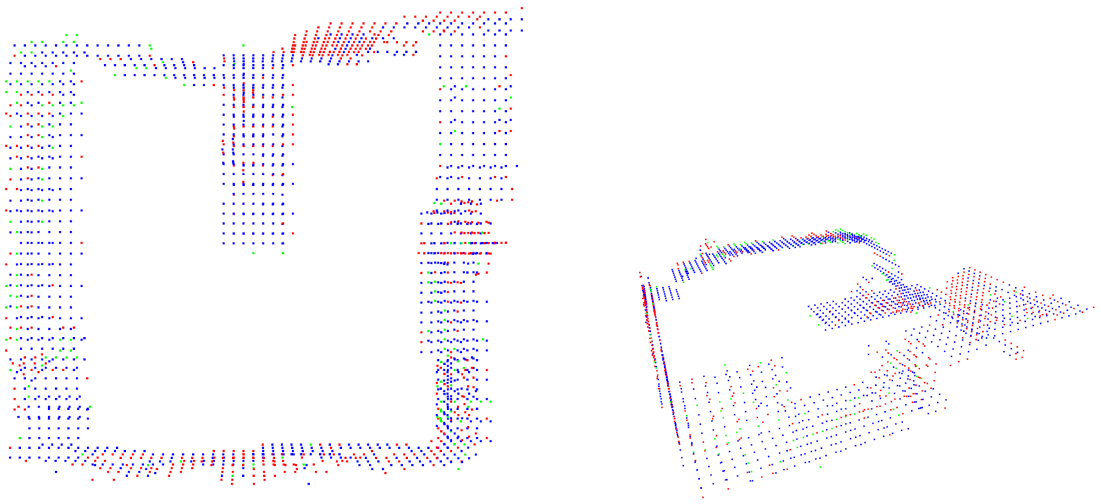


(a) *(b)*

Att. 3.6: **SLAM sistēmas vokseļu karte no augšas (a) un no sāniem (b)**

Globālās vokseļu kartes sakritības ar patiesuma vērtību vizualizāciju var redzēt 3.7 attēlā. Tāpat kā globālā punktu mākoņa gadījumā, sarkanie punkti attēlā atbilst patiesuma vērtības vokseļu kartes vokseļiem, kas netika kartēti (kļūdaina neatbilsme. Zaļie punkti atbilst SLAM sistēmas vokseļu kartes punktiem, kas nebija atrodami patiesuma vērtības vokseļu kartē (kļūdaina atbilsme). Zilie punkti atbilst SLAM sistēmas vokseļu kartes vokseļiem, kas bija atrodami patiesuma vērtības vokseļu kartē (pareiza atbilsme). Identiski globālajam punktu mākonim lie-

lākā neatbilstība starp SLAM sistēmas vokseļu karti un patiesuma vērtības vokseļu karti bija tajos apgabalos, kuros strauji mainījās augstums, un tajos apgabalos, kuros tika kartēts šaurs, bet augstuma ziņā daudzveidīgs (blīvs) apgabals.



(a) *(b)*

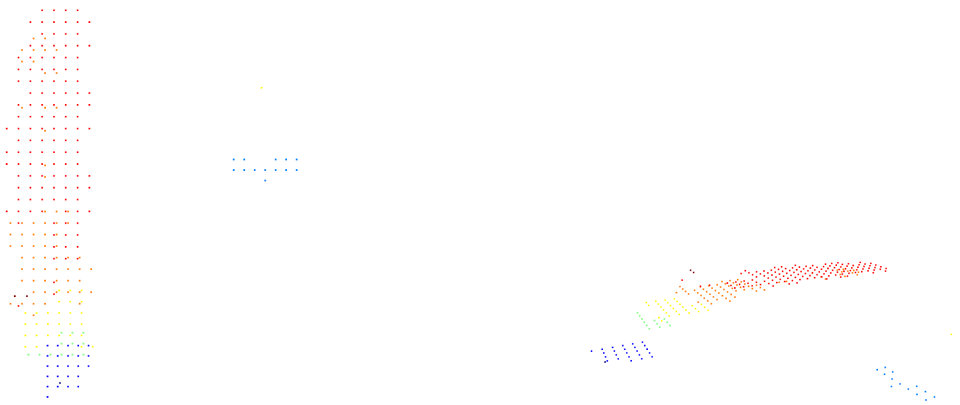
Att. 3.7: **Vokseļu kartes novērtējuma vizualizācija no augšas (a) un no sāniem (b)**

3.3. Diskusija

Novērtējot SLAM sistēmas kartēšanas rezultātā iegūto globālo vokseļu karti un punktu mākoni, iegūtie rezultāti ir vērtējami kā apmierinoši, izņemot klašu vektoru apvienošanas gadījumā. Jāpiemin, ka sākotnējā iecere ievietot klašu vektorus tikai tajos ierakstos un tikai tām koordinātām, kurās ir pieejami semantikas vektori, sniegtu vēl neapmierinošākus rezultātus, konkrētāk, lielākā daļa klašu vektoru pat netiktu iekļauti vokseļu kartē. Attiecīgi SLAM sistēmas kartēšanas rezultātā iegūtā vokseļu karte izskatītos kā 3.8 attēlā. Attēlā redzamās vokseļu kartes retinātā struktūra izskaidrojama ar klašu vektoru koordinātu nesakritību ar semantikas vektoru koordinātām. Šī iemesla dēļ klašu vektoru ievietošana tika pārveidota par neatkarīgu no semantikas vektoru ievietošanas, konkrētāk, klašu vektori tika ievietoti ierakstos pat tad, ja to koordinātas nesakrita ar semantikas vektoru koordinātām. Gadījumos, kad tās tomēr sakrita, semantikas vektori un klašu vektori tika ievietoti vienā kopīgā ierakstā.

Lai gan šāds atrisinājums nodrošina klašu vektoru ievietošanas neatkarību no semantikas vektoriem un to precizitātes, tas neatrisina vektoru koordinātu nesakritības cēloni. Iespējams, koordinātu nesakritība ir izskaidrojama ar atšķirīgām datu plūsmām, no kurām tiek iegūti attiecīgie vektori: atšķirīgiem ROS ziņojumiem, to buferiem un atšķirīgas buferu datu apstrādes (kartēšanas metodes). Konkrētāk, attēla semantikas vektori varētu tikt projicēti uz tādām koordinātām, kas nesakrīt ar lokālās kartes oktantkokā ievietoto punktu koordinātām. Lokālā kartētāja implementācijas ietvaros semantikas vektoru koordinātas, kas neatbilst lokālās kartes oktantkoka koordinātām, netiek ievietotas ierakstos (tāpat netiek ievietoti šo koordinātu semantikas vek-

tori). Rezultātā attiecīgie semantikas vektori netiek kartēti vispār, un koordinātu nesakritības dēļ arī klašu vektori netiek kartēti (iepriekšējā implementācijā), izveidojot gandrīz tukšu vokseļu karti (bez vokseļiem). Viens no potenciālajiem problēmas risinājumiem vektoru koordinātu sakritībai būtu kameras kalibrēšana, iegūstot precīzāku kameras pozas novērtējumu relatīvi bezpilota lidaparātam.



(a) *(b)*

Att. 3.8: **Sākotnējās implementācijas iegūtā SLAM sistēmas vokseļu karte no augšas (a) un no sāniem (b)**

SECINĀJUMI

Darbā izvirzītie mērķi tika sasniegti.

Darba ietvaros tika secināts, ka SLAM sistēmā notiek pastāvīga mijiedarbība starp lokalizāciju un kartēšanu. Aģenta lokalizācijas precizitāte būtiski ietekmē izveidotās kartes kvalitāti. Nav tik būtiski, kur aģents atrodas - lidojumā vai uz zemes, vai zem zemes -, būtiski ir zināt tā pozu. Respektīvi, lai gan darba ietvaros uzsvars tika likts uz lidojumā iegūtu datu apvienošanu, fakts, ka tie ir lidojumā iegūti dati, nebija noteicošais pētāmās problēmas atrisinājuma izvēlē. Attiecīgais risinājums būtu pielietojams arī datiem, kas iegūti no citām atrašanās vietām telpā.

Tāpat tika secināts, ka izvēlētā problēmas atrisinājuma metode spēj precīzi kartēt aģenta apkārtējo vidi, taču izveidotās globālās kartes ir retinātas. Ņemot vērā globālās kartes pamatprincipu, respektīvi, iekļaut informāciju par aģenta apkārtni visa tā pārvietošanās laikā, attiecīgais secinājums, ka kartes ir retinātas, nav problemātisks. Retinātu karšu gadījumā ietaupās atmiņas daudzums, kas ir nepieciešams kartes glabāšanai sistēmas atmiņā. Tā kā globālajās kartēs var tikt iekļautas kartētās teritorijas vairāku desmitu vai simtu metru garumā un platumā, retinātu karšu pielietojums var būt izdevīgs sistēmas resursu ietaupīšanas nolūkos.

Problemātisks ir secinājums, ka implementētā metode nespēj precīzi kartēt apkārtējās teritorijas virsmas klases. Kā viens no pirmajiem plānotajiem tālākās darbības soļiem ir novērst attiecīgo problēmu. Pēc problēmas novēršanas, tālākā darbība ietvertu sevī implementētās metodes koda refaktorēšanu (angliski: *refactoring*), uzlabojot vispārējo SLAM sistēmas struktūru. Tālākās darbības ietvaros SLAM sistēmas veiktspējas novērtēšanai varētu izmantot ievadi no diviem aģentiem. Objektīvai sistēmas novērtēšanai būtu nepieciešams sagatavot patiesuma vērtību datus abu aģentu ievadēm.

Darba ietvaros tika pieņemts, ka sistēmas ievadē padotā bezpilota lidaparāta poza ir patiesa. Reālu (ne imitētas vides) apstākļu gadījumā šāda situācija ir praktiski nesastopama. Attiecīgi kā vēl viens no tālākās darbības soļiem būtu implementētās metodes pielāgošana bezpilota lidaparāta pozām, kas ir trokšņainas (var būt kļūdainas), respektīvi, būtu jāpaplašina bezpilota lidaparāta lokalizācijas modulis.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] Christian Colliander. “Efficient 2D SLAM for a Mobile Robot with a Downwards Facing Camera”. 2018. URL: https://api.semanticscholar.org/CorpusID:55378096.

[2] Hailong Qin u. c. “Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-denied Environments”. *IEEE Transactions on Vehicular Technology* (2018. g. janv.). DOI: 10.1109/TVT.2018.2890416.

[3] Tiago Brito Novo. “EXAMINER-3D Multi-Robot Exploration in Irregular Terrains”. Mag. darbs. 2021.

[4] Shuran Zheng u. c. “Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis”. *Remote Sensing* 15.4 (2023). ISSN: 2072-4292. DOI: 10.3390/rs15041156. URL: https://www.mdpi.com/2072-4292/15/4/1156.

[5] Shanhua Chen. “Visual SLAM technology for medical”. *Applied and Computational Engineering* 12 (2023. g. sept.), 220.—224. lpp. DOI: 10.54254/2755-2721/12/20230345.

[6] Marziyeh Bamdad, Davide Scaramuzza un Alireza Darvishy. “SLAM for Visually Impaired People: A Survey”. *IEEE Access* 12 (2024), 130165.—130211. lpp. DOI: 10.1109/ACCESS.2024.3454571.

[7] Ārija Kalniņa. *Bezpilota lidaparāta sensoru integrācija SLAM sistēmās*. Latvijas Universitāte, Eksakto zinātņu un tehnoloģiju fakultāte. Kursa darbs. 2025.

[8] R.K. Shukla un Anchal Srivastava. *Mechanics*. New Age International Ltd, Daryaganj, 2006. ISBN: 9788122418750.

[9] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[10] Bruno Siciliano un Oussama Khatib, izdev. *Springer Handbook of Robotics*. Springer International Publishing AG, 2016.

[11] Herbert Goldstein, Charles Poole un John Safko. *Classical Mechanics (3rd Edition)*. 2001. g. jūn. ISBN: 0201657023.

[12] William Rowan Hamilton. “On Quaternions; or on a new System of Imaginaries in Algebra”. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1850). Izdevis David R. Wilkins. URL: https://www.maths.tcd.ie/pub/HistMath/People/Hamilton/OnQuat/OnQuat.pdf.

[13] Yanyan Li u. c. “RGB-D SLAM with Structural Regularities”. *2021 IEEE international conference on Robotics and automation (ICRA)*. 2021.

[14] Maxime Ferrera u. c. “OV^{2}SLAM : A Fully Online and Versatile Visual SLAM for Real-Time Applications”. *IEEE Robotics and Automation Letters* (2021).

[15] Devendra Singh Chaplot u. c. “Learning To Explore Using Active Neural SLAM”. *International Conference on Learning Representations (ICLR)*. 2020.

[16] Chao Yu u. c. *Learning Efficient Multi-Agent Cooperative Visual Exploration*. 2021. arXiv: 2110.05734 [cs.CV].

[17] Cesar Cadena u. c. “Simultaneous Localization And Mapping: Present, Future, and the Robust-Perception Age”. *IEEE Transactions on Robotics* 32 (2016. g. jūn.). DOI: 10.1109/TRO.2016.2624754.

[18] Peteris Racinskis, Janis Arents un Modris Greitans. “Constructing Maps for Autonomous Robotics: An Introductory Conceptual Overview”. *Electronics* 12.13 (2023). ISSN: 2079-9292. URL: https://www.mdpi.com/2079-9292/12/13/2925.

[19] Marina Indri un Roberto Oboe, izdev. *Mechatronics and Robotics: New Trends and Challenges*. Taylor & Francis Group, 2020.

[20] *Left: Recursive subdivision of a cube into octants. Right: The corresponding octree.* URL: https://upload.wikimedia.org/wikipedia/commons/thumb/2/20/Octree2.svg/800px-Octree2.svg.png. [Piekļūts: 14.01.2025.]

[21] Jun Zhang u. c. “4DRadarSLAM: A 4D Imaging Radar SLAM System for Large-scale Environments based on Pose Graph Optimization”. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, 8333.—8340. lpp. DOI: 10.1109/ICRA48891.2023.10160670.

[22] Zhimin Zhang u. c. “Review of monocular depth estimation methods”. *Journal of Electronic Imaging* 34.2 (2025), 20901.—20901. lpp.

[23] Zhengyou Zhang. “A Flexible New Technique for Camera Calibration”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), 1330.—1334. lpp.

[24] MathWorks. *What Is Camera Calibration?* 2025. URL: https://se.mathworks.com/help/vision/ug/camera-calibration.html.

[25] Steven Macenski u. c. “Robot Operating System 2: Design, architecture, and uses in the wild”. *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[26] Jay A Farrell. “Computation of the Quaternion from a Rotation Matrix”. *University of California* 2 (2015).

[27] ROS 2 Documentation. *geometry_msgs/msg/TransformStamped Message*. 2020. URL: https://docs.ros2.org/latest/api/geometry_msgs/msg/TransformStamped.html.

[28] ROS 2 Documentation. *sensor_msgs/msg/Image Message*. 2020. URL: https://docs.ros2.org/latest/api/sensor_msgs/msg/Image.html.

[29] Webots. *http://www.cyberbotics.com*. Izdevis Cyberbotics Ltd. Open-source Mobile Robot Simulation Software. URL: http://www.cyberbotics.com.

[30] Webots. *Webots GPS Documentation*. Izdevis Cyberbotics Ltd. 2025. URL: https://cyberbotics.com/doc/reference/gps.

[31] Webots. *Webots InertialUnit Documentation*. Izdevis Cyberbotics Ltd. 2025. URL: https://cyberbotics.com/doc/reference/inertialunit.

[32] Webots. *Webots RangeFinder Documentation*. Izdevis Cyberbotics Ltd. 2025. URL: https://cyberbotics.com/doc/reference/rangefinder.

[33] Webots. *Webots Camera Documentation*. Izdevis Cyberbotics Ltd. 2025. URL: https://cyberbotics.com/doc/reference/camera.

[34] ROS Documentation. *sensor_msgs/msg/PointCloud2 Documentation*. 2025. URL: https://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/PointCloud2.html.

[35] Charles R. Harris u. c. "Array programming with NumPy". *Nature* 585.7825 (2020. g. sept.), 357.—362. lpp. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

DOKUMENTĀRĀ LAPA

Bakalaura darbs “Lidojumā iegūtu datu apvienošana ar vokseļu SLAM sistēmu” izstrādāts LU Eksakto zinātņu un tehnoloģiju fakultātē Datorikas nodaļā.

Ar savu parakstu (drošs elektroniskais paraksts) apliecinu, ka pētījums veikts patstāvīgi un izmantoti tikai tajā norādītie informācijas avoti

Autors: (drošs elektroniskais paraksts) Ārija Kalniņa (stud. apl. Nr. ak21374)

Darba vadītājs: (drošs elektroniskais paraksts) Mg. sc. comp. Pēteris Račinskis

Recenzents: Oskars Ozols

Darbs augšupielādēts LUIS’ā 26.05.2025.