SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Ing. Filip Štec

Autoreferát dizertačnej práce

GLOBÁLNA SÉMANTICKÁ MAPA PRE NAVIGÁCIU AUTONÓMNYCH LIETAJÚCICH PROSTRIEDKOV

na získanie akademického titulu
„doktor“ („philosophiae doctor“, v skratke „PhD.“)

v doktorandskom študijnom programe:	Robotika a Kybernetika
v študijnom odbore:	9.2.7 kybernetika
Forma štúdia:	denná forma
Miesto a dátum:	Bratislava 2024

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Dizertačná práca bola vypracovaná na:
Slovenskej technickej univerzity v Bratislave,
Fakulta elektrotechniky a informatiky,
Ústav robotiky a kybernetiky

Predkladateľ: Ing. Filip Štec
Slovenská technická univerzita v Bratislave,
Fakulta elektrotechniky a informatiky,
Ústav robotiky a kybernetiky

Školiteľ: Ing. Jozef Rodina, PhD.
Slovenská technická univerzita v Bratislave,
Fakulta elektrotechniky a informatiky,
Ústav robotiky a kybernetiky

Oponent: prof. Ing. Michal Kelemen, CSc.
Technická univerzita v Košiciach,
Strojnícka fakulta,
Ústav automatizácie, mechatroniky, robotiky a výrobnej techniky

Oponent: prof. Ing. Luděk Žalud, Ph.D.
Vysoké učení technické v Brně,
Fakulta elektrotechniky a komunikačních technologií,
Ústav automatizace a měřicí techniky

Autoreferát bol rozoslaný: .....

Obhajoba dizertačnej práce sa koná: 22.8.2024 o 9:00 h

na Fakulte elektrotechniky a informatiky STU v Bratislave, Ilkovičova 3, Bratislava, v miestnosti D424 pred komisiou pre obhajobu dizertačnej práce doktorandského štúdia vymenovanou predsedom spoločnej odborovej komisie 9.2.7 Kybernetika.

.....
prof. Ing. Vladimír Kutiš, PhD.
dekan fakulty
Fakulta elektrotechniky a informatiky
Slovenská technická univerzia

ANOTÁCIA DIZERTAČNEJ PRÁCE

Kľúčové slová: mapovanie, autonómny lietajúci stroj, sémantická segmentácia, výpočtová optimalizácia

Lietajúca robotika sa dostáva postupne do viacerých oblastí využitia, kde voľná možnosť pohybovania sa v trojrozmernom priestore prináša veľkú výhodu oproti iným technikám. Jednou z takýchto oblastí je inventúra skladov, kde autonómny dron dokáže nahradiť ľudskú prácu, čím je tento proces efektívnejší, presnejší a bezpečnejší. Tieto sklady však predstavujú uzavreté priestory s úzkymi uličkami a rôznymi prekážkami. Pre vykonanie inšpekcie musí teda daný autonómny dron poznať mapu prostredia aj s vyznačenými inšpekčnými oblasťami, aby sa dokázal navigovať k tovaru umiestnenému v sklade.

Táto dizertačná práca popisuje inovačný prístup k automatickej tvorbe mapy skladového prostredia pomocou autonómneho lietajúceho stroja a demonštruje využitie tejto globálnej sémantickej mapy pre navigovanie inšpekčného drona. Riešenie tejto práce využíva výhody fotorealistickej simulácie pre vývoj jednotlivých častí a ich testovanie, ktorý je aj vďaka tejto práci rozšírený o ďalšie systémy. Pre mapovanie prostredia je použitý FUEL algoritmus prehľadávania neznámeho prostredia, ktorý je obohatený o graficky akcelerované algoritmy, umožňujúce mapovanie prostredia na palubnom počítači v reálnom čase. Hlboká neurónová sieť U-NET je natrénovaná na simulačne vytvorených dátach, pre pridanie sémantickej informácie o objektoch v sklade. Autonómne vytvorené sémanticky segmentované mračno bodov je spracované pre vytvorenie globálnej sémantickej mapy skladu s označenými regálmi. Takto vytvorená mapa je použitá pre naplánovanie globálnej trajektórie pre inšpekčný let drona.

Obsah

Úvod

Autonómne bezpilotné lietajúce stroje (UAV) sú výborným kandidátom pre zjednodušenie a zrýchlenie práce ako je inventúra tovarov vo veľkých skladoch. Vďaka jednoduchosti pohybu vo všetkých troch osiach sa dokážu rýchlo dostať na miesta, kde je to pre ľudí náročné a aj nebezpečné. Použiteľnosť UAV pre aplikáciu inventarizácie skladu je dokázaná v publikácii [1], ktorá je aj „de facto“ priamym pokračovaním výsledku tejto práce.

Prostredie skladov je obyčajne dobre štruktúrované prostredie so známymi objektami, rovnými líniami. Aby sa dala naplánovať trajektóriu, ktorá umožní prezrieť sklad, a následne určiť typ tovaru a jeho umiestnenie, musí existovať presná sémantická mapa tohto prostredia. Typicky je táto mapa vytvorená použitím plánu skladu alebo zameraním skladu, čo je následne ručne prepísané do sémantickej formy vhodnej pre plánovanie inšpekčných misií. V tejto práci si ukážeme, ako sa dá pomocou UAV a všeobecných informácií o sklade automaticky vytvoriť takúto mapu.

Cieľom tejto práce je celý proces autonómneho vytvorenia sémantickej mapy pomocou UAV, od prieskumného letu drona v neznámom prostredí, cez algoritmy potrebné pre mapovanie prostredia fungujúce v reálnom čase na riadiacej jednotke drona, až po vytvorenie sémantickej mapy použitej na naplánovanie trasy inšpekcie tovaru v sklade.

1 Tézy dizertačnej práce

Tézy dizertačnej práce sú:

1. Výskum autonómneho navigovania bezpilotného lietajúceho prostriedku vo vnútornom prostredí skladu za účelom tvorby hĺbkovej a sémanticky segmentovanej mapy skladu.
2. Vytváranie dát na trénovanie DNN a ich vyhodnocovanie vo foto-realistickej simulácii prostredia skladu.
3. Využitie hardvérovej akcelerácie pre optimalizáciu systému tak, aby bol vytvorený systém schopný fungovať priamo na palubnom počítači na drone v reálnom čase.
4. Vytvorenie sémantickej reprezentácie prostredia skladu využiteľnej pre globálne plánovanie trasy pre aplikáciu inšpekcie tovaru v sklade.

2 Návrh riešenia

Cieľom tejto práce je pridať sémantický vnem autonómnemu lietajúcemu prostriedku, vďaka ktorému sa výrazne zlepší jeho rozhodovacia činnosť, keďže bude rozumieť veciam, na ktoré sa pozerá. Táto práca sa najmä zaoberá sprístupnením automatických inventúr skladových priestorov pomocou autonómnych dronov, vďaka ktorým je práca skladníkov rýchlejšia, presnejšia a hlavne bezpečnejšia. Nie je to však jediné využitie daného systému, poznať okolité prostredie umožňuje aj iným odvetviam zefektívniť ich prácu.

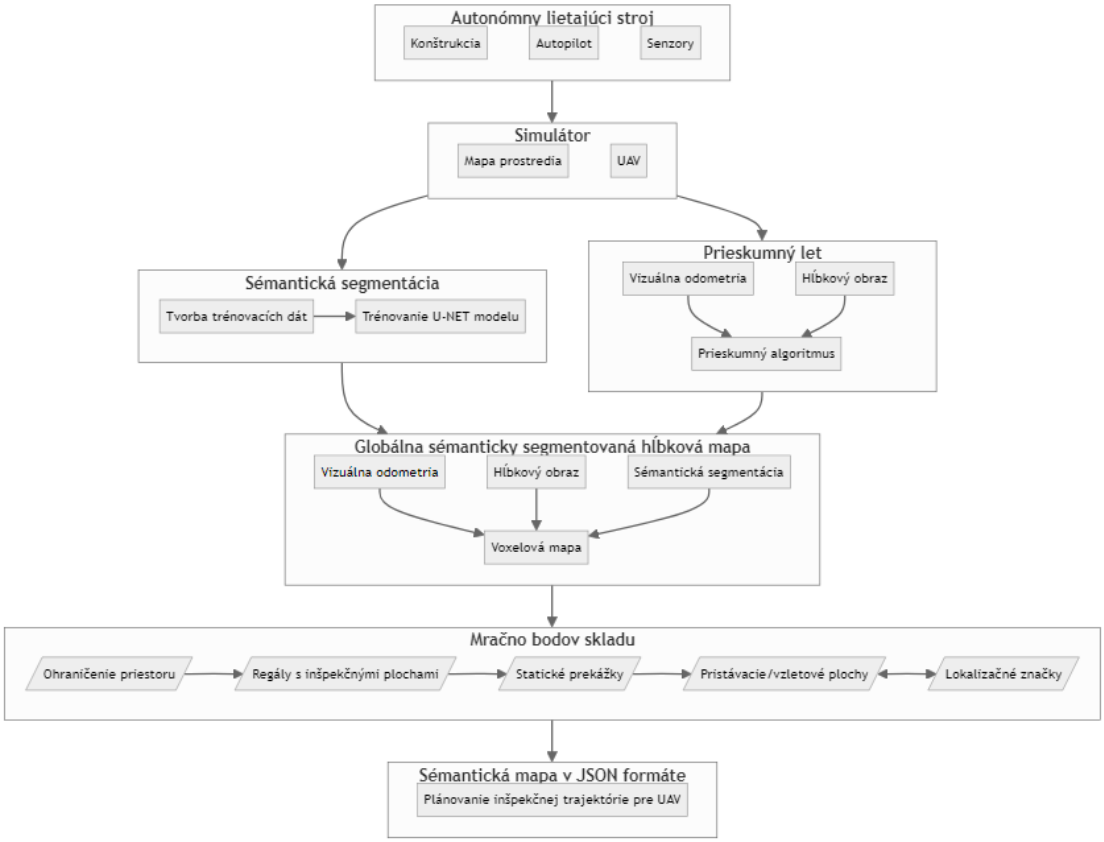
Na rozdiel od tovaru, prostredie skladu nie je často menené. Tvoria ho uličky regálov tak, aby boli dostupné vysokozdvižnými vozíkmi a zároveň aby sa tam zmestilo maximálne množstvo uskladneného tovaru. Preto mapu prostredia stačí vytvoriť vždy iba raz po zmene rozloženia skladu. Lietajúci stroj sa dokáže voľne pohybovať vo všetkých troch osiach, takže potrebuje mať zmapované aj prekážky, ktoré nie sú podstatné pre navigovanie vysokozdvižného vozíka, a teda nemusia byť správne zaznačené v pláne skladu. Preto v tejto práci uvažujeme o autonómnom vytvorení mapy skladu tak, aby si sám dron dokázal určiť priestory, ktorými sa môže pohybovať. Zároveň počas prieskumného letu budú význačné objekty zaznamenávané s pridanou sémantickou značkou, ako sú regály, steny, podlaha, prekážky a iné. To umožní vytvorenie globálnej sémanticky segmentovanej mapy prostredia skladu.

Na vývoj takéhoto autonómneho lietania v neznámom prostredí nie je vhodné hneď použiť reálny stroj, lebo je vysoké riziko pre spôsobenie škody drona alebo aj prostredia. Celý systém sa dá preniesť do simulačného prostredia, kde nehrozí takéto riziko a umožňuje rýchle upravovanie jednotlivých algoritmov ako aj izoláciu problémov. Použitím akéhokoľvek simulátora by možno umožnilo vytvoriť tento systém, avšak môže to byť ešte ďaleko od použitia v reálnom prostredí, keďže použité algoritmy budú najmä využívať vizuálne dáta.

Tento systém sa dá rozdeliť do nasledujúcich šesť častí, ktoré sú zobrazené na Obrázok 1 v diagramovej forme a následne bližšie popísané:

- Autonómny lietajúci stroj
- Simulátor
- Sémantická segmentácia
- Prieskumný let
- Tvorba globálnej segmentovanej hĺbkovej mapy skladu

- Mračno bodov skladu
- Sémantická mapa v JSON formáte



Obrázok 1: Diagram návrhu riešenia

Lietanie vo vnútornom prostredí vyžaduje vyššiu presnosť lokalizácie a navigácie, a taktiež väčšiu citlivosť na detekciu okolitých prekážok. Na základe týchto aspektov musí byť použitý stroj navrhnutý tak, aby sa zmestil do úzkych priestorov v sklade, dokázal letieť dostatočne dlho, a bol vybavený vhodným senzorovým vybavením. Na začiatku bude popísaný dron, ktorý bol navrhnutý v spolupráci s firmou Airvolute za účelom vykonávania automatických inventúr v skladovom priestore.

Dôležitou súčasťou tejto práce je pripraviť si vhodné vývojové a testovacie prostredie, keďže testovanie rôznych algoritmov môže viesť k zničeniu lietajúceho prostriedku. Na vytvorenie prostredia bude využitý fotorealistický simulátor Isaac Sim a na simulovanie letu drona Pegasus Simulátor, ktoré boli spomenuté v oddieli 5.2.3. Pegasus Simulátor pritom musí byť rozšírený, aby sprístupnil potrebné dáta a príkazy do prostredia ROS2, ktoré bude slúžiť ako komunikačný medzičlánok medzi prvkami systému.

Aby sa v mape dali určiť potrebné význačné objekty, ktoré potom budú zapísané do sémantickej mapy, bude potrebné natrénovať hlbokú neurónovú sieť na sémantickú segmentáciu. Keďže vhodný trénovací dátový set na tento konkrétny účel neexistuje, využije sa na to už spomínaný fotorealistický simulátor Isaac Sim a jeho rozšírenie Replicator. Na trénovanie DNN bude použitý nástroj TAO Toolkit z oddielu 6.3.3, ktorý zjednodušuje ako proces trénovania, tak aj nasádzania na cieľové zariadenie.

Mapa prostredia bude tvorená autonómne pomocou UAV, na čo je potrebné použiť vhodný algoritmus prehľadávania neznámeho prostredia. Na tento účel je najvhodnejší FUEL z podkapitoly 4.6, ktorý dokáže v reálnom čase navigovať dron pre úplné prebádanie daného prostredia. Pre fungovanie potrebuje poznať aktuálnu pozíciu UAV v priestore ako aj vidieť hĺbku prostredia. Tieto nástroje musia rovnako vedieť fungovať v reálnom čase na palubnom počítači drona, preto bude daný dôraz na ich výpočtovú náročnosť. Ako zdroj lokalizácie bude použitá stereo vizuálna odometria a na hĺbkový obraz sa použije model umelej inteligencie, schopný fungovať v skladovom prostredí.

Ďalším krokom je vytvorenie globálnej sémanticky segmentovanej hĺbkovej mapy skladu, ktorá v sebe bude obsahovať farebné kódy jednotlivých typov objektov, na ktoré je natrénovaná DNN pre sémantickú segmentáciu. Na tvorbu mapy bude použitá knižnica VoxBlox, spomenutá v podkapitole 4.3, ktorá vytvára TSDF reprezentáciu mapy a podporuje jej exportovanie do mračna bodov v PLY formáte. Na mapovanie priestoru je potrebný hĺbkový obraz v spojení s lokalizáciou, takže rovnaké vstupy ako prieskumný let. Navyše je však mapovanie obohatené o sémantickú informáciu, na čo bude použitá hlboká neurónová sieť a vykonávaná na dedikovanom hardvérovom akcelerátori pre efektívne využitie palubného počítača. Hĺbkové a sémanticky segmentované dáta budú zarovnané na rovnakú kameru a tak sa bude dať priamo spojiť 3D bod v priestore so sémantickou informáciou.

Z vytvoreného zafarbeného mračna bodov v PLY formáte bude vytvorená sémantická mapa prostredia vhodná pre aplikáciu inšpekcie skladov. Z mračna bodov budú extrahované: steny, podlaha, regály, a statické prekážky. Na základe získaných objektov do mapy pribudnú pristávacie/vzletové oblasti a lokalizačné značky.
Vyhodnotenie riešenia bude demonštrované pomocou naplánovania inšpekčného letu na vytvorenej JSON sémantickej mape.

3 Robotické UAV

Táto práca je vykonávaná v spolupráci so spoločnosťou Airvolute s.r.o. [52], ktorá sa špecializuje na návrh, výrobu a implementáciu bezpilotných autonómnych dronov použiteľných vo vnútorných priestoroch. Jeden takýto dron je model Discovery Indoor, ktorý je zobrazený na Obrázok 2. Typickými aplikáciami, ktoré takýto dron dokáže vykonávať sú: skladová inventúra, inšpekcia priestoru, zásahové vozidlo, mapovanie priestoru.

Obrázok 2: Dron použitý pre účely tejto dizertačnej práce.

Konkrétny dron Discovery Indoor je X-frame kvadrokoptéra s rozpätím krídiel 600mm a dĺžkou vrtúľ 10″. S vysoko-napäťovou šesťčlánkovou lítium polymérovou (LiPo) batériou o kapacite 8200mAh dokáže lietať až 30 minút vkuse, pričom dosahuje hmotnosť do dvoch kilogramov.

Jadrom tohto drona je autopilot doska DroneCore 1.2 tiež od firmy Airvolute [2], ktorá zabezpečuje celú elektroniku v drone a konektivitu k senzorom a iným komponentom. Na autopilot doske sa nachádza Cube Orange^{+} od CubePilot Pty. Ltd. [3] s ArduPilot firmvérom. Zároveň je tam výkonný vnorený počítač Nvidia Jetson Xavier NX [4] s grafickou kartou Volta a hardvérovými akcelerátormi pre prácu s obrazovými dátami a hlbokými neurónovými sieťami.

Dôležitým senzorickým vybavením sú dve stereo-kamery, jedna dopredu a jedna dozadu, určené pre algoritmus vizuálnej odometrie a tvorby hĺbkového obrazu. Do všetkých

smerov sú umiestnené laserové senzory merania času letu svetla (TOF) s 27° zorným polom pre detekcie prekážok v prostredí. Pre aplikáciu inšpekcie skladu je dron vybavený 12MPx kamerou s globálnou uzávierkou a silným kužeľovým svetlom.

4 Príprava simulovaného prostredia skladu

Prvým krokom pre vývoj autonómneho lietajúceho stroja za účelom tvorby globálnej mapy prostredia pre efektívne navigovanie sa, je vytvoriť si dané prostredie. Aby tento systém nebol príliš napasovaný na jeden typ prostredia, ktoré je k dispozícii v reálnom svete, cieľom bude vytvoriť si vhodné a dostatočne variabilné prostredie pre zbieranie potrebných dát a testovanie navrhnutého systému.

4.1 Pegasus Simulátor

Pegasus Simulátor [5] je simulátor kvadrokoptér postavený na NVIDIA® Isaac Sim [6] fotorealistickom simulátore. Isaac Sim dokáže naplno využiť možnosti moderných grafických kariet od NVIDIA pre zobrazenie vierohodného prostredia, podporuje rozsiahle možnosti úpravy svetelných podmienok scény, fyzika prostredia využíva grafickú akceleráciu s nástrojom PhysX, umožňuje tvorbu syntetických dátových setov na trénovanie hlbokých neurónových sietí, a je úzko prepojený s ekosystémom NVIDIA Omniverse. Pegasus Simulátor je doplnok k Isaac Sim, ktorý využíva Python API pre tvorbu simulačného prostredia a lietajúceho stroja.

Na použitie Pegasus Simulátora za účelom vyvíjania mapovania vnútorného priestoru boli spravené tieto zmeny a ponúknuté autorovi projektu pre rozšírenie funkcií tohto simulátora:

- Pridanie lokalizácie na základe vizuálnej odometrie
- Možnosť pridať kamery a meniť ich parametre
- Vybavenie drona senzormi času letu svetla (TOF)
- Výstup senzorov do ROS2
- Pridanie ArduPilot autopilota

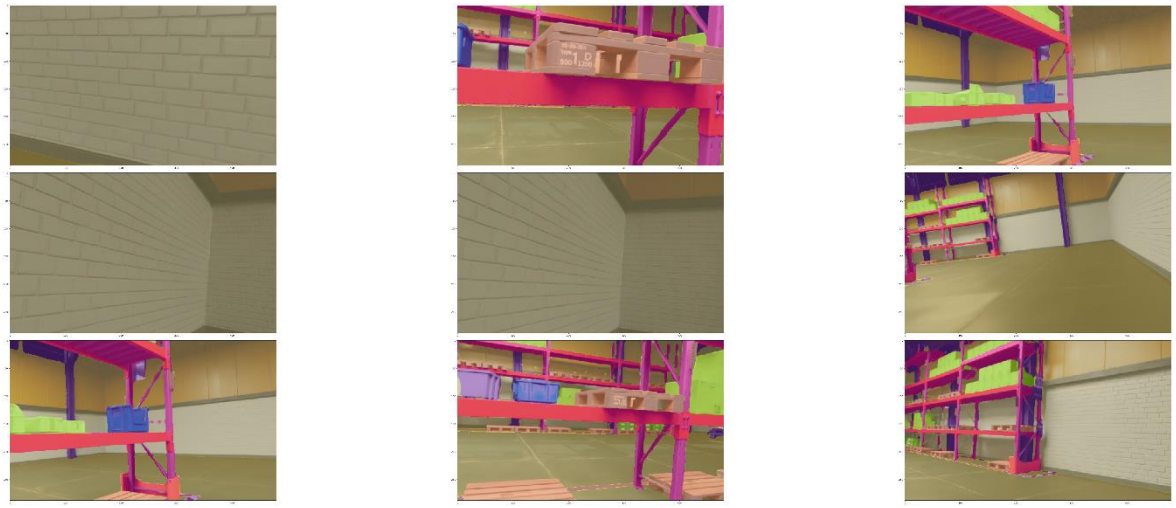
5 Trénovanie sémanticky segmentačnej hlbokej neurónovej siete

Aby sme vedeli tvoriť sémanticky segmentovanú globálnu hĺbkovú mapu skladu, je potrebné mať DNN pre rozpoznanie dôležitých častí skladu z kamier na lietajúcom stroji. Neurónové siete sú pritom stále aplikačne špecializované, neexistuje natrénovaná sieť presne pre dané účely a ani vhodný dátový set.

5.1 Tvorba segmentačného dátového setu

Keďže na tento typ aplikácie je potrebná sémantická segmentácia, je potrebné pre každý obrázok mať vytvorenú masku, kde každý pixel predstavuje identifikačné číslo triedy objektu vo farebnom obrázku. Proces tvorby segmentačného dátového setu sa dá zrýchliť a zjednodušiť s využitím fotorealistického simulátora Isaac Sim a jeho rozšírenia Isaac Sim Replicator.

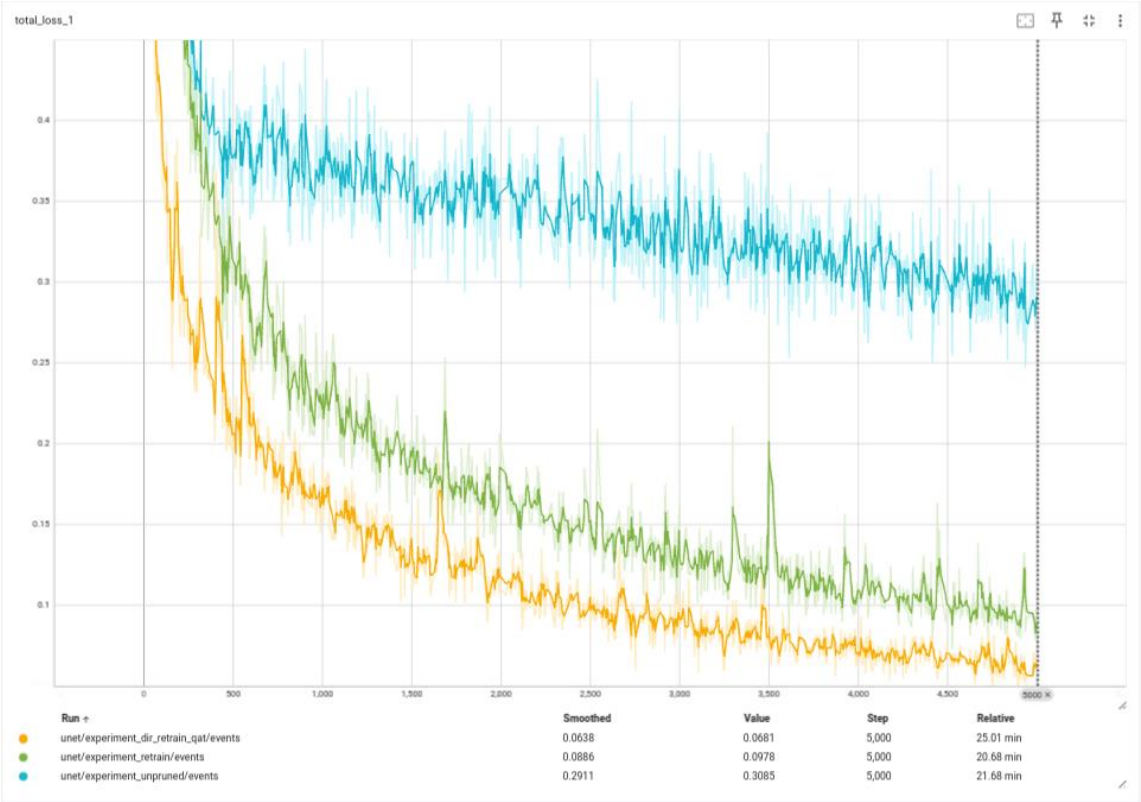
Replicator pridáva možnosť meniť scénu v simulátore na základe daných pravidiel v určitých krokoch a zaznamenávanie dát vhodných pre trénovanie rôznych DNN. Vo vytvorenej scéne skladu boli nasledujúce typy objektov označené sémantickou značkou: regál, paleta, krabica, podlaha, a stena. Modifikovaním parametrov scény ako je pozícia a natočenie kamery, svetelné podmienky, bol vytvorený vhodný trénovací dátový set.



Obrázok 3: Ukážka vytvorených obrázkov z Isaac Sim Replicator. Segmentačná maska s priradenými RGB hodnotami pre každú triedu objektu je preložená cez farebný obrázok.

5.2 Trénovanie DNN pre sémantickú segmentáciu

Na hlboké učenie je použitý nástroj TAO Toolkit, ktorý zjednodušuje proces prípravy, trénovania a nasádzania hlbokých neurónových sietí. TAO Toolkit podporuje prenesenie vedomostí (transfer learning) dvoch známych modelov pre sémantickú segmentáciu, a to U-NET a SegFormer. V tejto práci je využitý model U-NET, keďže je dôležité jeho fungovanie na vnorenom zariadení Jetson Xavier NX v reálnom čase a dostatočným vzorkovaním. Rozlíšenie trénovaného modelu bolo prispôsobené rozlíšeniu použitého modelu light-ess pre určovanie disparity, o ktorom je písané v ďalšej kapitole.



Obrázok 4: Porovnanie priebehu funkcie straty pre trénovanie (modrá), dotrénovanie bez kvantovane vedomého trénovania (zelená) a dotrénovanie s kvantovaním (žltá)

Proces trénovania pozostáva z viacerých krokov. Najprv je už prednaučený model na inom datasete preučený na dáta pre segmentáciu objektov v sklade. Výsledná hodnota funkcie straty po trénovaní bola 0.3085 a trénovanie trvalo 22 minút, ako vidno na Obrázok 4. Pri evaluácii dosahoval tento model presnosť detekcie pozitívnych vzoriek (recall) 77.9%, vierohodnosť detekcie pozitívnych vzoriek (precision) 78.8%, harmonický priemer presnosti a vierohodnosti (F1 score) 78.2%, a priemer IOU 66.8%.

Pre zefektívnenie modelu sa používa orezávanie, kedy sú prepojenia a neuróny, ktoré nie sú potrebné pre dané natrénované využitie potrebné, odstránené. Počas orezávania sa však znižuje presnosť modelu, avšak orezaný model je znova dotrénovaný pre navrátenie jeho presnosti. Dotrénovanie bolo vykonané dvoma spôsobmi, kvantovane vedomé trénovanie (quantization aware traning) a bez toho. Použitie kvantovania vedomého trénovania je vhodné pre účely, kde model bude inferovaný s redukovanou presnosťou váh, ako je FP16 a INT8. Dotrénovanie bez kvantovania vedomého trénovania dosiahlo hodnotu funkcie straty 0.0978 za 21 minút a s touto metódou stratu 0.0681 za 25 minút. Tým je vidno, že síce proces trénovania sa touto metódou predĺžil, dosahuje však lepšie výsledky, ktoré sa taktiež odzrkadlia na finálnom produkte. Evaluácia kvantovane vedome trénovaného modelu dosahovala výborné výsledku, a to recall 95.3%, precision 94.7%, F1 score 95% a mean IOU 90.6%.

6 Autonómna tvorba mapy skladu

Aby bolo možné autonómne tvoriť mapu skladu pomocou lietajúceho vozidla, je potrebné prepojiť viaceré algoritmy. Zároveň je podstatné, aby tieto algoritmy boli efektívne a teda mohli byť použité na lietajúcom robote s obmedzeným výpočtovým výkonom. Vďaka využitiu simulátora Isaac Sim, vieme tieto časti najprv vyvinúť a otestovať samostatne na simulovaných dátach, potom na reálnych a až potom všetko prepojiť dokopy. To výrazne zjednoduší prácu a zároveň zabezpečí správne fungovanie jednotlivých častí.

6.1 Prehľadávací let

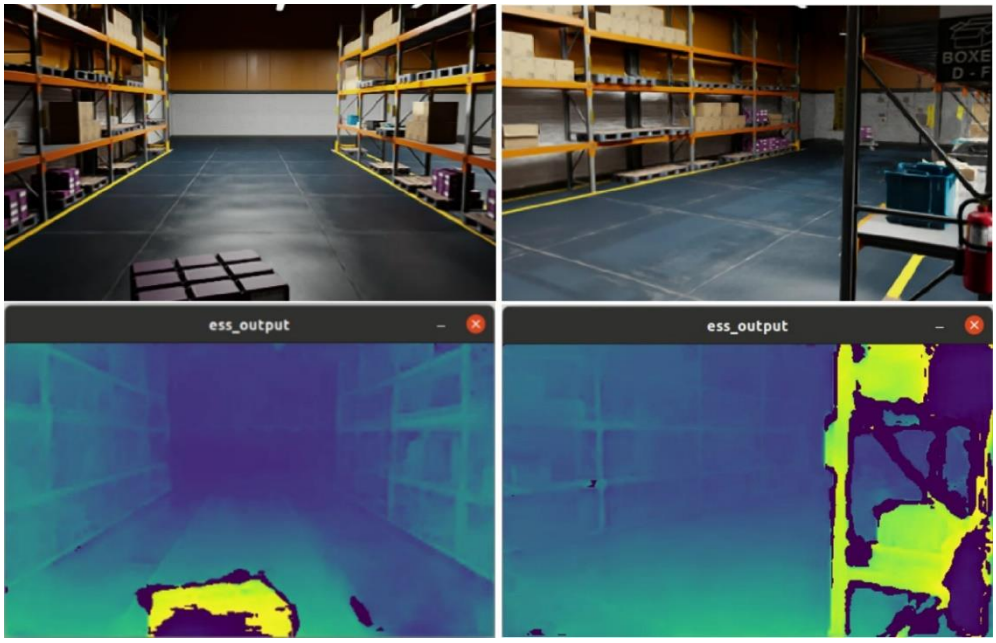
Základom je možnosť navigovať UAV pre objavovanie daného prostredia, na čo bol vybratý algoritmus rýchleho prieskumného letu FUEL [7]. Je to momentálne state-of-the-art pre rýchle prehľadávanie vnútorných priestorov pomocou autonómnych dronov. Aby dokázal fungovať tento prehľadávač priestorov, bolo potrebné ho upraviť a dať mu podstatné vstupy. Prvým krokom bolo prepísanie celého algoritmu do ROS2 Humble (z ROS1 Kinetic), aby dokázal komunikovať so simulátorom a zvyšnými algoritmami. Vstupnými dátami sú pritom odometria drona a hĺbkový obraz a výstupom je publikovanie nových príkazov pozície a natočenia pomocou Mavlink správy *SET_POSITION_TARGET_LOCAL_NED*.

6.2 Vizuálna odometria

Ako zdroj vizuálnej odometrie je použitá Nvidia Elbrus [8] knižnica, ktorá implementuje hardvérovo akcelerovanú simultánnu lokalizáciu a mapovanie pomocou stereo vizuálnej inerciálnej odometrie, dosahujúca 113fps pri 720p vstupnom obraze na palubnom počítači. Sklad je veľmi špecifické prostredie, kde je bežné, že sa často opakuje rovnaký typ objektu, či je to regál alebo krabica s tovarom. Preto je bežné, že výrazný prvok je spojený s iným objektom ako ten, na ktorom bol identifikovaný, čo skôr pridáva chybu odometrie. Preto za účelom lietania v tomto prostredí je časť mapovania vypnutá.

6.3 Hĺbkový obraz

Vo vnútornom prostredí sa objekty skôr nachádzajú v blízkosti robota, preto využitie stereo kamery pre tvorbu hĺbkového obrazu zvyšuje jeho presnosť do tejto vzdialenosti. Pre získanie disparitného obrazu bola použitá metóda založená na hlbokých neurónových sieťach. Na tento účel bol vybraný Efficient Supervised Stereo model (ESS) [9], ktorý zo stereo kamery určuje kontinuálnu disparitnú mapu. Existujú dve verzie, ESS 960x576x3 (šírka x výška x farieb) a Light ESS (480x288x3), pričom na Jetson Xavier NX je použiteľná iba Light ESS kvôli výkonu tohto zariadenia. ESS dosahuje iba 16 fps pri vysokom vyťažení zariadenia zatiaľ čo Light ESS vie ísť 40 fps s ponechaním dostatočného výkonu pre iné algoritmy.

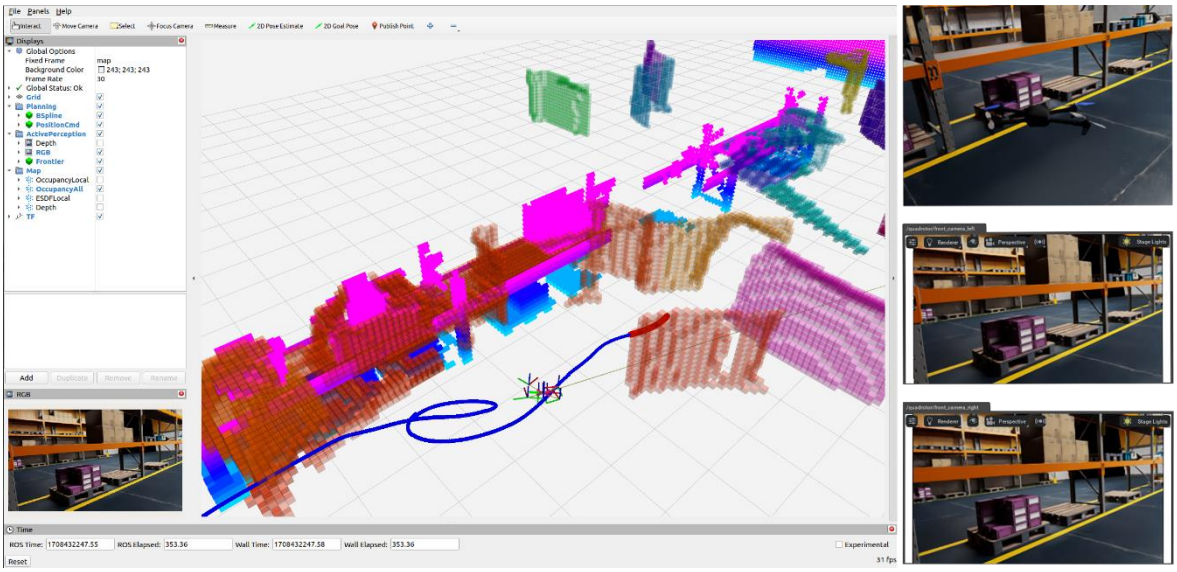


Obrázok 5: Hĺbkový obraz zo záznamov vytvorených v Isaac Sim simulátore pomocou Light ESS modelu stereo disparity a prahu istoty 0.9

6.4 Preskúmaná mapa skladu

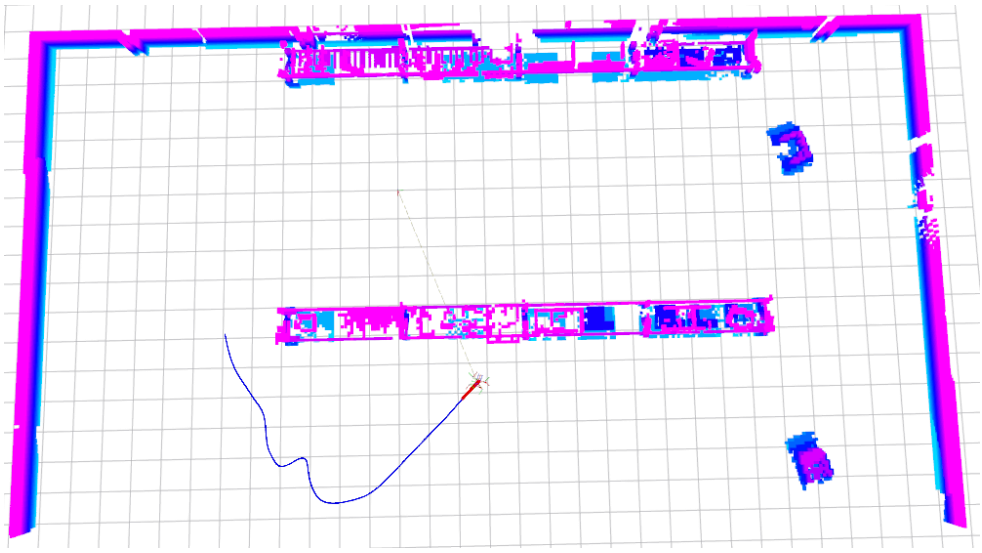
Vytvorený prehľadávací algoritmus bol použitý pre tvorbu mapy skladu. Pre účely tejto práce bol použitý model z IsaacSim nazvaný Sklad s poličkami (Warehouse with shelves). Tento model obsahuje dostatok variability pre otestovanie princípu mapovania: regál dostupný iba z jednej strany, regál dostupný z oboch strán, statické prekážky.

Keďže prenos ROS2 dát medzi Isaac Sim simulátorom a Jetson zariadením nie je dostatočne stabilný a rýchly, je tento proces rozdelený na dve časti. Najprv sa v simulovanom prostredí vykoná prehľadávací algoritmus, aby bol zmapovaný celý priestor skladu. Počas mapovania sú všetky potrebné ROS2 dáta nahrávané. Zaznamenaný prieskumný let s obrazovými dátami sa dá potom použiť pre tvorbu globálnej sémanticky segmentovanej hĺbkovej mapy skladu bez potreby napojenia na simulátor.



Obrázok 6: Proces mapovania skladu zobrazený v RVIZ. V RVIZ je zobrazený dron s aktuálne prejdenou dráhou, zmapovanou časťou regála a hranicami zmapovaného prostredia. Vpravo sú z hora zobrazené: pohľad na dron v Isaac Sim, pohľad z ľavej kamery, pohľad z pravej kamery.

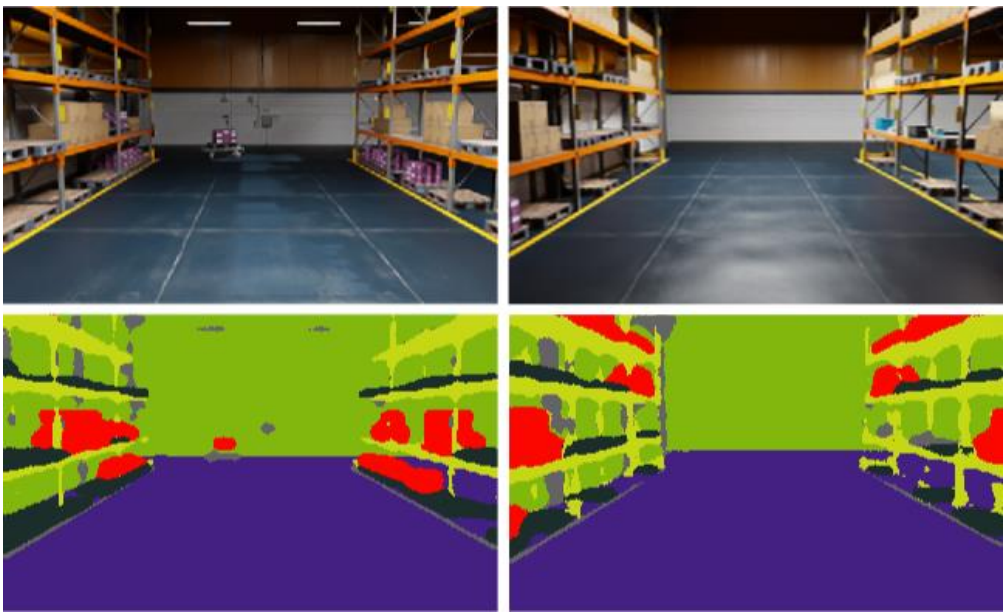
FUEL prehľadávací algoritmus bol navrhnutý pre let iba v jednej letovej výške, preto je výsledný prehľadaný priestor skladu obmedzení na výške. Výsledok je možné vidieť na Obrázok 7.



Obrázok 7: Zobrazenie zmapovaného skladu v RVIZ.

6.5 Sémanticky segmentovaný obraz

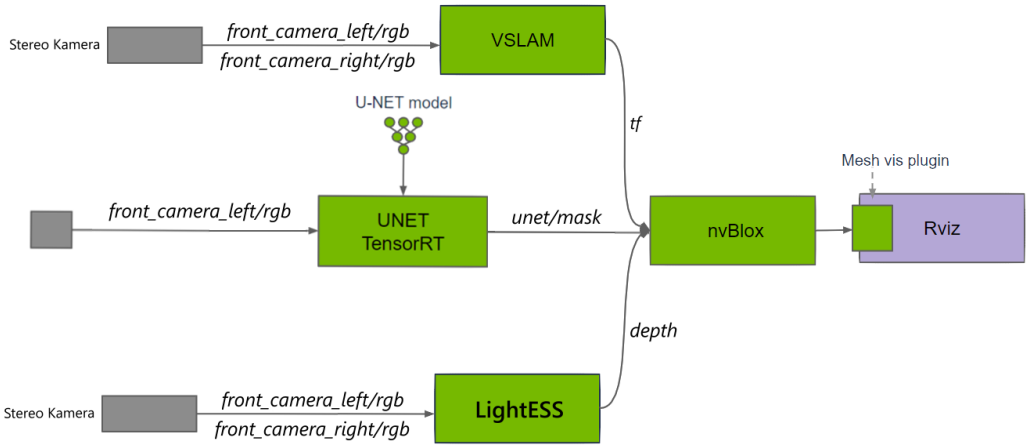
Keďže pre účely sémantickej segmentácie obrazu bol použitý model U-NET, ktorý je priamo podporovaný nástrojmi od NVIDIA, dá sa taktiež jednoducho tento pretrénovaný model na segmentáciu objektov v sklade nasadiť na zariadenie Jetson Xavier NX. Pre nasadenie bol natrénovaný model konvertovaný do TensorRT [10] modelu, ktorý optimalizuje daný model na nasadenie na platformu podporujúcu Nvidia CUDA. Vďaka tomu sa dá využiť hardvérová akcelerácia a zrýchliť tak inferenciu sietí až 40-násobne.



Obrázok 8: Sémanticky segmentovaný obraz z natrénovanej U-NET siete na záznamoch z Isaac Sim.

6.6 Globálna sémanticky segmentovaná mapa skladu

Na vytvorenie globálnej sémanticky segmentovanej mapy skladu je použitý graficky optimalizovaný nástroj VoxBlox [11], ktorý vytvára TSDF reprezentáciu mapy. Odometria vo forme transformácií *map->odom->base_link* určuje aktuálnu pozíciu robota v 3D priestore. Zo stereo páru kamier sa získava disparitná mapa a z nej sa následne vypočítava hĺbková mapa, ktorá určuje videné objekty. Spojením pózy drona a hĺbkovej mapy dostávame globálnu hĺbkovú reprezentáciu mapy. Tá sa pomocou voxblox ukladá vo forme TSDF do generovanej mapy.



Obrázok 9: Schéma prvkov pre vytvorenie globálnej sémantickej mapy skladu.

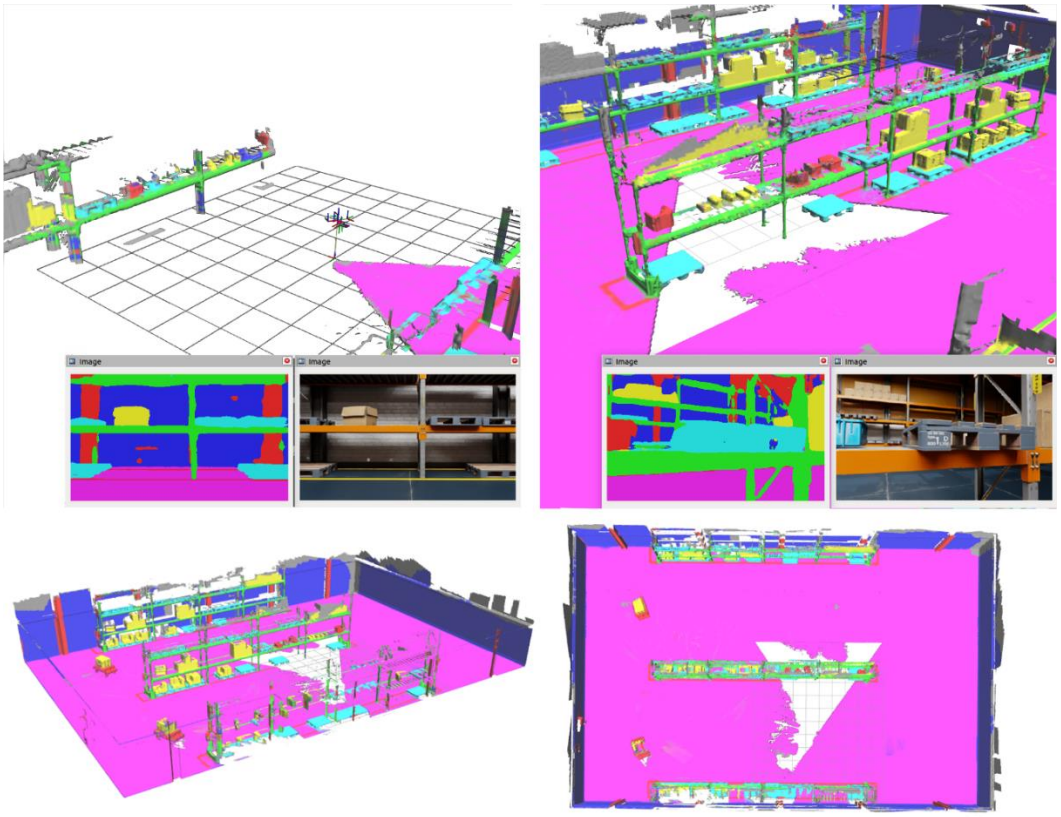
TSDF reprezentácia navyše podporuje ukladanie farby pre každý bod mapy. To je dôležité, lebo farebný obraz z kamery môže byť ľahko nahradený farebnou segmentačnou maskou a tak vznikne globálna sémanticky segmentovaná mapa skladu, ako je vidno na Obrázok 10. Táto mapa sa dá následne uložiť do formátu .ply, užitočnú pre ďalší krok.

6.7 Výsledný systém autonómneho mapovania

Pre prepojenie týchto prvkov boli použité balíčky Nvidia Isaac ROS, ktoré využívajú spomínané algoritmy a využívajú hardvérovú akceleráciu pre ich zrýchlenie funkčnosti na vnorenom zariadení Nvidia Jetson Xavier NX. Balíčky Nvidia Isaac ROS vyžadujú ROS2 Humble, na čo bol použitý docker na zariadení (Jetson Xavier NX nemá priamu podporu ROS2 Humble). Pre komunikáciu medzi prvkami sa využíva Nitros, čo odstraňuje nutnosť neustáleho kopírovania medzi CPU a GPU pamäťou pri prenose obrazových dát z jedného prvku do druhého, čím sa výrazne zefektívňuje tento proces.

Aby dokázal celý systém fungovať na zariadení v reálnom čase, boli upravené vzorkovania jednotlivých častí. Vizuálna odometria, hĺbkový obraz a segmentačný obraz boli tvorené 40 fps zatiaľ čo mapovanie a plánovanie prieskumného letu na 5 fps, čo vyťažilo systém na približne 80 percent. To dáva vôľu pre náhle výpočtové špičky, ktoré by mohli nevhodne obmedziť kritické systémy v lietajúcom drone.

Takto navrhnutý systém bol overený pripojením Jetson zariadenia cez ethernet s počítačom, na ktorom išiel simulátor. Zenoh DDS bol použitý pre efektívne posielanie ROS2 správ z počítača do palubného počítača. Celý systém prehľadávania a mapovania simulovaného prostredia bolo vykonávané na Jetson zariadení Xavier NX v reálnom čase.



Obrázok 10: Výsledná globálna sémanticky segmentovaná mapa skladu z rôznych uhlov pohľadu.

7 Sémantická mapa skladu

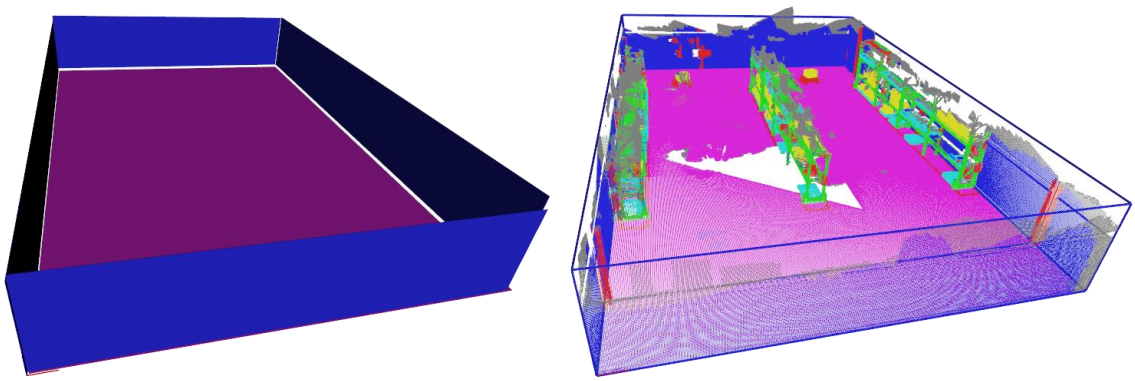
Prostredie skladu je štruktúrované, so známymi typmi objektov a zväčša rovnomerným rozložením, ktoré uľahčuje organizovanie tovaru. Pre aplikáciu inšpekcie skladových

priestorov potrebujeme vedieť, kde sa nachádzajú plochy záujmu, ktoré chceme skontrolovať. Na základe zvolených inšpekčných plôch sa tvorí globálna trajektória drona. Zároveň z tejto mapy vieme určiť rozmery skladu a statické prekážky, ktoré sa v ňom nachádzajú. Táto sémantická mapa teda poslúži aj na vytvorenie mapy prostredia vhodnú pre globálne plánovanie pre vytvorenie optimálnej trajektórie k zvoleným inšpekčným plochám s vyhnutím sa prípadných statických prekážok. V tejto kapitole je spracovaný PLY model pomocou PCL knižnice [12] pre vytvorenie sémantickej mapy skladu v JSON formáte. Tento JSON formát pritom obsahuje nasledovné uložené informácie:

- hlavička súboru – identifikácia mapy
- rozmery priestoru – polygón ohraničujúci pracovný priestor
- informácie o drone – typ a rozmery použitého drona
- nabíjacie/pristávacie stanice – miesta určené na vzlet a pristátie
- lokalizačné značky – značky určené pre periodické spresnenie pozície drona
- prekážky – známe statické prekážky
- regály – každé prístupné poschodie regála predstavuje jedno inšpekčné miesto

7.1 Nájdenie ohraničenia priestoru mapy

Keďže máme farebne odlíšené jednotlivé typy objektu záujmu v načítanom mračne bodov, je výrazne zjednodušený krok odfiltrovania jednotlivých častí od seba. Pre nájdenie ohraničenia priestoru bude potrebné oddeliť podlahu a jednotlivé steny a následne z nich vytvoriť celistvý priestor skladu.



Obrázok 11: Nájdené ohraničenie priestoru mapy tvoriace z podlahy a štyroch stien. Naľavo sú zobrazené jednotlivé steny a podlaha. Napravo modré čiary zobrazujú ohraničenie po spresnení rohových bodov

Pre nájdenie podlahy bol najprv použitý farebný filter podľa farebného kódu zo sémantickej segmentácie. Následne zvyšné body boli segmentované modelom *SACMODEL_PLANE* a rovnica roviny bola iteratívne optimalizovaná metódou *SAC_RANSAC*.

Podobným spôsobom sú spracované aj steny skladu. Po farebnom odfiltrovaní pomocou farebného kódu stien sú podľa hrán podlahy oddelené jednotlivé rovné časti stien geometrickým filtrom. Následne je použitá rovnaká metóda segmentácie pre získanie rovnice roviny steny.

Spoje stien a podlahy sú následne optimalizované pomocou hľadania priesečníka troch rovín (podlaha, stena a stena). Rovnako boli optimalizované aj spoje susediacich stien, aby sa pretínali v rovnakej výške. Na základe optimalizovaných bodov bol 3D model transformovaný tak, aby podlaha mala čo najmenšiu zmenu v osi *z* a počiatok os *x* a *y* bol v jednom z rohov priestoru.

Algoritmus 1: Nájdenie ohraničenia priestoru

```
Vstup: Mračno bodov vytvorenej mapy priestoru
Výstupy: Rozmery mapy v JSON formáte
         Ohraničujúce kvádre podlahy a stien pre hľadanie prekážok
1:  Odfiltruj mračno bodov podlahy podľa farby → FilterPointCloudByColor(mračno bodov, RGB farba, okrajová hodnota)
2:  Vypočítaj rovnicu roviny podlahy → PlaneEquation(mračno bodov, vzdialenostný prah, počet iterácií)
3:  Vypočítaj ohraničujúci kváder podlahy → GetPlaneBoundingBox(mračno bodov, rovnica roviny, vzdialenostný prah)
4:  Odfiltruj mračno bodov stien → FilterPointCloudByColor
5:  for horizontálnu hranu podlahy do
6:      Vyber oblasť v okolí hrany podlahy → FilterPointCloudBy-
        Position(mračno bodov, minimálny bod, maximálny bod)
7:      Vypočítaj rovnicu roviny steny → PlaneEquation
8:      Vypočítaj ohraničujúci kváder steny → GetPlaneBoundingBox
9:  end
10: for horizontálny rohový bod podlahy do
11:     Nájdi bod prieniku rovníc podlahy a dvoch stien →
        ThreePlaneIntersection(rovnica 1, rovnica 2, rovnica 3)
```

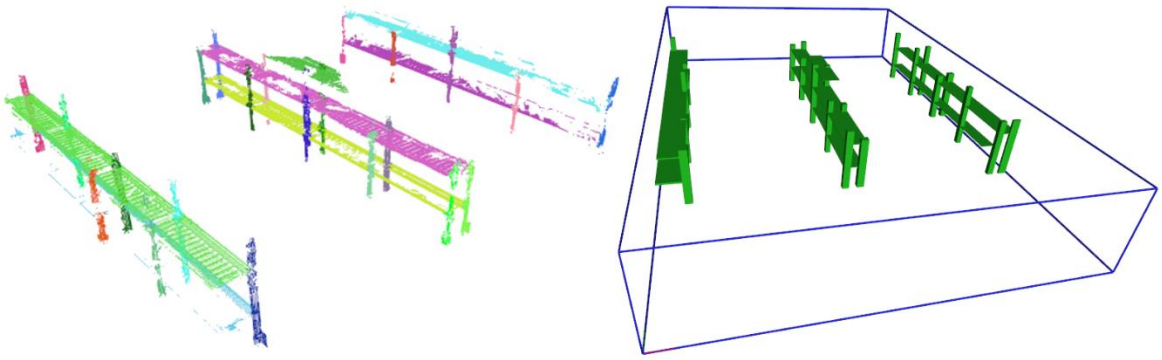
```
12:  end

13:  Vypočítaj z bodov prienikov nové minimálne a maximálne
     rozmery priestoru
14:  Zapíš do Json::Value hodnotu výšky stropu, limity priestoru
     a body polygónu podlahy
```

7.2 Nájdenie inšpekčných oblastí v regáloch

Regály majú výrazne zložitejšie mračno bodov ako steny alebo podlaha. V modeli sa môže nachádzať niekoľko regálov, ktoré môžu mať rôzne pozície a natočenia. Taktiež tieto regály môžu mať rôzne konfigurácie poschodí, dĺžok, hĺbok a podobne. Regály sú kvádrového tvaru, kde každý má štyri nohy a niekoľko vodorovných podlaží. Viacero regálov môže byť spojených dohromady, čím sa dve ich nohy dotýkajú, vytvárajúc jednu hrubšiu nohu. Je teda potrebné vedieť rozlíšiť jednotlivé regály od seba a zároveň aj pomocou horizontálnych platní určiť ich výšku a počet poschodí.

Rovnako ako v predchádzajúcej podkapitole, môžeme vychádzať z mračna bodov, ktoré je farebne rozlíšiteľné od zvyšku modelu. Tým vzniknú v tomto modeli tri veľké zhluky bodov, ktoré boli od seba oddelené a malé zhluky bodov boli odfiltrované.



Obrázok 12: Nájdené podlažia a nohy regálov. Naľavo je každé nájdené podlažie alebo noha oddelené inou farbou a napravo sú nahradené kvádrami o ohraničujúcich rozmeroch daného mračna bodov.

Nasledujúcim krokom je oddelenie podlaží z jednotlivých radov regálov. Podlažie sa dá brať ako horizontálna platňa, pričom každý rad regálov môže obsahovať niekoľko takýchto podlaží. S použitím segmentačného modelu *SACMODEL_PERPENDICULAR–_PLANE*, osou *z* a optimalizačnou metódou *SAC_RANSAC*, sú od seba oddelené jednotlivé podlažia.

Podobným spôsobom sú aj nájdené jednotlivé nohy regálov. Nohy regálov sa dajú predstaviť si ako vertikálne čiary o nejakej hrúbke. Na takýto typ segmentácie slúži v PCL knižnici model *SACMODEL_PARALLEL_LINE*.

Algoritmus 2: Nájdenie rovnobežných plôch → *FilterPerpendicularPlanes*

Vstupy: Mračno bodov, os, odchýlka uhla, vzdialenostný prah, maximum iterácií optimalizácie, prah pre vynechanie, prah pre ukončenie

Výstupy: Mračná bodov nájdených plôch a ich rovnice rovín

1: Vytvor kópiu mračna bodov pre účel odstraňovania nájdených plôch

2: while true do

3: Použi *pcl::SACSegmentation* s modelom *pcl::SACMODEL_PERPENDICULAR_PLANE,* kolmou osou, odchýlkou uhla a optimalizáciou segmentácie *PCL::SAC_RANSAC* s iteráciami a vzdialenostným prahom

4: if počet bodov plochy je menší ako prah pre ukončenie do → ukonči hľadanie a vráť nájdené plochy a ich rovnice

5: if počet bodov plochy je menší ako prah pre vynechanie do → odstráň body z hľadaného mračna bodov a pokračuj v hľadaní ďalších plôch

6: Vytvor mračno bodov z nájdenej plochy a odpamätaj

7: Odstráň body z hľadaného mračna a pokračuj v hľadaní ďalších plôch

8: end

9: Vráť nájdené plochy a ich rovnice

Na základe rozloženia nôh regála, kde je potrebné mať aspoň jednu nájdenú nohu oddeľujúcu dané regály od seba, vieme rozdeliť rad regálov na jednotlivé regály. Rovnako sa dá každý regál rozdeliť na jednotlivé podlažia pomocou nájdených horizontálnych podlaží. Inšpekčné oblasti tým pádom sú vyznačené v rámci každého regála až po najvyššie nájdené podlažie, čo aj určuje výšku regála. Avšak nie všetky strany regála sú prístupné pre inšpekciu, preto inšpekčné plochy, ktoré sú blízko okraja priestoru alebo prekážky sú zrušené.

Algoritmus 3: Nájdenie inšpekčných oblastí

Vstup: Mračno bodov vytvorenej mapy priestoru

Výstupy: Regály s inšpekčnými plochami v JSON formáte
Ohraničujúce kvádre regálov pre hľadanie prekážok

1: Odfiltruj mračná bodov regálov → *FilterPointCloudByColor(mračno bodov, RGB farba, okrajová hodnota)*

2: Odstráň neplatné hodnoty NaN s *pcl::removeNaNFromPointCloud*

3: Oddeľ od seba jednotlivé rady regálov hľadaním veľkých zhlukov → *FilterPointCloudToClusters(mračno bodov, tolerancia, minimálna veľkosť, maximálna veľkosť)*

4: for rad regálov do

5: Nájdi všetky roviny rovnobežné s rovinou podlahy → *FilterPerpendicularPlanes(mračno bodov, os, odchýlka uhla, vzdialenostný prah, maximum iterácií optimalizácie, prah pre vynechanie, prah pre ukončenie)*

6: for poschodie regálu do

7: Vypočítaj ohraničujúci kváder poschodia regálu → *GetPlaneBoundingBox(mračno bodov, rovnica roviny, vzdialenostný prah)*

8: end

9: Nájdi všetky paralelné čiary s rovinou podlahy → *FilterParallelLines(mračno bodov, os, odchýlka uhla, vzdialenostný prah, maximum iterácií optimalizácie, prah pre vynechanie, prah pre ukončenie)*

10: for noha regála do

11: Vypočítaj ohraničujúci kváder nohy regála → *pcl::getMinMax3D*

12: end

13: Vypočítaj ohraničujúci kváder radu regálov

14: end

15: Zapíš do *Json::Value* nájdené regály podľa nájdených poschodí a nôh regálov

7.3 Nájdenie statických prekážok

Statické prekážky obmedzujú priestor, v ktorom sa môže UAV pohybovať. Zaujímavé sú však iba tie prekážky, ktoré už nie sú pokryté inými známymi objektami, napríklad regálmi. Preto z mračna bodov sú najprv odstránené: podlaha, steny, regály.

V tomto prípade farebná informácia nepomôže, keďže je potrebné nájsť všetky objekty, ktoré môžu obmedziť let drona. Prekážka je definovaná v JSON reprezentácii ako polygón s výškou. Najprv sú teda nájdené všetky väčšie zhluky bodov a oddelené od seba. Pre získanie ohraničujúceho polygónu, sú všetky body zhluku sploštené do *x* a *y* osi. Na týchto bodoch je použitá metóda hľadania konkávneho trupu, čím je získaný konkávny polygón prekážky.

Algoritmus 4: Nájdenie statických prekážok

Vstupy: Mračno bodov vytvorenej mapy priestoru
Ohraničujúce kvádre nájdených objektov

Výstup: Statické prekážky v JSON formáte

1: Odstráň už nájdené objekty z mračna bodov → *FilterOutPointCloudByPosition(mračno bodov, minimálny bod, maximálny bod, extra okraj)*

2: Nájdi a oddeľ od seba zhluky bodov väčšie ako 200 bodov reprezentujúce statické prekážky → *FilterPointCloudToClusters(mračno bodov, tolerancia, minimálna veľkosť, maximálna veľkosť)*

3: for statická prekážka do

4: Premietni všetky body do roviny XY → *ProjectTo2D(mračno bodov)*

5: Vypočítaj body ohraničujúceho polygónu → *ComputeConcaveHull(2D mračno bodov)*

6: Vypočítaj minimálnu a maximálnu výšku bodov z originálneho mračna bodov statickej prekážky → *pcl::getMinMax3D*

7: Zapíš do *Json::Value* nájdené statické prekážky ako body ohraničujúceho polygónu s výškou

8: end

Obrázok 13: Nájdené statické prekážky. Naľavo je mračno bodov bez už nájdených objektov podlahy, stien, regálov. Napravo sú zobrazené polygóny s výškou reprezentujúce prekážky.

7.4 Pridanie pristávacích plôch a lokalizačných značiek

Pridanie pristávacej plochy je úzko späté s pozíciou lokalizačnej značky, ktorá k nej patrí, keďže vždy pri vzlete musí UAV vykonať relokalizáciu, aby získal presnú globálnu pozíciu. Najjednoduchšie umiestnenie lokalizačnej značky je na priečku regála prvého poschodia, preto na začiatku každého regála bude umiestnená jedna lokalizačná značka. K tejto značke taktiež bude prislúchať štartovacia/pristávacia plocha, ktorá je umiestnená 1.75m od tejto značky tak, aby dron hneď po vzlete na predpísanú výšku mal dobrý výhľad na značku.

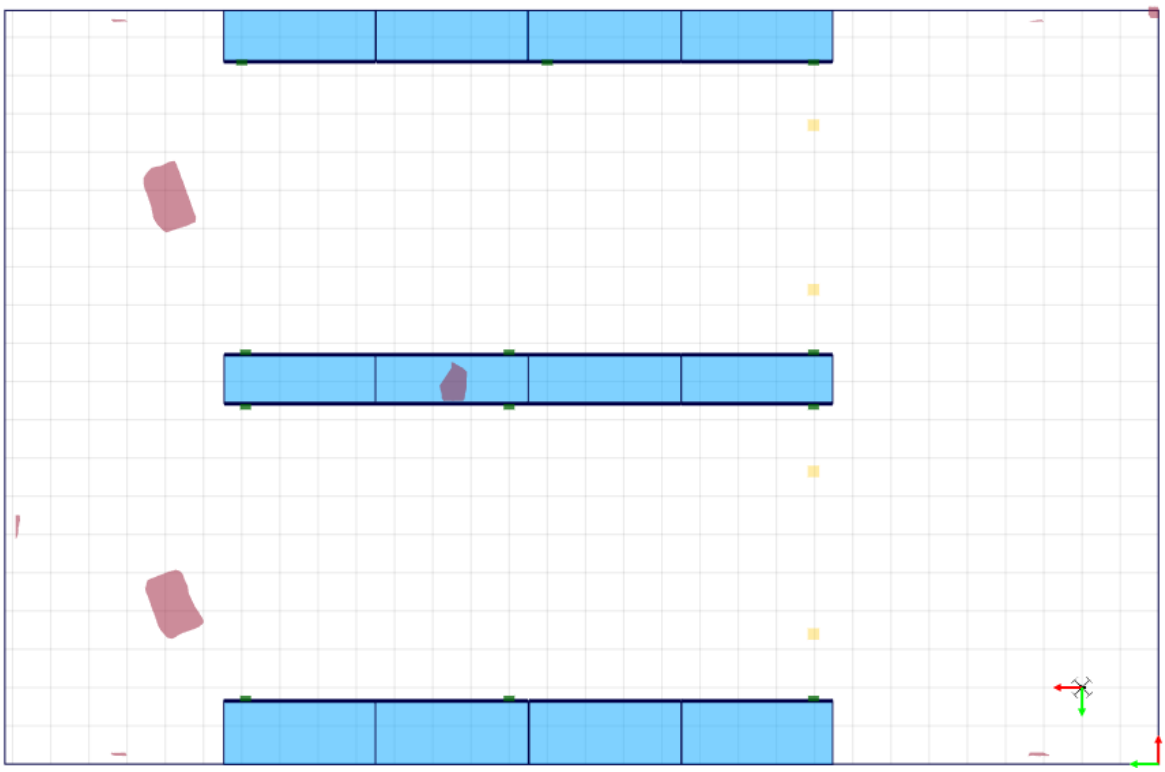
Lokalizačné značky sa využívajú aj na periodickú relokalizáciu, takže táto značka je umiestnená v sklade každých maximálne 10 metrov dĺžky regála a keď naplánovaná trajektória je blízko tejto značky, tak je do trajektórie pridaná procedúra relokalizácie. Lokalizačné značky sú tým pádom rozložené v rovnomerných rozostupoch po celej dĺžke radu regálov tak, aby vzdialenosť nebola väčšia ako 10 metrov a taktiež aby značka nevychádzala na nohu regála, kde sa nedá umiestniť.

Algoritmus 5: Pridaj lokalizačné značky a vzletové/pristávacie plochy

Vstupy: Nájdené rady regálov s rozdelením na regály
Maximálna vzdialenosť medzi lokalizačnými značkami
Minimálna vzdialenosť lokalizačnej značky od nohy regála
Vzdialenosť vzletovej plochy od lokalizačnej značky

Výstup: Lokalizačné značky a vzletové plochy v JSON formáte

```
for rad regálov do
    for inšpekčnú stranu do
        Urči počiatočný a konečný bod prvej priečky poschodia na rade regálov
        Rozdeľ vzdialenosť medzi počiatočným a konečným bodom podľa maximálnej vzdialenosti medzi lokalizačnými značkami pre získanie počtu značiek
        Vypočítaj priemernú vzdialenosť medzi značkami tak, aby značka bola vždy na začiatku a na konci aspoň určitú vzdialenosť od nohy regála
        for pozíciu lokalizačnej značky do
            Urči identifikačné čísla páriu AprilTag značiek, prvé id je poradie radu regálu a druhé je poradie regálu v rade sčítané o inšpekčnú stranu (zprava +100)
            Nájdi najbližšiu nohu regálu a posuň značku tak, aby bola aspoň určitú vzdialenosť od nohy regála
            Zapíš do Json::Value lokalizačné značky s pozíciou, veľkosťou a identifikačnými číslami
        end
    end
end
for prvú lokalizačnú značku v rade regálov do
    Urči pozíciu vzletovej plochy na zemi priamo oproti lokalizačnej značky v určenej vzdialenosti
    Zapíš do Json::Value vzletovú plochu s pozíciou a vzletovou výškou podľa lokalizačnej značky
end
```



Obrázok 14: Výsledná sémantická mapa. Modrou farbou sú označené regále s hrubou stranou vyznačujúcou inšpekčnú stranu, červené sú prekážky, žlté sú pristávacie plochy, a zelené sú lokalizačné značky.

8 Použitie sémantickej mapy pre inšpekciu skladu

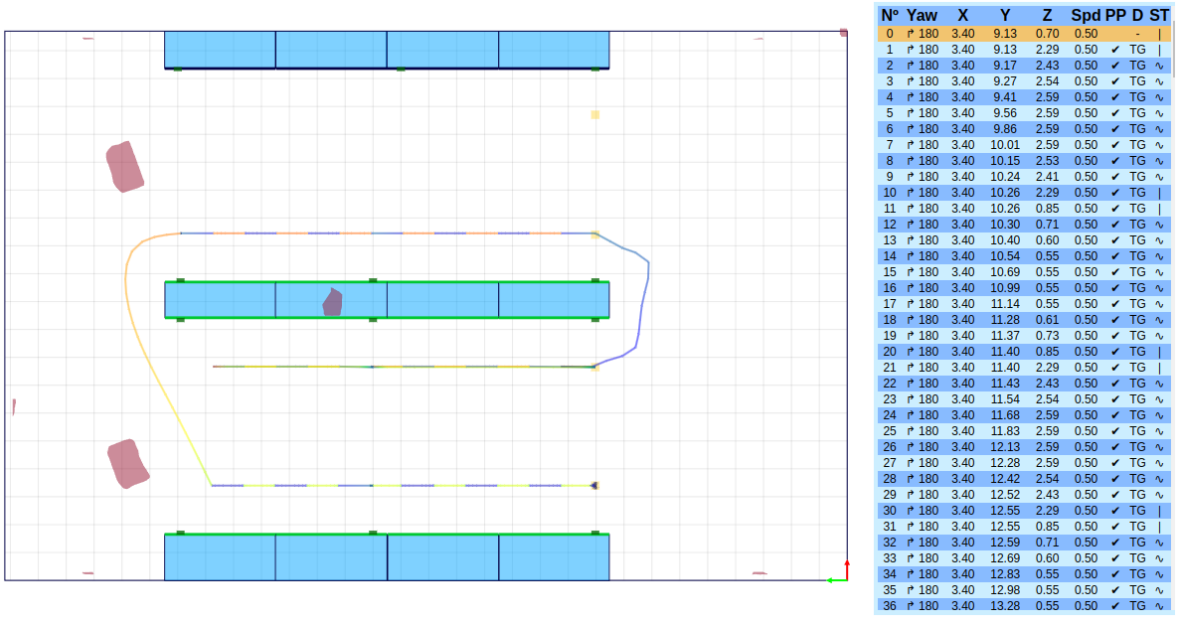
Vytvorená sémantická mapa je vhodná na použitie pre naplánovanie optimálnej a bezkolíznej trajektórie inšpekcie skladu. Detaily algoritmus plánovania tejto trajektórie a popis celej aplikácie inventarizácie skladu je popísaný v článku [1], preto tu je iba prehľad jednotlivých častí, ktorými sú:

1. Vytvorenie oktálovej mapy pre plánovanie
2. Výber parametrov plánovania
3. Naplánovanie trajektórie
4. Vykonanie inšpekčného letu
5. Dekódovanie tovarových štítkov pre vytvorenie zoznamu

V sémantickej mape sú zaznamenané všetky statické objekty, ktoré sa nachádzajú v pracovnom priestore. Všetky tieto objekty sa teda dajú brať ako prekážky pre plánovanie trajektórie. Každý objekt je teda konvertovaný na reprezentáciu polygónovej podstavy s výškou a tak je zaznamenaný do oktálovej mapy.

Webová aplikácia popísaná v publikácii [13] slúži na ovládanie častí potrebných pre vykonanie inšpekcie skladu. V tejto aplikácii je možné spraviť výber inšpekčných plôch, ktoré majú byť skontrolované, pričom dá sa vybrať jedna a viac plôch, takže aj celý rad regálov. Ďalším potrebným vstupom je výber štartovacej pozície drona, od kadiaľ bude začínať naplánovaná trajektória. Na tento účel slúžia pristávacie plochy. Tento výber je následne poslaný do plánovača trajektórie spoločne aj s oktálovou mapou.

Trajektória je plánovaná tak, aby zorné pole aplikačnej kamery pokrylo všetky vybraté inšpekčné plochy. Trajektória pozostáva z vertikálnych letov, počas ktorých sú zaznamenávané skladové štítky. Plánovač taktiež rieši periodické relokalizácie pomocou lokalizačných značiek. Výsledná trajektória je zobrazená v aplikácii v mape a aj v zozname bodov trajektórie.



N°	Yaw	X	Y	Z	Spd	PP	D	ST
0	↱ 180	3.40	9.13	0.70	0.50		-	|
1	↱ 180	3.40	9.13	2.29	0.50	✔	TG	|
2	↱ 180	3.40	9.17	2.43	0.50	✔	TG	∿
3	↱ 180	3.40	9.27	2.54	0.50	✔	TG	∿
4	↱ 180	3.40	9.41	2.59	0.50	✔	TG	∿
5	↱ 180	3.40	9.56	2.59	0.50	✔	TG	∿
6	↱ 180	3.40	9.86	2.59	0.50	✔	TG	∿
7	↱ 180	3.40	10.01	2.59	0.50	✔	TG	∿
8	↱ 180	3.40	10.15	2.53	0.50	✔	TG	∿
9	↱ 180	3.40	10.24	2.41	0.50	✔	TG	∿
10	↱ 180	3.40	10.26	2.29	0.50	✔	TG	|
11	↱ 180	3.40	10.26	0.85	0.50	✔	TG	|
12	↱ 180	3.40	10.30	0.71	0.50	✔	TG	∿
13	↱ 180	3.40	10.40	0.60	0.50	✔	TG	∿
14	↱ 180	3.40	10.54	0.55	0.50	✔	TG	∿
15	↱ 180	3.40	10.69	0.55	0.50	✔	TG	∿
16	↱ 180	3.40	10.99	0.55	0.50	✔	TG	∿
17	↱ 180	3.40	11.14	0.55	0.50	✔	TG	∿
18	↱ 180	3.40	11.28	0.61	0.50	✔	TG	∿
19	↱ 180	3.40	11.37	0.73	0.50	✔	TG	∿
20	↱ 180	3.40	11.40	0.85	0.50	✔	TG	|
21	↱ 180	3.40	11.40	2.29	0.50	✔	TG	|
22	↱ 180	3.40	11.43	2.43	0.50	✔	TG	∿
23	↱ 180	3.40	11.54	2.54	0.50	✔	TG	∿
24	↱ 180	3.40	11.68	2.59	0.50	✔	TG	∿
25	↱ 180	3.40	11.83	2.59	0.50	✔	TG	∿
26	↱ 180	3.40	12.13	2.59	0.50	✔	TG	∿
27	↱ 180	3.40	12.28	2.59	0.50	✔	TG	∿
28	↱ 180	3.40	12.42	2.54	0.50	✔	TG	∿
29	↱ 180	3.40	12.52	2.43	0.50	✔	TG	∿
30	↱ 180	3.40	12.55	2.29	0.50	✔	TG	|
31	↱ 180	3.40	12.55	0.85	0.50	✔	TG	|
32	↱ 180	3.40	12.59	0.71	0.50	✔	TG	∿
33	↱ 180	3.40	12.69	0.60	0.50	✔	TG	∿
34	↱ 180	3.40	12.83	0.55	0.50	✔	TG	∿
35	↱ 180	3.40	12.98	0.55	0.50	✔	TG	∿
36	↱ 180	3.40	13.28	0.55	0.50	✔	TG	∿

Obrázok 15: Zobrazenie naplánovanej trajektórie vo webovej aplikácii. Farba trajektórie zobrazuje výšku v danom mieste, pričom zelená je nízka a červená je vysoká. Napravo je znázornená tabuľka s jednotlivými bodmi trajektórie, zobrazujúca pozíciu, orientáciu, rýchlosť, a typ pohybu v danom bode.

Po vytvorení trajektórie je operátor schopný spustiť inšpekčný let. Dron vtedy nasleduje predpísanú cestu a pomocou DNN detektora skladových štítkov zaznamenáva tieto štítky a ich nekomprimované výrezy ukladá do svojej pamäte.

Po dokončení inšpekčného letu sú výrezy štítkov presunuté na server, ktorý na výkonnejšom počítači dekóduje podstatné informácie zo zaznamenaných štítkov. Vytvorený zoznam nájdeného tovaru je následne možné použiť pre porovnanie s informačným systémom v sklade a vyhodnotiť tak inventúru.

9 Prínosy dizertačnej práce

- Práca prináša jedinečný spôsob autonómnej tvorby statickej mapy prostredia s pridanou sémantickou informáciu. Tento princíp bol demonštrovaný na príprave mapy skladu vhodnej pre aplikáciu inšpekcie tovaru v sklade. Taktiež táto mapa je obohatená o skúsenosti získanými počas tvorby a testovania aplikácie inšpekcie skladov.
- V práci je riešený spôsob prehľadávacieho letu pomocou UAV schopný fungovať priamo na výpočtovej jednotke umiestnenej v drone. Dokáže teda fungovať samostatne v neznámom prostredí bez zásahov operátora alebo externých výpočtových jednotiek.
- Práca demonštruje využiteľnosť fotorealistickej simulácie pre vývoj zložitých algoritmov, ako je autonómna tvorba mapy. Simulátor NVIDIA Isaac Sim je ešte pomerne mladý nástroj, a teda touto prácou sa môže zvýšiť jeho povedomosť o jeho schopnostiach.
- Taktiež práca ukazuje výhodnosť fotorealistickej simulácie pre tvorbu vhodných trénovacích dát pre trénovanie hlbokých umelých neurónových sietí. Takéto trénovacie dáta sú výrazne rýchlejšie na tvorbu, pričom sa dajú aj jednoduchšie upravovať a rozširovať, pričom stále blízko reprezentujú reálny svet.
- Počas tejto práce bol rozšírený simulátor UAV Pegasus Simulator o senzory a rozhrania, čo viedlo aj k nárastu aktivity na tomto projekte.
- Prehľadávací algoritmu FUEL bol prepísaný z ROS1 na ROS2. Tento projekt bol vyvíjaný na už veľmi starej verzii ROS1 Kinetic, rozšírením o ROS2 podporu je výrazne rozšírená životnosť tohto nástroja.

Záver

V tejto dizertačnej práci bol predstavený komplexný návrh tvorby sémantickej mapy za účelom navigovania autonómneho lietajúceho prostriedku v štruktúrovanom prostredí skladu.

Na vytvorenie tejto mapy bolo použité autonómne UAV, ktoré malo za účel preletieť vopred neznámym prostredím skladu a zmapovať ho. Stroj bol vybavený vnoreným počítačom Jetson Xavier NX, ktorý vďaka svojmu výkonu umožňuje spracovávanie vizuálnych senzorov priamo na drone. Na autonómnu tvorbu mapy skladu bol vybraný prehľadávací algoritmus Fast UAV Exploration, ktorý bol napojený na vizuálnu odometriu z Isaac ROS Visual SLAM, hĺbkový obraz z Isaac ROS Stereo Depth. Na tvorbu mapy bol použitý nástroj Isaac ROS NvBlox obohatený o vstupný sémanticky segmentovaný obraz z Isaac ROS Image Segmentation z natrénovanej U-NET hlbokej neurónovej siete na význačné objekty v sklade. Vytvorená sémanticky segmentovaná mapa v PLY formáte bola spracovaná pre vytvorenie sémantickej reprezentácie mapy v JSON textovom formáte.

Celý mapovací algoritmus bol vykonaný na palubnom počítači drona v reálnom čase za použitia fotorealistického simulátora Isaac Sim a jeho rozšírenia pre simulovanie UAV Pegasus Simulátora vo forme softvér v slučke (SITL). Simulátor bol taktiež použitý pre vytvorenie trénovacieho dátového setu pre sémantickú segmentáciu. Na výslednej sémantickej mape bolo demonštrované plánovanie globálnej trajektórie pre aplikáciu inšpekcie tovaru v sklade s možnosťou výberu skenovanej oblasti.

Táto dizertačná práca bola realizovaná v spolupráci s firmou Airvolute s.r.o., ktorá si vyhradila právo vlastníctva vytvorených zdrojových kódov.

Literatúra

[1] J. Stanko, F. Stec, L. Palkovic, J. Rodina a D. Rau, „Towards Automatic Inventory Checking Using an Autonomous Unmanned Aerial Vehicle,“ rev. *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.

[2] Airvolute, [Online]. Available: https://airvolute.com/. [Cit. 29 12 2023].

[3] CubePilot Pty. Ltd., „Most Advanced Cube Yet!,“ [Online]. Available: https://www.cubepilot.com/#/cube/features. [Cit. 29 12 2023].

[4] NVIDIA, „Jetson Xavier NX Series,“ [Online]. Available: https://www.nvidia.com/en-eu/autonomous-machines/embedded-systems/jetson-xavier-nx/. [Cit. 29 12 2023].

[5] M. Jacinto, J. Pinto, J. Patrikar, J. Keller, R. Cunha, S. Scherer a A. Pascoal, „Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation,“ *arXiv preprint arXiv:2307.05263,* 2023.

[6] NVIDIA, „NVIDIA Isaac Sim,“ 2022. [Online]. Available: : https://developer.nvidia.com/isaac-sim. [Cit. 16 12 2023].

[7] B. Zhou, Y. Zhang, X. Chen a S. Shen, „Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning,“ *IEEE Robotics and Automation Letters,* zv. 6, %1. vyd.2, pp. 779-786, 2021.

[8] „Elbrus Stereo Visual SLAM based Localization,“ NVIDIA Corporation, 1 2 2023. [Online]. Available: https://docs.nvidia.com/isaac/archive/2021.1/packages/visual_slam/doc/elbrus_visual_slam.html. [Cit. 29 10 2023].

[9] NVIDIA, „ESS DNN Stereo Disparity,“ 11 12 2023. [Online]. Available: https://catalog.ngc.nvidia.com/orgs/nvidia/teams/isaac/models/dnn_stereo_disparity. [Cit. 4 2 2024].

[10] H. Vanholder, „Efficient inference with tensorrt,“ rev. *GPU Technology Conference,* 2016.

[11] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart a J. Nieto, „Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning,“ rev. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2017.

[12] R. B. Rusu a S. Cousins, „3D is here: Point Cloud Library (PCL),“ rev. *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[13] F. Štec a J. Rodina, „Designing human-machine interface for UAVs in structured environment using ROS and flask,“ rev. *ELITECH´21*, 2021.

[14] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford a I. Reid, „Meaningful maps with object-oriented semantic mapping,“ rev. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] J. McCormac, A. Handa, A. Davison a S. Leutenegger, „SemanticFusion: Dense 3D semantic mapping with convolutional neural networks,“ rev. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Zoznam publikačnej činnosti autora

V2 Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka

V2_01 RAU, Dávid - RODINA, Jozef - ŠTEC, Filip. Generating instant trajectory of an indoor UAV with respect to its dynamics. In *ISMCR 2020 : 23rd International Symposium on Measurement and Control in Robotics. Budapest, Hungary. October 15-17, 2020*. Piscataway : IEEE, 2020, [5] s. ISBN 978-0-7381-4269-2. V databáze: IEEE: 9263769 ; DOI: 10.1109/ISMCR51255.2020.9263769 ; SCOPUS: 2-s2.0-85099595745 ; WOS: 000833324500034.
Kategória publikácie do 2021: AFC

V2_02 STANKO, Jaromír - ŠTEC, Filip - PALKOVIČ, Lukáš - RODINA, Jozef - RAU, Dávid. Towards automatic inventory checking using an autonomous unmanned aerial vehicle. In *ETFA 2022 : 27th International Conference on Emerging Technologies and Factory Automation. Stuttgart, Germany. September 6-9, 2022*. Danvers : IEEE, 2022, [8] s. ISBN 978-1-6654-9996-5. V databáze: DOI: 10.1109/ETFA52439.2022.9921460 ; WOS: 000934103900038 ; SCOPUS: 2-s2.0-85141410119 ; IEEE: 9921460.
Typ výstupu: príspevok z podujatia; Výstup: zahraničný; Kategória publikácie do 2021: AFC

V2_03 ŠTEC, Filip - RODINA, Jozef. Designing human-machine interface for UAVs in structured environment using ROS and flask. In *ELITECH´21 [elektronický zdroj] : 23th Conference of Doctoral Students, May 26, 2021*. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2021, [4] s. ISBN 978-80-227-5098-1.
Výstup: domáci; Kategória publikácie do 2021: AFD

V3 Vedecký výstup publikačnej činnosti z časopisu

V3_01 STANKO, Jaromír - ŠTEC, Filip - RODINA, Jozef. Process automation of warehouse inspection using an autonomous unmanned aerial vehicle. In *MM Science Journal*. October (2022), s. 5864-5869. ISSN 1803-1269(P) (2022: 0.700 - IF, 0.239 - SJR, Q3 - SJR Best Q). V databáze: DOI: 10.17973/MMSJ.2022_10_2022063 ; WOS: 000860510800001 ; SCOPUS: 2-s2.0-85139174041.
Typ výstupu: článok; Výstup: zahraničný; Kategória publikácie do 2021: ADM

O3 Odborný výstup publikačnej činnosti z časopisu

O3_01 DUCHOŇ, František - ČORŇÁK, Marek - TÖLGYESSY, Michal - MRÁZ, Eduard - ŠTEC, Filip - TREBUĽA, Marek - CHOVANEC, Ľuboš. Automatica 2022 očami návštevníkov. In *ATP Journal*. Roč. 29, č. 9 (2022), s. 58-60. ISSN 1335-2237. Typ výstupu: článok; Výstup: domáci; Kategória publikácie do 2021: BDF

D1 Dokument práv duševného vlastníctva

D1_01 STU FEI v Bratislave. 2024. *Zariadenie na určenie polohy robotického prostriedku pomocou páru lokalizačných značiek na magnetickej doske*. Filip ŠTEC et al. Slovenská republika. PUV 62-2024. 12.04.2024

Riešené projekty

Robustná lokalizácia pre drony v priemysle 4.0 (1/0599/20)

Výskum a vývoj využiteľnosti autonómnych lietajúcich prostriedkov v boji proti pandémii spôsobenej COVID-19 (UAVLIFE)

Navigačný stack pre autonómne drony v priemyselnom prostredí (APVV-21-0352)

Štatistika: kategória publikačnej činnosti od 2022

V2	Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka	3
V3	Vedecký výstup publikačnej činnosti z časopisu	1
O3	Odborný výstup publikačnej činnosti z časopisu	1
D1	Dokument práv duševného vlastníctva	1
Súčet		**6**

Štatistika: kategória ohlasov od 2022

1 Citácia v publikácii registrovaná v citačných indexoch				**1**
	Zahraničné		1	
Súčet				**1**