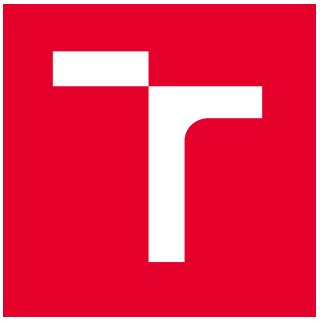


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

VIZUÁLNĚ-INERCIÁLNÍ ODOMETRIE PRO STABILIZACI UAV

VISUAL-INERTIAL ODOMETRY FOR UAV STABILIZATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marco Pintér**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Marcoň, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Marco Pintér ***ID:*** 221010

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Vizuálně-inerciální odometrie pro stabilizaci UAV

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou fúze vizuální a inerciální odometrie pro estimaci přesné polohy bezpilotních letadel typu kvadrokoptéra v 3D prostoru bez signálu GNSS. Proveďte literární rešerši a prostudujte dostupné metody zpracování dat.
2. Na základě rešerše vytvořte nebo zvolte vhodné algoritmy zpracování obrazu, které umožní stabilizaci pozice dronu v prostoru bez zásahu uživatele na základě detekce příznakových bodů okolního prostředí společně s daty z inerciálního systému řídící jednotky UAV.
3. Navrhněte softwarovou infrastrukturu v prostředí robotického operačního systému ROS2, která umožní fúzi inerciálních a vizuálních dat. Proveďte porovnání přesnosti se známou polohou z GNSS systému.
4. Optimalizujte navržené řešení na základě provedených testů a implementujte vizuálně inerciální odometrii pro zpětnovazební řízení polohy UAV.
5. Ověřte funkčnost vytvořeného řešení provedením sady testovacích letů v různých typech reálného prostředí. Vyhodnoťte vlastnosti, přesnost a spolehlivost systému.

DOPORUČENÁ LITERATURA:

SURBER, Julian, Lucas TEIXEIRA a Margarita CHLI. Robust visual-inertial localization with weak GPS priors for repetitive UAV flights. In: 2017 IEEE International Conference on Robotics and Automation (ICRA) [online]. IEEE, 2017, s. 6300-6306 [cit. 2023-09-10]. ISBN 978-1-5090-4633-1. Dostupné z: doi:10.1109/ICRA.2017.7989745

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 15.5.2024

Vedoucí práce: doc. Ing. Petr Marcoň, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Táto diplomová práca sa zaoberá vývojom systému pre estimáciu polohy bezpilotného lietadla v priestore bez použitia GNSS (Global navigation satellite system). K estimácií polohy bezpilotného prostriedku v priestore sú použité dáta z obrazových a inerciálnych senzorov. V práci sú popísané rôzne spôsoby určovania polohy pomocou vizuálnej a inerciálnej odometrie. V ďalšej časti práce je popísaný výber vhodného kamerového senzoru a palubného počítača. Taktiež je popísaný výber vhodného operačného systému, distribúcie ROS-u (Robot Operating System) a vhodného algoritmu pre spracovanie vizuálnych a inerciálnych dát. Pre získavanie telemetrických dát a základné ovládanie bezpilotného prostriedku v priestore bol navrhnutý ROS node (uzol), ktorý bol otestovaný v simulačnom prostredí Gazebo s PX4-SITL. S navrhnutým systémom boli vykonané experimenty porovnania presnosti určovania polohy oproti GNSS v reálnom prostredí. Z dosiahnutých výsledkov je možné konštatovať, že vizuálna odometria dosahuje veľmi dobré výsledky v porovnaní s GNSS.

KĽÚČOVÉ SLOVÁ

Vizuálna odometria, Inerciálna odometria, Estimovanie polohy, GNSS, ROS, Bezpilotné lietadlo, RealSense

ABSTRACT

This master's thesis deals with the development of a system for estimating the position of an unmanned aerial vehicle in space without using GNSS (Global navigation satellite system). Visual and inertial data are employed for the position estimation of the unmanned aerial vehicle in space. The thesis describes various methods of determining the position using visual and inertial odometry. In the subsequent part of the thesis, the selection of a suitable camera sensor and onboard computer is discussed. The choice of an appropriate operating system, ROS (Robot Operating System) distribution, and a suitable algorithm for processing visual and inertial data is also described. To acquire telemetry data and basic control of an unmanned vehicle in space, a ROS node was implemented, which was tested in the Gazebo simulation environment with PX4-SITL. With the designed system experiments of comparing the accuracy of position determination against GNSS was done in real enviroment. From the achieved results, it can be concluded that visual odometry achieves very good results compared to GNSS.

KEYWORDS

Visual odometry, Inercial odometry, Position estimation, GNSS, ROS, Unmanned aerial vehicle, RealSense

Vysadené pomocou balíčku `thesis` verzie 4.07; `http://latex.feec.vutbr.cz`

PINTÉR, Marco. *Vizuálně-inerciální odometrie pro stabilizaci UAV*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024, 101 s. Diplomová práca. Vedúci práce: doc. Ing. Petr Marcoň, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Marco Pintér
VUT ID autora:	221010
Typ práce:	Diplomová práca
Akademický rok:	2023/24
Téma závěrečnej práce:	Vizuálně-inerciální odometrie pro stabilizaci UAV

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi doc. Ing. Petrovi Marcoňovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Úvod **14**

1 Problematika vizuálnej a inerciálnej odometrie **15**
1.1 Vizuálna odometria . . . 15
1.1.1 Relatívna vizuálna lokalizácia . . . 16
1.1.2 Absolútna vizuálna lokalizácia . . . 17
1.2 SLAM . . . 18
1.3 Inerciálna odometria . . . 19

2 Senzory pre získavanie obrazových dát **21**
2.1 Monokulárne kamery . . . 21
2.2 Stereo kamery . . . 22
2.3 HĺbkovÉ kamery . . . 23

3 Systém ROS **25**
3.1 História ROSu . . . 25
3.2 Filozofia . . . 26
3.3 Štruktúra komunikácie ROS1 . . . 27
3.4 Štruktúra komunikácie ROS2 . . . 27
3.5 Súčasti ROS-u . . . 29
3.5.1 Nodes . . . 29
3.5.2 Topics . . . 29
3.5.3 Services . . . 29
3.5.4 Parameter server . . . 30
3.5.5 Nástroje . . . 30
3.6 Distribúcie ROS-u . . . 32

4 Výber hardvéru **33**
4.1 Výber počítaču . . . 33
4.1.1 Raspberry Pi 4 . . . 33
4.1.2 NVIDIA Jetson Nano . . . 34
4.1.3 Intel NUC . . . 34
4.1.4 Porovnanie . . . 35
4.1.5 Parametre zvoleného Intel NUC . . . 36
4.2 Výber kamery . . . 37
4.2.1 HĺbkovÉ kamery Intel . . . 37
4.2.2 Mapovacie kamery . . . 39
4.2.3 Lidar kamery . . . 39

4.2.4 Vstavaná IMU jednotka . . . 40
4.3 Zvolená kamera . . . 40

5 Inštalácia **41**
5.1 Operačný systém . . . 41
5.2 ROS Humble . . . 41
5.3 Ovládače kamery . . . 41
5.4 RealSense Wrapper . . . 41
5.5 RealSense Viewer . . . 42
5.6 Kalibrácia kamery . . . 42

6 Vyčítavanie obrazu z kamery a IMU **43**
6.1 RealSense kamera ROS node . . . 43
6.1.1 Spustenie nodu . . . 44

7 Výber vhodného algoritmu **45**
7.1 RTAB-Map . . . 45
7.1.1 Detekčné algoritmy príznakov . . . 46
7.1.2 Loop closure detection . . . 48
7.1.3 Bag of words . . . 50
7.1.4 Správa pamäte . . . 50
7.1.5 ROS štruktúra RTAB-Map . . . 51

8 Ovládanie bezpilotného prostriedku **52**
8.1 Letová jednotka . . . 52
8.2 Letové režimy . . . 52
8.2.1 Letový režim offboard . . . 53
8.3 Implementácia komunikácie s FC . . . 55
8.4 Protokol MAVLink . . . 55
8.4.1 Point to point . . . 56
8.4.2 Publish-subscribe . . . 56
8.5 Použité triedy MAVSDK . . . 56
8.5.1 Action . . . 56
8.5.2 Telemetry . . . 56
8.5.3 Offboard . . . 56
8.6 Implementácia riadiaceho uzlu v ROS . . . 57
8.7 Implementované servisy . . . 57
8.7.1 Vzlet . . . 57
8.7.2 Pristátie . . . 57
8.7.3 Let na bod . . . 58

8.7.4 Návrat na štart . . . 58
8.8 Exekútor . . . 59
8.8.1 Použitý exekútor . . . 59
8.9 Implementácia telemetrie . . . 59
8.10 Simulácia v prostredí Gazebo . . . 60
8.10.1 Použitý model . . . 61
8.10.2 PX4-SITL . . . 61
8.10.3 Prevod tém z Gazebo do ROSu . . . 61

9 Softvérová štruktúra **63**
9.1 Prepojenie jednotlivých komponentov . . . 63
9.2 Prepojenie kamery a RTAB-Map . . . 63
9.3 Komunikácia s letovým kontrolérom . . . 64

10 Porovnanie s GPS **66**
10.1 Prepočet GPS súradníc na relatívne . . . 66
10.1.1 Vzťahy pre prepočet WGS84 na ECEF . . . 67
10.2 Implementácia prepočtu . . . 67
10.3 Meranie odchýlky medzi trajektóriami . . . 69

11 Experiment porovnania presnosti **70**
11.1 Priebeh prvého experimentu . . . 70
11.1.1 Vyhodnotenie experimentu . . . 72
11.1.2 Klimatické podmienky . . . 74
11.2 Ďalšie merania . . . 74
11.2.1 Prvé meranie . . . 75
11.2.2 Druhé meranie . . . 75
11.2.3 Výsledky merania . . . 76
11.2.4 Klimatické podmienky . . . 77

Záver **78**

Literatúra **80**

Zoznam symbolov a skratiek **84**

Zoznam príloh **85**

A Obsah priloženého média **87**
A.1 ros2_ws . . . 87
A.2 namerane_hodnoty . . . 87

B Inštalácia **88**
B.1 Ubuntu 22.04 . . . 88
B.2 ROS Humble . . . 88
B.3 RealSense SDK 2.0 . . . 88
B.3.1 Inštalácia dalších potrebných ovládačov . . . 89
B.4 RealSense ROS wrapper . . . 89
B.4.1 Príklady spustenia . . . 89
B.5 Zoznam topicov z Realsense wrapperu . . . 89
B.6 RTAB-Map . . . 90
B.6.1 Prílad spustenia . . . 90
B.6.2 Príklad spustenia s vlastným launch file . . . 90

C Simulácia **91**
C.1 Spustenie Gazebo . . . 91
C.2 Spustenie PX4-SITL . . . 91
C.3 Spustenie simulovaných pozičných dát . . . 91
C.4 Spustenie riadiaceho uzlu . . . 91

D Implementované ROS správy a servisy **92**
D.1 Implementované ROS správy . . . 92
D.1.1 Battery.msg . . . 92
D.1.2 Eulerangle.msg . . . 92
D.1.3 Flightmode.msg . . . 92
D.1.4 Gpsposition.msg . . . 92
D.1.5 Rcstatus.msg . . . 92
D.2 Implementované servisy . . . 93
D.2.1 Position.srv . . . 93
D.3 Použité servisy . . . 93
D.3.1 std_srvs\Trigger.srv . . . 93

E Namerané dáta z prvého experimentu **94**
E.1 Histogram chyby v jednotlivých osiach . . . 98

F Namerané dáta z ďalších experimentov **100**
F.1 Prvý experiment . . . 100
F.2 Druhy experiment . . . 101

Zoznam obrázkov

1.1 Príklad vizuálnej odometrie . . . 15
1.2 RVL . . . 16
1.3 AVL . . . 17
1.4 Príklad feature points matching AVL . . . 18
1.5 Príklad 2D mapy pomocou SLAM . . . 18
1.6 Príklad 3D mapy pomocou SLAM . . . 19
1.7 Znázornenie osí inerciálnej jednotky . . . 20
2.1 Raspberry Pi kamera . . . 21
2.2 Príklad merania vzdialenosti stereo kamerou . . . 22
2.3 Príklad skreslenia pri projekcií svetla . . . 23
2.4 Príklad výstupného obrazu hĺbkovej kamery . . . 24
3.1 Filozofia ROSu . . . 26
3.2 Štruktúra ROS1 . . . 27
3.3 Komunikácia ROS2 . . . 28
3.4 Prostredie rviz . . . 30
3.5 Simulačné prostrenie Gazebo . . . 31
3.6 ROS2 Humble poster . . . 32
4.1 Počitač Rasberry Pi 4 . . . 33
4.2 Počitač NVIDIA Jetson Nano . . . 34
4.3 Doska počitaču Intel NUC . . . 35
4.4 Počitač Intel NUC . . . 36
4.5 Intel Realsense D415 . . . 37
4.6 Intel Realsense D455 . . . 37
4.7 Intel Realsense T265 . . . 39
4.8 Intel Realsense L515 . . . 39
5.1 Prostredie Intel RealsenseViewer . . . 42
6.1 Štruktúra uzlu RealSense kamery . . . 44
7.1 Príklad získanej trajektórie pomocou algoritmu RTAB-Map . . . 45
7.2 Príklad detekcie algoritmom FAST . . . 46
7.3 Príklad spájania príznakov desktriptormi BRIEF . . . 47
7.4 Príklad použitia algoritmu SURF . . . 48
7.5 Diagram priebehu detekcie loop closure . . . 49
7.6 Príklad detekcie loop closure . . . 49
7.7 Diagram správy pamäte . . . 50
7.8 Prepojenie RTAB-Map v ROSe . . . 51
8.1 Letové jednotky Pixhawk 4 a Cubepilot Orange . . . 52
8.2 Súradnicový systém NED . . . 54

8.3 Zdielané témy z riadiaceho uzlu 60
8.4 Bezpilotný prostriedok v prostredí Gazebo 60
8.5 Bezpilotný prostriedok X500 61
8.6 Komunikácia v simulačnom prostredí PX4 SITL 62
8.7 Prevod tém medzi Gazebom a ROSom 62
9.1 Prepojenie jednotlivých komponentov 63
9.2 Prepojenie kamery a RTAB-Map v ROSe 64
9.3 Riadiaci uzol uav_control 65
10.1 Súradnicový systém ECEF 66
10.2 Prepojenie tém k ROS node na prepočet súradníc 68
10.3 Príklad vizualizácie trajektórie v prostredí rviz 68
10.4 Prepojenie tém k ROS node pre výpočet odchýlky 69
11.1 Pripravený drone na experiment 70
11.2 Pripravený dron na experiment 71
11.3 Výsledná trajektória RTAB-Map 71
11.4 Vizualizované trajektórie v prostredí rviz 72
11.5 Graf porovnania odchýlky presnosti v jednotlivých osiach 73
11.6 Graf porovnania absolútnej odchýlky presnosti v jednotlivých osiach . 73
11.7 Graf porovnania euklidovskej vzdialenosti 74
11.8 Vizualizované trajektórie v prostredí rviz 75
11.9 Graf porovnania odchýlky presnosti v jednotlivých osiach 76
11.10Vizualizované trajektórie v prostredí rviz 76
11.11Graf porovnania odchýlky presnosti v jednotlivých osiach 77
E.1 Histogram chyby na ose X 98
E.2 Histogram chyby na ose Y 99
E.3 Histogram chyby na ose Z 99
F.1 Graf porovnania absolútnej odchýlky 100
F.2 Graf porovnania euklidovskej vzdialenosti 100
F.3 Graf porovnania absolútnej odchýlky 101
F.4 Graf porovnania euklidovskej vzdialenosti 101

Zoznam tabuliek

3.1 Distribúcie ROSu 32
4.1 Parametre zvoleného Intel NUC [19] 36
4.2 Parametre modelov hľbových kamier rady D400 [18] 38
4.3 Parametre IMU jednotky BMI085 [20] 40
8.1 Letové režimy pre PX4 autopilot[33] 53
8.2 Implementované services 57
E.1 Tabuľka k nameraným dátam v experimente 94

Úvod

Táto diplomová práca sa zaoberá návrhom systému pre estimáciu presnej polohy bezpilotných lietadiel bez použitia GNSS. K estimácií polohy bezpilotného lietadla v priestore sú použité dáta z obrazových a inerciálnych senzorov. Hlavnou motiváciou pri tvorbe tejto práce je možné využitie vytvoreného systému pri rôznych aplikáciách automatizácie dronov. Ako je známe väčšina súčasných systémov pre určovanie polohy je závislá na príjme signálu zo satelitov. Avšak tieto systémy ako napríklad celosvetovo najznámejší používaný spôsob lokalizácie objektov v priestore GNSS, nie je v praxi možné použiť vo vnútorných priestoroch z dôvodu blokovania signálu stenami. Prijímače nie sú schopné zachytiť signál zo satelitov, pretože vysielaný signál zo satelitov nemôže prejsť cez stropy a steny budov. Taktiež vo vojenských priestoroch môžu byť signály GNSS zámerne rušené, prípadne môžu byť vysielané signály, ktoré by mohli viesť k zámernému skresleniu lokalizácie a navigácie daného objektu.

V prvej časti práce sú popísané rôzne spôsoby určovania polohy pomocou vizuálnej a inerciálnej odometrie. Taktiež sú diskutované výhody a nevýhody jednotlivých metód a vhodnosť použitia pre určovanie polohy v priestore.

V ďalšej časti práce je popísaný výber vhodného kamerového senzoru, ktorý bude možné použiť pre aplikáciu vizuálnej odometrie. Taktiež je popísaný výber vhodného palubného počítača. Zároveň je popísaný výber vhodného operačného systému a distribúcie ROS-u. V tejto časti práce je taktiež popísaný spôsob spracovania obrazových a inerciálnych dát.

Ďalej je v práci popísaný výber vhodného algoritmu pre určovanie polohy v priestore na základe vizuálnych a inerciálnych dát. V rámci tejto časti práce sú popísané algoritmy pre detekciu príznakov v obraze a optimalizáciu určenej trajektórie. Taktiež je popísané prepojenie jednotlivých častí v rámci systému ROS a porovnanie s pozičným systémom GPS. V rámci systému ROS bol navrhnutý riadiaci uzol, pomocou ktorého je možné riadiť polohu bezpilotného prostriedku v priestore pomocou implementovaných obslužných servisov. Tento riadiaci uzol bol následne otestovaný v simulačnom prostredí Gazebo.

Výsledkom tejto práce by mal byť systém, pomocou ktorého budeme môcť estimovavať polohu bezpilotného prostriedku v danom známom prostredí bez využitia GNSS. Jedným z hlavných predpokladov lokalizácie bude dosiahnutie čo najvyššej presnosti a spoľahlivosti určenia polohy v priestore.

1 Problematika vizuálnej a inerciálnej odometrie

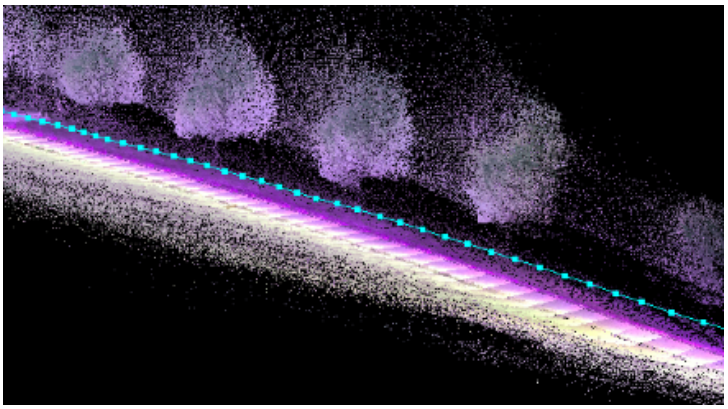
V tejto kapitole sa postupne oboznámime s problematikou vizuálnej a inerciálnej odometrie. Budú popísané možnosti spracovania obrazu a estimovania polohy na základe obrazových a inerciálnych dát.

1.1 Vizuálna odometria

Vizuálna odometria (VO) je proces estimovania polohy a pohybu kamery na základe sekvencie obrázkov. Momentálne sa stáva kľúčovou súčasťou navigačných systémov pre rôzne aplikácie ako napr. autonómne autá, drony, mobilné roboty. Hlavným cieľom použitia vizuálnej odometrie je možnosť sledovania polohy v rôznych typoch terénu ako napr. les, prípadne vnútorné priestory.

Celkovo je vizuálna odometria technológia, ktorá umožňuje autonómnym systémom navigovať a fungovať pri zložitých aplikáciach, kde nie je možné použiť GPS (Global Position System), prípadne je zarušené. Na rozdiel od GPS je výstupom VO relatívna poloha v priestore vzhľadom k úvodnej póze kamery. V praxi sa často spája zároveň s metódou SLAM (Simultaneous localization and mapping), kedy zároveň dochádza aj k mapovaniu prostredia.

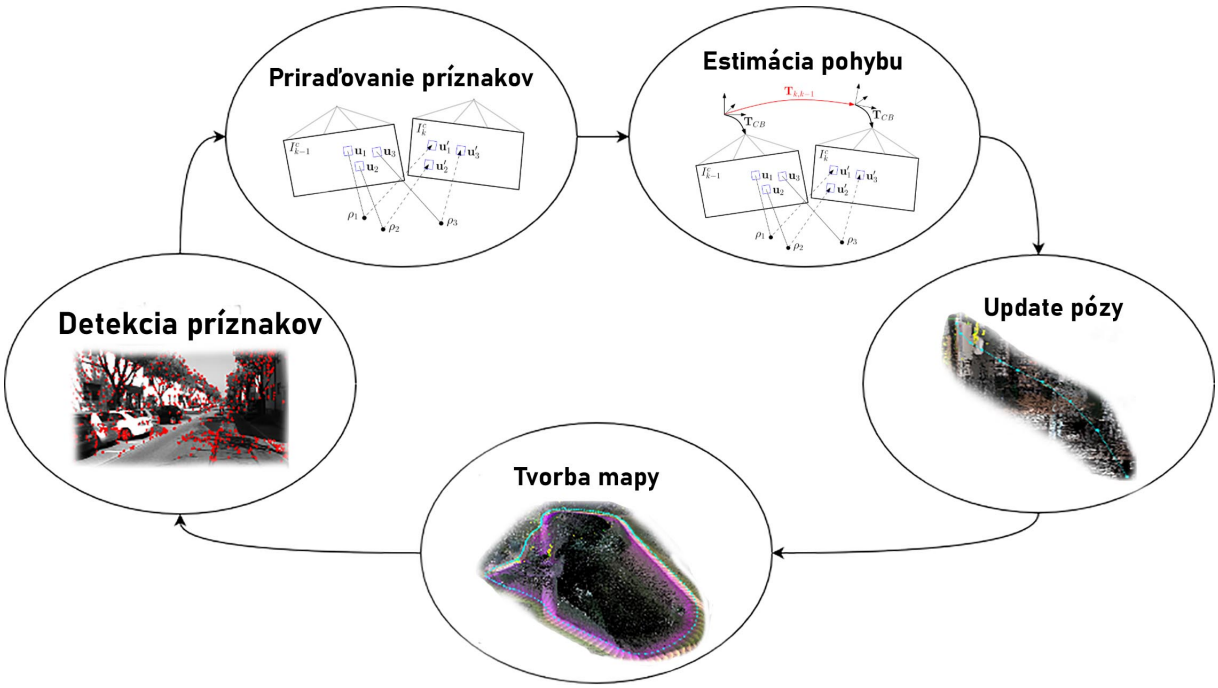
VO je technika, ktorá porovnáva aktuálne snímaný obraz s predošlým a hľadá rozdiely zmeny polohy nájdených príznakov prípadne referenčných bodov pomocou metódy optical flow. Nová póza je získaná pridaním odhadovaného vektora polohy oproti predošlej póze. Zisťovanie zmeny polohy sa rozdeľuje na dva hlavné prístupy: relatívna vizuálna lokalizácia (RVL) a absolútna vizuálna lokalizácia (AVL). Na obrázku 1.1 je znázorený príklad získanej trajektórie pomocou vizuálnej odometrie.[1][2]



Obr. 1.1: Príklad vizuálnej odometrie

1.1.1 Relatívna vizuálna lokalizácia

Pri tejto metóde sa k estimácií zmeny polohy využíva metóda optical flow. K získavaniu kľúčových bodov v obraze sa používajú rôzne algoritmy pre detekcie hrán, rohov, zmeny jasu pixelov v obraze a iné vizuálne výrazné body. Medzi dvoma snímkami prebehne priradenie príznakov, pomocou ktorého sa označia kľúčové body v oboch snímkoch. Zmena polohy je následne vypočítaná na základe zmeny polohy kľúčových bodov v obraze. Po výpočte zmeny polohy sa aktualizuje aktuálna póza kamery. Následne sa v priebehu času poloha akumuluje a ukladá ako trajektória. Na obrázku 1.2 je znázornený diagram priebehu RVL(Relative visual localization).



Obr. 1.2: Diagram priebehu RVL

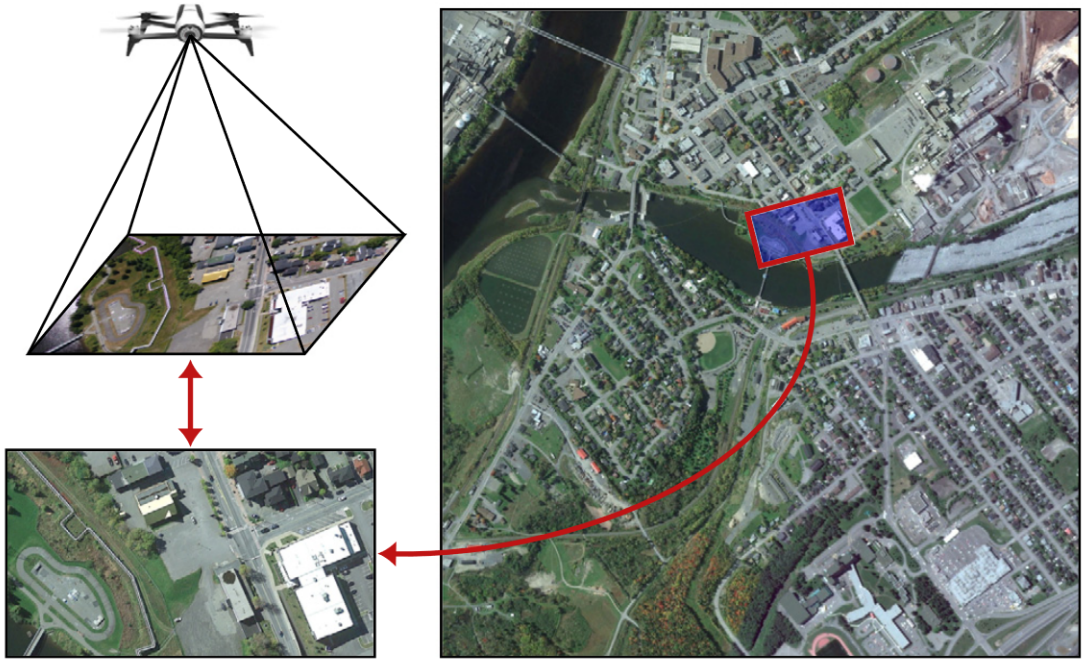
Hlavným problémom samotnej metódy je akumulovanie chyby v čase (drift over time). Odchýlka je prakticky produktom rekurzívneho používania odhadu zmeny polohy pre výpočet novej zmeny polohy. Problém je v tom, že chyba v predchádzajúcom výpočte ovplyvní presnosť aktuálneho. Avšak daný problém je možné vyriešiť rôznymi spôsobmi ako napr. Loop Closure Detection, Graph-Based SLAM, fúziou s IMU (Inertial measurement unit), prípadne reinicializáciou.

Výhodou tejto metódy je hlavne, že k estimácií polohy nepotrebuje mať k svojej funkcii známe prostredie, prípadne snímky (mapu) prostredia. Taktiež je vhodná na použitie vo vnútorných priestoroch. Hlavnou výhodou je jej nezávislosť na systéme GNSS a taktiež jej možnosť použitia v rôznych prostrediach.[1][2]

1.1.2 Absolútna vizuálna lokalizácia

Prístup tejto metódy je zásadne rozdielny oproti RVL. Hlavný rozdiel je v detekcií a priraďovaní príznakov. Metóda využíva vopred zhromaždené referenčné údaje. Predpokladom je, že tieto dáta musia byť presne georeferencované pred ich možným použitím na lokalizáciu. Dáta môžu pozostávať z leteckých snímkov, ktoré sú nasnímané vopred samotným UAV a georeferencované pomocou GNSS na drone. Po spojení vytvoria jednu veľkú ortofotografiu ako je napr. na obrázku 1.3. Prípadne je možné použiť satelitné snímky napr. z Google Earth.

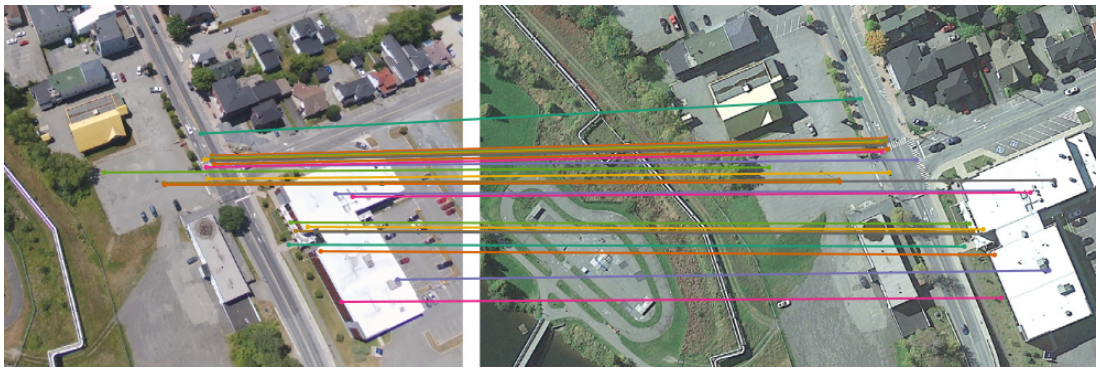
Výhodou tejto metódy oproti RVL je, že nemá problém s akumulovaním chyby v čase, nakoľko pracuje s georeferencovanými dátami. Avšak je funkčnosť a presnosť je závislá na pripravenom datasete s ktorým pracuje. Pri použití voľne dostupných dát je najväčší problém v tom, že každý snímok je pri iných svetelných podmienkach. Problémom môže byť taktiež aj samotná zmena scény ako napr. pridanie/odobratie nejakého objektu.



Obr. 1.3: Príklad AVL s UAV[2]

Ďalším hlavným rozdielom oproti RVL je, že AVL nepracuje s po sebe idúcimi snímkami nakoľko počas lokalizácie porovnáva snímaný obraz z databázou a hľadá zhodu v obraze. Jednou z možností je hľadanie zhody takz. template matching, kedy sa priamo porovnáva snímaný obraz so známou databázou a vyhodnocuje sa podobnosť obrazu. Ďalšou možnosťou je použiť tak ako aj u RVL feature matching.

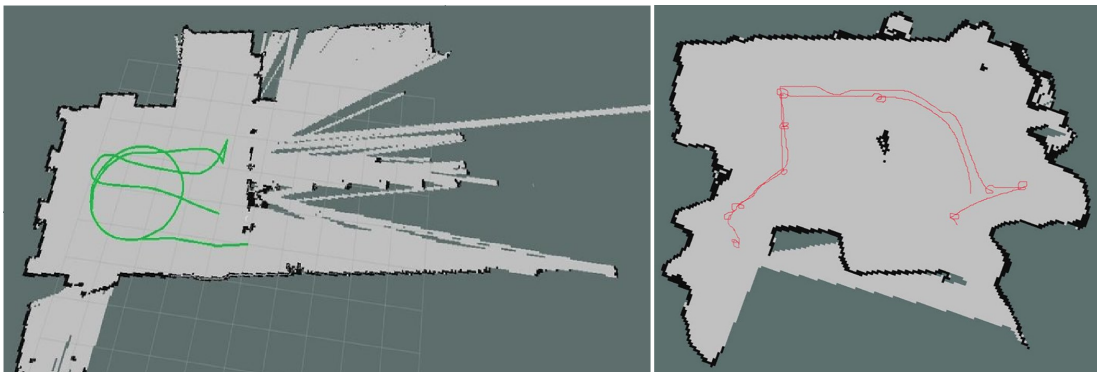
S tým rozdielom, že hľadanie význačných bodov v snímanom obraze sa porovnáva s bodmi na datasete. Príklad je znázornený na obrázku 1.4.



Obr. 1.4: Príklad feature points matching AVL[2]

1.2 SLAM

SLAM (Simultaneous localization and mapping) je technika používaná v robotike a počítačovom videní pre mapovanie a určovanie polohy zariadenia v neznámom prostredí. Tento algoritmus je možné použiť s rôznymi senzormi ako napr. ultrasonickými, laserovými skenermi, RGB a hĺbkovými kamerami. Presnosť a rýchlosť spracovania meraných údajov je kľúčovou pre správnu funkčnosť. V praxi sa taktiež používa na získavanie 2D a 3D mapy neznámeho priestoru. Lokalizácia a mapovanie sú navzájom veľmi závislé. Presná mapa je dôležitá pre lokalizáciu v priestore avšak taktiež správna lokalizácia je zásadná pre mapovanie. Klasické SLAM algoritmy estimujú polohu a súčasne tvoria mapu. Avšak modernejšie technológie spracovania túto úlohu rozdeľujú na dve paralelné úlohy. Na obrázku 1.5 je znázornená vytvo-

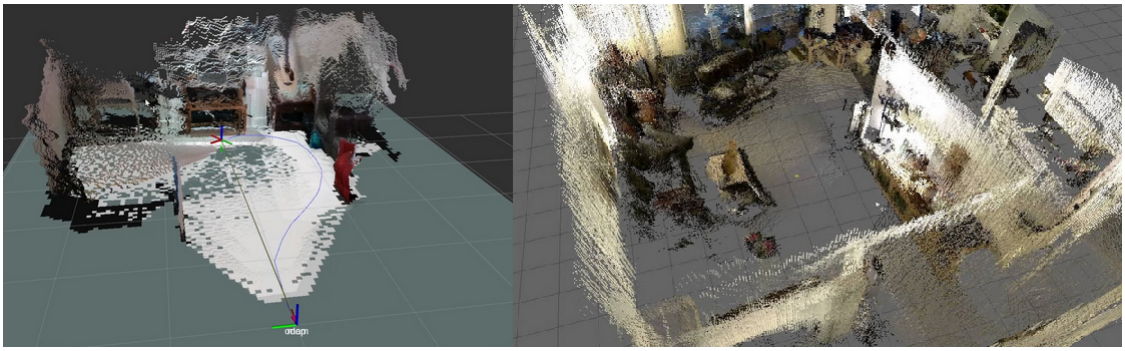


Obr. 1.5: Príklad 2D mapy pomocou SLAM

rená 2D mapa pomocou SLAM. Sivá oblasť predstavuje dostupnú oblasť po ktorej je možné sa pohybovať, čierna oblasť predstavuje prekážky, prípadne steny.

Pri jednoduchších aplikáciách si na tvorbu 2D mapy s nižším rozlíšením, menšou vzdialenosťou a pomalším spracovaním bude postačovať ultrasonický senzor. Avšak na tvorbu komplexnejších 3D máp je vhodnejšie použiť LiDAR (Light Detection and Ranging) prípadne hĺbkovú kameru. Hlavnou výhodou hĺbkovej kamery oproti ultrasonickým senzorom je väčší rozsah meranej vzdialenosti a rýchlejšie meranie a spracovanie väčšieho množstva bodov.

Pri použití hĺbkovej kamery je možné získať viac informácií o meranom bode v porovnaní s LiDARom. V-SLAM (Visual Simultaneous Localization and Mapping), je v literatúre špecifický druh SLAM technológie, ktorý sa zameriava na využívanie informácií získaných z obrazových dát na lokalizáciu a mapovanie v priestore. Na obrázku 1.6 je znázornený príklad použitia pre tvorbu 3D mapy.[5]



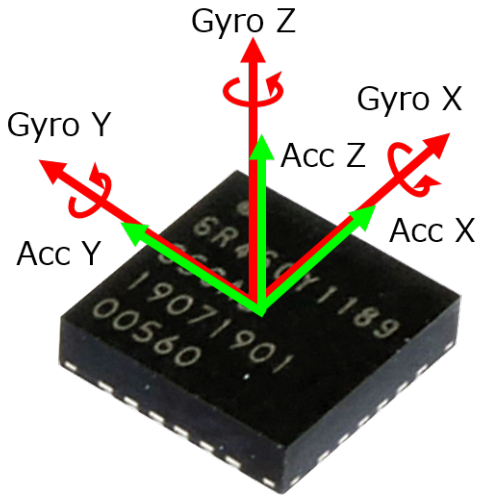
Obr. 1.6: Príklad 3D mapy pomocou V-SLAM

1.3 Inerciálna odometria

Taktiež známa pod názvom INS (Inertial Navigation System), je spôsob určovania polohy, ktorý využíva pohybové senzory ako akcelerometre, rotačné senzory (gyroskopy) pre výpočet aktuálnej polohy, orientácie a rýchlosti pohybujúceho sa objektu bez potreby externej referencie. K výpočtu aktuálnej polohy sa používa tzv. dead reckoning. Je to proces pre výpočet aktuálnej polohy pohybujúceho sa objektu na základe predošlej určenej polohy, známej estimovanej rýchlosti, smeru, a uplynutého času.

Inerciálne jednotky pozostávajú z troch komponentov - akcelerometru, gyroskopu prípadne magnetometra. Pomocou akcelerometru je meraná zmena rýchlosti objektu vo všetkých osiach. Gyroskop poskytuje informáciu o uhlovej rýchlosti, čiže rýchlosti otáčania okolo každej osi. Magnetometer často tvorí súčasť inerciálnej jednotky, kde

slúži na detekciu intenzity magnetického poľa. Týmto spôsobom poskytuje absolútnu orientáciu vzhľadom k magnetickému severu Zeme. Pomocou týchto komponentov sme schopní získať komplexné informácie o pohybe daného objektu, ktoré je možné následne spracovávať. Na obrázku 1.7 je znázornená inerciálna jednotka so šiestimi stupňami voľnosti.



Obr. 1.7: Znázornenie osí inerciálnej jednotky

Všetky inerciálne navigačné systémy majú problém s integračnou chybou. Problémom je, že aj najmenšia chyba v meraní akcelerácie alebo rýchlosti sa postupne v čase bude zväčšovať. Aj najlepšie akcelerometry na trhu v priemere naakumulujú odchýlku okolo 50 metrov za 17 minút. Táto odchýlka sa tiež prejaví v tom, že aj keď zariadením nepohybujeme, systém bude po určitom čase deklarovať zmenu oproti úvodnej póze.

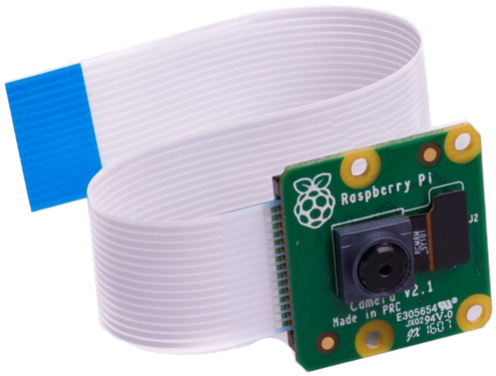
Inerciálna odometria je primárne určená na meranie malých vzdialeností. Pre meranie väčších vzdialeností je nutné túto technológiu kombinovať s inými spôsobmi určenia polohy ako GPS, LiDAR alebo iným zdroj určovania polohy. Pri meraní väčších vzdialeností je potrebné eliminovať integračnú chybu tejto metódy obnovením polohy z iného zdroja. Touto kombináciou je možné dosiahnuť systému, ktorý bude schopný určovať polohu s vyššou presnosťou. [3][4]

2 Senzory pre získavanie obrazových dát

V tejto podkapitole sú popísané typy kamerových senzorov, ktoré je možné použiť pre snímanie obrazu. Zároveň je popísaný možný spôsob merania vzdialenosti medzi danou kamerou a scénou, možnosť použitia v rôznych typoch terénu. Taktiež je diskutovaná vhodnosť použitia daného typu kamery pre vizuálnu odometriu.

2.1 Monokulárne kamery

Monokulárne kamery sú prvou voľbou pre rôzne aplikácie v oblasti automatizácie, robotiky ako napr. snímanie obrazu, detekciov pohybu v obraze, či prípadne fotogrammetriou. Avšak pre odhadovanie vzdialenosti neznámeho objektu v obraze ju prakticky nie je možné použiť. Vzdialenosť objektu v obraze je možné odhadovať iba v prípade, že máme známe rozmery objektu. Jedna z hlavných výhod monokulárnych kamier je ich jednoduchosť a flexibilita. Hlavnou výhodou sú taktiež kompaktné rozmery a ich cenová dostupnosť a možnosť použitia na snímanie rôznych typov terénu. Na obrázku 2.1 je príklad často používanej a cenovo dostupnej monokulárnej kamery.[6]

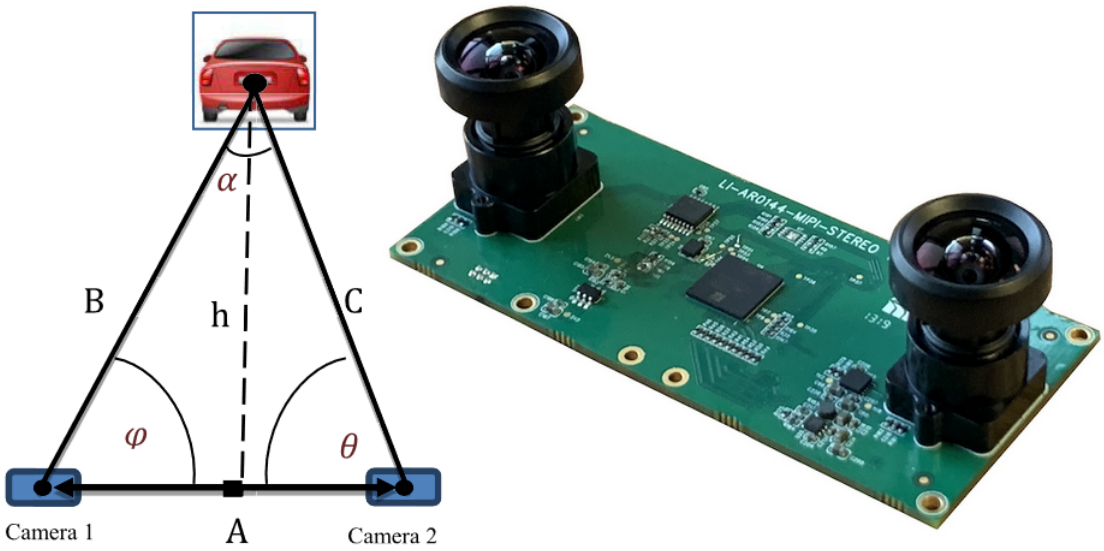


Obr. 2.1: Raspberry Pi kamera

Nakoľko s monokulárnou kamerou nie je možné merať vzdialenosť príznakov v obraze, algoritmy vizuálnej odometrie pracujúcimi len so zmenou pozície kľúčových bodov v obraze dosahuje neuspokojivé výsledky. Avšak v súčasnej dobe sa vyvíjajú algoritmy s využitím hlbokého učenia, ktoré sa naučia detekovať zmeny polohy na predchádzajúcom datasete. Výhodou by mala byť hlavne nezávislosť na presnosti detekcie príznakov a párovaní obrázkov. Avšak pre real-time aplikácie to je náročne použiteľná metóda.[8]

2.2 Stereo kamery

Stereo kamery sú v podstate rozšírením monokulárnych kamier. Prakticky stereo kamera pozostáva z dvoch horizontálne zarovnaných a vertikálne posunutých monokulárnych kamier. Celkový výstupný obraz vytvára dojem hĺbky v scéne. Hlavným rozdielom oproti monokulárnym kamerám je, že pomocou stereo kamery je možné vypočítať vzdialenosť objektu v obraze bez znalosti rozmerov daného objektu. Tento princíp je založený na paralaxii, čiže zmenou pozorovacieho uhla na snímaný objekt. Príklad je znázornený na obrázku 2.2. Objekt je nasnímaný z dvoch rôznych uhľov ϕ a θ, a na základe zmeny polohy objektu medzi snímkami je pomocou trigonometrických funkcií vypočítaná vzdialenosť objektu nakoľko je známa vzdialenosť **A** medzi kamerami. Výpočet je bližšie popísaný v citovanom článku 7. Vo všeobecnosti je možné merať vzdialenosť približne do 10m. Veľkosť maximálnej meranej vzdialenosti a presnosti je závislá od vzdialenosti medzi kamerami a kalibrácií. [7]



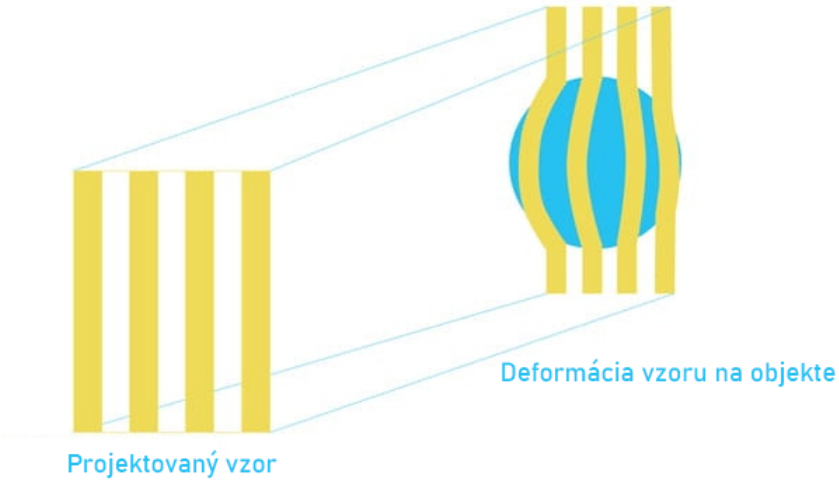
Obr. 2.2: Príklad merania vzdialenosti stereo kamerou[7]

Pri použití s vizuálnou odometriou sa ako prvé nájdu príznaky v obraze z ľavej kamery. Následne sa získajú príznaky z obrazu pravej kamery a dôjde k výpočtu vzdialeností príznakov pre daný obraz. Následne sa tento proces opakuje s novým obrazom. V ďalšom kroku dôjde k spájaniu príznakov medzi jednotlivými obrazmi a estimovaniu zmeny polohy kamery v priestore. Presnosť systému značne závisí od kalibrácie objektívov a svetelných podmienok, nakoľko rozdiely v snímkoch môžu spôsobiť nežiadúce chyby. [9]

2.3 Hĺbkové kamery

Kamery s detekciou hĺbky používajú rôzne technológie pre určovanie vzdialenosti medzi kamerou a objektom v scéne. Pre výpočet sa používa napr. ToF (Time of Flight) alebo technológia štruktúrovaného svetla. Hlavnou výhodou je, že výstupom hĺbkovej kamery je obraz na ktorom má každý pixel priradenú farbu, ktorá zodpovedá vzdialenosti bodu v obraze. Na obrázku 2.4 je príklad nasnímanej scény kde môžeme pozorovať farebnú škálu závislú na meranej vzdialenosti.

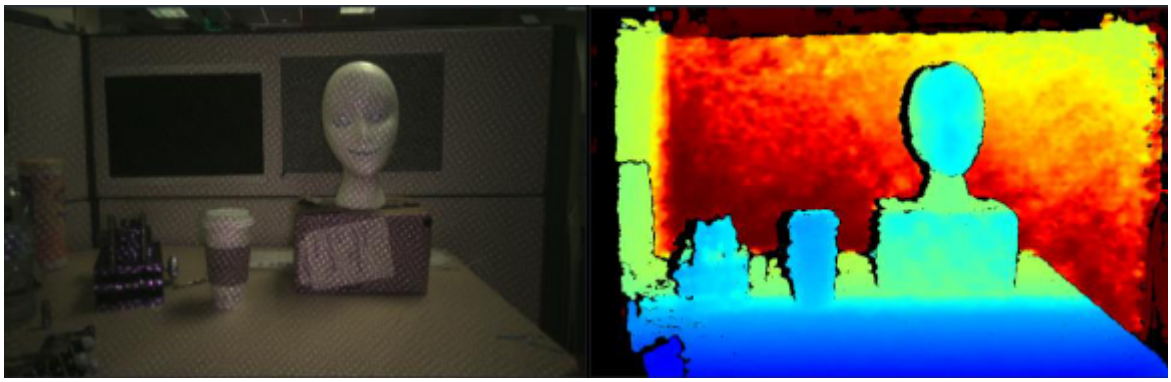
Jednou z možností ako merať hĺbku v obraze, je pomocou štruktúrovaného svetla. Táto metóda je závislá na zdroji nejakého svetla na scénu. Premietaný zdroj svetla je vzorkovaný, vizuálne alebo v priebehu času prípadne kombináciou. Pretože premietaný vzor je známy, zmena vzoru v obraze bude reprezentovať hĺbku. Príklad deformácie vzoru napr. na lopte je znázornený na obrázku 2.3. Nakoľko je táto tech-



Obr. 2.3: Príklad skreslenia pri projekcií svetla[10]

nológia závislá na projektovaní svetla na scénu je primárne určená na použitie vo vnútorných priestorov. Rozsah meranej vzdialenosti je závislí na výkone emitujúceho svetla. Taktiež nie je možné v jednej scéne použiť väčší počet kamier s touto technológiou, nakoľko by sa vzájomne interferovali.

Každý druh hĺbkovej kamery sa pri získavaní hĺbky obrazu spolieha na nejaký známy parameter. Ako napr. pri stereo kamerách je známa vzdialenosť medzi snímačmi. Pri použití štrukturovaného svetla je známy vzor svetla. Hĺbkové kamery založené na meraní hĺbky obrazu pomocou ToF, je známa rýchlosť šírenia svetla. Tento typ kamery tak ako aj lidar využíva laserové svetlo. Následne je pomocou známej rýchlosti šírenia svetla a doby letu vypočítaná vzdialenosť. Maximálna meraná vzdialenosť je závislá od výkonu a vlnovej dĺžky použitého svetla. Hlavnou



Obr. 2.4: Príklad výstupného obrazu hĺbkovej kamery

nevýhodou tohto typu kamery je možná interferencia inou kamerou.

Stereo hĺbkové kamery používajú dva senzory umiestnené v malej vzdialenosti od seba. Tak ako aj pri stereo kamerách, taktiež dochádza k porovnávaniu snímaného obrazu medzi oboma snímačmi. Nakoľko je vzdialenosť medzi snímačmi známa, na základe porovnaní zmien v obraze je možné určiť hĺbku. Pretože hĺbkové stereo kamery môžu využiť akékoľvek vizuálne prvky na meranie hĺbky, fungujú dobre vo väčšine svetelných podmienok, vrátane vonkajšieho prostredia. S pridaním infračerveného projektoru je taktiež možné zachycovať detail hĺbky aj pri slabších svetelných podmienkach. Ďalšou výhodou tohto typu kamery je, že neexistujú žiadne obmedzenia týkajúce sa počtu kamier, ktoré môžete použiť v konkrétnom priestore. Tento typ kamery sa navzájom neinterferuje ako napr. by to robili štruktúrované svetelné kamery alebo kamery na základe času letu svetla.[6][10]

3 Systém ROS

ROS(Robot Operating System) je zbierka open-source middleware balíčkov, ktoré je možné použiť na komplexné ovládanie robotických systémov. Nejde teda o operačný systém, ale o zbierku softvérových frameworkov určených na vývoj robotických systémov. V praxi predstavuje premostenie medzi aktuátormi, senzormi a riadením robota, bezpilotného prostriedku v rámci jedného komplexného systému. Jedným z hlavných prvkov ROSu je jeho schopnosť spravovať komunikáciu medzi jednotlivými časťami systému, ktoré môžu bežať v rámci jedného zariadenia prípadne na väčšom počte zariadení.

ROS je široko využívaný v akademickom prostredí a pri priemyselnom vývoji robotických systémov. Jeho flexibilita a rozšíriteľnosť z neho robia obľúbený nástroj pre výskum, vývoj a prototypovanie v oblasti robotiky.

3.1 História ROSu

Niekedy pred rokom 2007 dvaja študenti doktorandského štúdia Standfordskej univerzity Eric Berger a Keenan Wyrobek pracovali na projekte Personal Robotics Program. Pri práci s robotmi, ktoré vykonávali manipulačné úlohy v ľudskom prostredí, si títo študenti všimli, že ich pri vývoji veľmi brzdí takzvaná *„rozmanitosť robotiky“*. Tým mysleli, že vývojár softvéru nemusí mať potrebnú znalosť hardvéru, alebo niekto, kto pracuje na vývoji najmodernejšej cesty plánovania trasy, nemusí vedieť, ako vytvoriť dokonalé počítačové videnie. V snahe napraviť túto situáciu, sa títo dvaja študenti rozhodli, že vytvoria nejaký základný systém, na ktorom by mohli stavať ostatní v akademickej obci.

Takto započali vývoj základnej verzie na ktorú získali dotáciu 50 000 dolárov. Pri hľadaní financií na ďalší vývoj sa stretli s Scottom Hassanom, zakladateľom spoločnosti Willow Garage, ktorý v tej dobe pracoval na vývoji autonómnych áut. Hassan zdieľal ich myšlienku a pozval ich, aby prišli pracovať do jeho firmy Willow Garage. Následne bola siedmeho novembra 2007 vydaná prvá verzia ROS-u. Od tej doby počet používateľov ROSu zaznamenal veľký nárast s viac ako 16 miliónmi stiahnutí v roku 2018. V súčasnosti sa vyvíja ROS2 kde je komunikačná vrstva založená na službe distribúcie údajov (DDS), ktorá umožňuje rozsiahle distribuované riadenie. architektúry.

Prvá distribúcia ROS-u bola vyvíjaná do roku 2020. Medzi najznámejšie verzie ROS-u patria Kinetic (Ubuntu 16.04), Melodic (Ubuntu 18.04) a Noetic (Ubuntu 20.04). Posledná distribúcia by mala mať podporu do roku 2025. V súčasnosti väčšina systémov prechádza na druhú distribúciu ROS2. Medzi súčasne používané verzie patrí Foxy a Humble. [12][13]

3.2 Filozofia

ROS bol navrhnutý s ohľadom na open-source s cieľom, aby si používatelia mohli vybrať konfiguráciu nástrojov a knižníc, ktoré zaintegrujú do jadra ROS-u tak, aby vyhovovali aplikácii používateľa. Hlavným cieľom ROS-u nie je len prepájať jednotlivé štruktúry. Poskytuje sadu nástrojov na simulácie, širokú škálu knižníc, pomocou ktorých môžu rôzne systémy medzi sebou komunikovať a taktiež komunitu (ekosystém).

Z hľadiska komunikácie je cieľom decentralizovanosť, čiže zabezpečiť peer-to-peer komunikáciu medzi jednotlivými uzlami. Taktiež je možné používať rôzne programovacie jazyky, ako napr. C++ alebo Python, alebo aj iné novšie jazyky. Dôraz sa taktiež kladie na to, aby bol open-source.

Obr. 3.1: Filozofia ROSu [11]

Plumbing znamená prepojenie. ROS poskytuje systém výmeny správ, často nazývaný ako *middleware*. Komunikácia je jednou z prvých vecí, ktoré je potrebné implementovať pri vývoji nejakého robotického systému alebo nejakého softvéru, ktorý bude interagovať s hardvérom. Vstavaný systém ROS-u urýchľuje vývoj a šetrí čas pri spracovávaní komunikácií medzi jednotlivými uzlami prostredníctvom publish/subscribe konceptu. Tento prístup podporuje osvedčené postupy vp vývoji softvéru, vrátane izolácií prípadných chýb a funkčných zariadení.

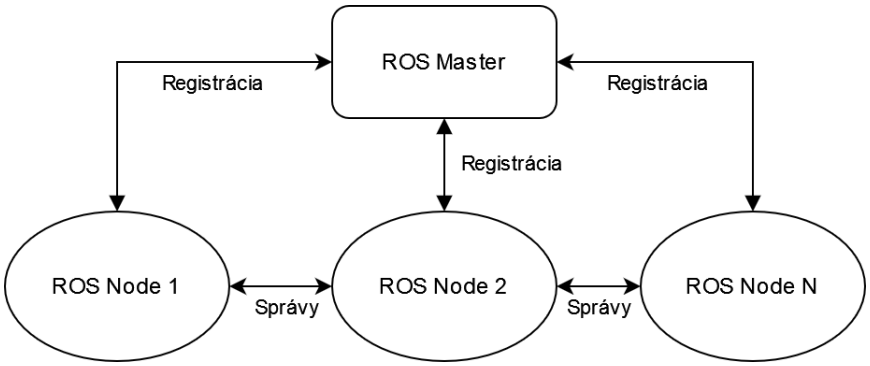
Tools (nástroje) sú dôležitou súčasťou pri vývoji aplikácií. Súčasťou ROS-u sú nástroje, ktoré umožňujú simuláciu, vizualizáciu, zaznamenávanie a prehrávanie. Jedným z nástrojov je napr. Gazebo.

Capabilities (schopnosti). Cieľom projektu ROS je neustále znižovať bariéru, ktorá stojí medzi nápadom a realizáciou aplikácie. Každý s dobrým nápadom by ho mal byť schopný zrealizovať aj bez toho, aby rozumel všetkému o základnom hardvéri a softvéri.

V neposlednom rade ROS taktiež poskytuje komunitu, ktorá pozostáva zo študentov a fanúšikov až po nadnárodné korporácie. Týmto sa neustále podporuje vývoj ROS-u a rôznych dostupných projektov.[11][12]

3.3 Štruktúra komunikácie ROS1

Všetky procesy v štruktúre sú reprezentované ako nodes (uzly), navzájom sú prepojené pomocou topics (tém). Jednotlivé uzly môžu medzi sebou odosielať správy pomocou topics, poskytovať služby iným uzlom, nastaviť alebo získavať zdieľané dáta zo spoločnej databázy, ktorá sa volá parameter server. Hlavným procesom je ROS Master, ktorý uskutočňuje registrovanie jednotlivých uzlov, vytvára komunikáciu medzi jednotlivými uzlami prostredníctvom tém. Správy medzi jednotlivými uzlami neprechádzajú cez master. Štruktúra komunikácie ROS1 je znázornená na obrázku 3.2. Master nastaví peer-to-peer komunikáciu medzi všetkými uzlami, potom čo sa zaregistrovali u mastera. Táto decentralizovaná architektúra je vhodná pre systémy, ktoré pozostávajú z menších podmnožín, ktoré môžu medzi sebou komunikovať.



Obr. 3.2: Štruktúra ROS1

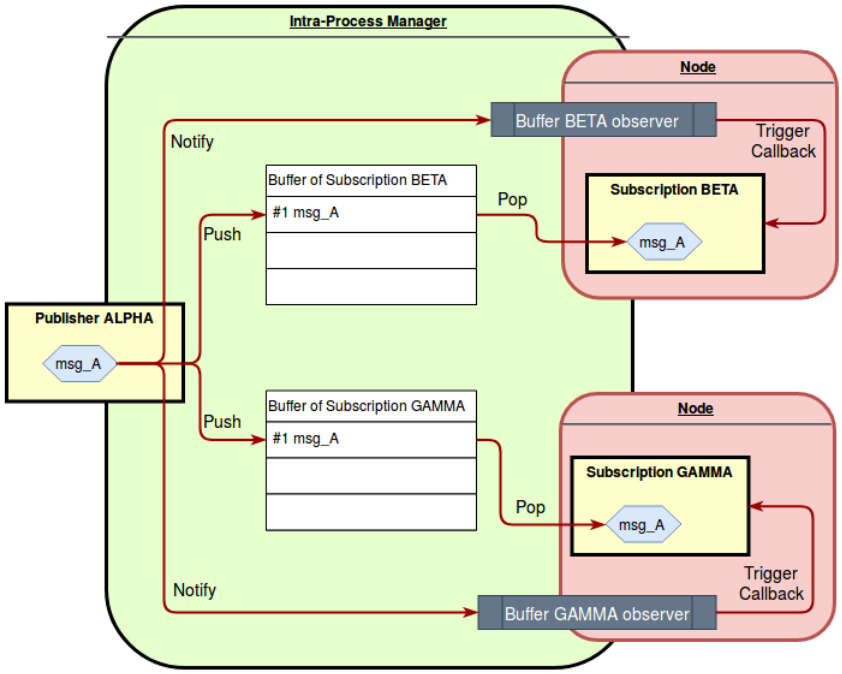
Tento spôsob komunikácie pri ktorom je komunikácia medzi jednotlivými uzlami závislá na ROS masterovi, nie je príliš robustná a ťažšie realizovateľná hlavne pokiaľ sa jedná o väčšie množstvo uzlov na viacerých zariadeniach. Taktiež pri multi-master aplikáciach je potrebné vyriešiť vhodné nastavenie siete a smerovanie na správny ROS master, nakoľko v rámci jednej siete je možné mať iba jednu inštanciu ROS master.[11][12][13]

3.4 Štruktúra komunikácie ROS2

Na základe limitácií komunikácie v ROS1 bol vývojármi v rámci druhej distribúcie ROSu implementovaný štandard Object Management Group's štandard DDS (Data Distribution Service) ako komunikačný *middleware*. Tento systém komunikácie sa považuje za škáľovateľný, robustný a osvedčený v kritických systémoch. Výsledkom je, že DDS umožňuje implementáciu ROSu v priemyselných aplikáciach. DDS je v

ROS-e implementovaný pomocou RMW (ROS Middleware Interface), ktorého súčasťou je taktiež implementácia QoS (Quality of Service). Používateľ má možnosť si vybrať medzi možnosťami politík kvality služieb. QoS v ROS2 umožňuje konfigurovať rôzne parametre komunikácie, ako sú spoľahlivosť, oneskorenie, frekvencia vzoriek a mnoho ďalších. Tieto parametre umožňujú prispôsobiť komunikáciu podľa potrieb konkrétnej aplikácie alebo systému.

Keďže ROS2 používa DDS pre výmenu správ, Discovery modul protokolu RTPS (Real Time Publish-Subsribe protocol) umožňuje aby sa ROS2 uzle našli automaticky navzájom v rámci siete, takže už nie je potrebný ROS2 master pre sprostredkovanie komunikácie, čiže už nie je problém implementovať multi-master aplikácie. ROS2 intra-process manažér umožňuje komunikáciu medzi jednotlivými uzlami pomocou publish-subsribe modelu. Tento model umožňuje efektívne zdieľanie dát medzi jednotlivými uzlami. Na obrázku 3.3 je znázornený spôsob prijímania správy od



Obr. 3.3: Komunikácia ROS2 [14]

ľubovoľného uzlu. Nakoľko je v rámci ROS2 implementovaný QoS, všetky správy sa ukladajú do fronty. Následne sú správy načítavané z fronty, pokiaľ sú dostupné dáta. *Publisher node* pomocou notify príznaku dá signál *subscriber* uzlom o dostupnosti nových dát. Používanie QoS umožňuje konfiguráciu správ podľa požiadaviek aplikácie, ako je napríklad nastavenie maximálneho počtu správ v fronte alebo určenie priority správ na základe dôležitosti.

Jedinou menšou nevýhodou ROS2 je v súčasnej dobe menší počet implementovaných projektov. Avšak je implementovaná experimentálna možnosť prepojenia ROS1 aplikácií s ROS2.[11][12][14]

3.5 Súčasti ROS-u

V nasledujúcich podkapitolách sú popísané jednotlivé súčasti ROS-u, ktoré sú základnými blokmi pri práci s ROS-om. Pomocou týchto súčastí je možné vyvíjať komplexné systémy.

3.5.1 Nodes

Každý uzol reprezentuje jeden proces v štruktúre. Každý uzol musí mať unikátne meno, ktoré sa musí zaregistrovať u ROS Mastera predtým, ako bude môcť vykonávať nejakú činnosť. V prípade, že meno nebolo definované, môže ísť o anonymný uzol. K anonymnému uzlu sa vždy automaticky vytvorí nejaký iný identifikátor. Na rozdiel od ROS1, v ROS2 sa už uzle nemusia registrovať u ROS Mastera. Uzol je schopný si nájsť iný uzol na sieti bez ROS Masteru pomocou DDS.

Každý uzol v systéme je nezávislý na ostatných uzloch a vykonáva nejakú špecifickú úlohu. Napríklad môže byť uzol, ktorý spracováva senzorické dáta pre iný uzol, ktorý bude riadiť systém na základe spracovaných dát. Týmto spôsobom je možné rozdeliť komplexnú úlohu do logistických celkov.[12]

3.5.2 Topics

Uzly využívajú témy na vzájomné posielanie a prijímanie správ. Každé meno témy musí byť unikátne, inak môže dôjsť k nežiadúcej zámene dát. Na poslanie správy musí uzol zverejniť (publish) správu v danej téme. Naopak, pre prijatie správy musí odoberať danú tému. Tento model zverejňovania/odoberania (publish/subsribe) je anonymný. To znamená, že žiadny uzol nevie, ktoré uzly v danej téme vysielajú alebo prijímajú. V danej téme je známe len to, či sa v nej vysiela alebo prijíma správa a koľko odberateľov má.

Formát správy môže byť štandardný, ktorý je voľne dostupný a používa sa vo väčšom množstve projektov. Prípadne je taktiež možné si vytvoriť vlastný typ správy podľa požiadaviek danej aplikácie.[12]

3.5.3 Services

Uzly môžu taktiež poskytovať služby. Niektorý klientský uzol môže požiadať o poskytnutie nejakých dát iný uzol, ktorý ho obslúži. Klientský uzol vyšle správu o

požiadavku a čaká na odpoveď od nejakého uzla, ktorý poskytuje daný servis. V praxi sa používajú na vykonanie nejakej úlohy ako napr. vzlet/pristátie bezpilotného prostriedku.[12]

3.5.4 Parameter server

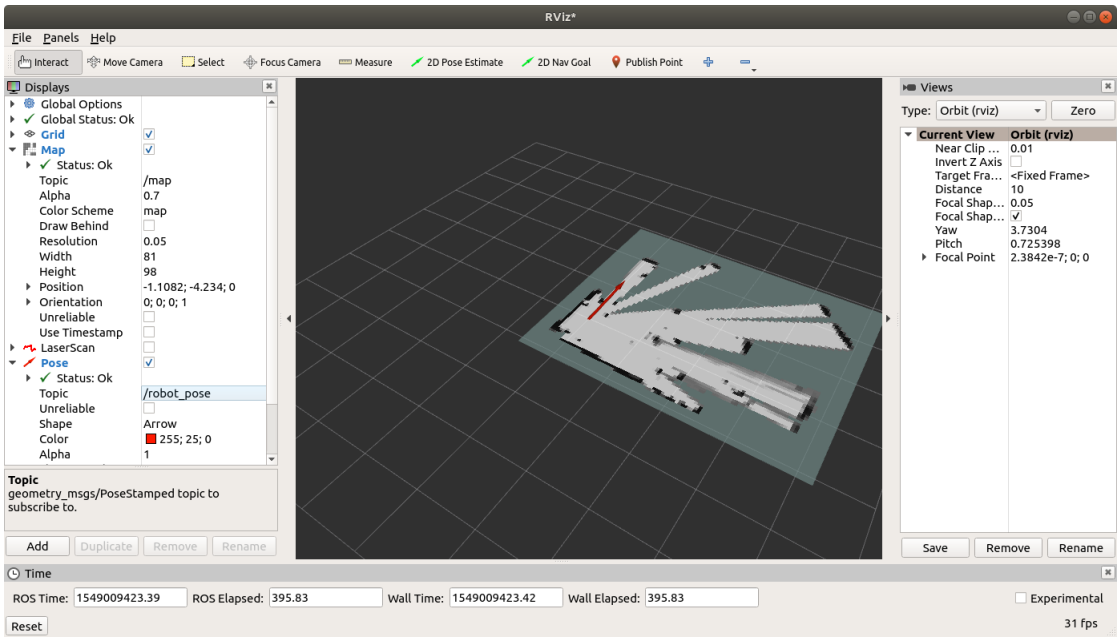
Je databáza, ktorá je zdieľaná medzi všetkými uzlami. Táto databáza umožňuje spoločný prístup k statickým alebo nie často aktualizovaným informáciám. Nie je určený pre náročné dátové operácie. Je určený pre dáta, ktoré sa nebudú často aktualizovať ako napríklad nejaké konfiguračné parametre.[12]

3.5.5 Nástroje

Súčasťou ROS-u sú taktiež nástroje pomocou ktorých je možné efektívne testovať a vyvíjať rôzne aplikácie.

Rviz

Rviz je nástroj na vizualizáciu dát v ROS 2. Pomocou tohto nástroju, je možné zobrazovať rôzne typy dát v 2D a 3D priestore, vizualizovať a overiť funkčnosť svojich algoritmov a senzorov v reálnom čase. Je to dôležitý nástroj pri vývoji, testovaní a ladení aplikácií. Na obrázku 3.4 je znázornené prostredie rviz v ktorom je možné vizualizovať rôzne typy správ.



Obr. 3.4: Prostredie rviz

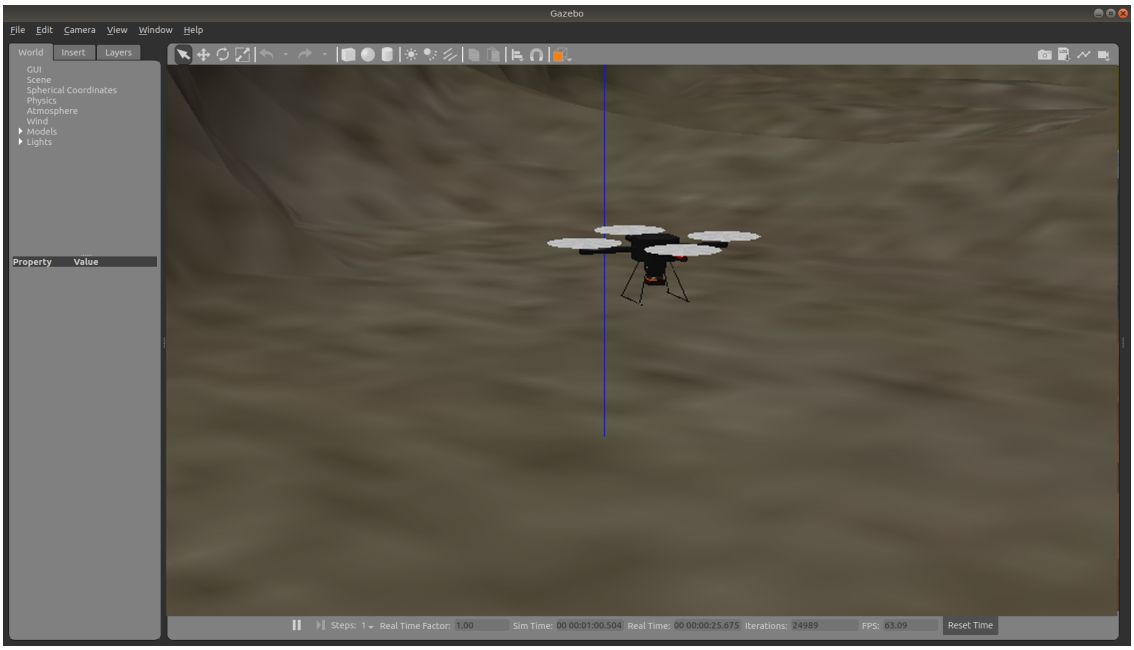
Vďaka schopnosti zobrazovať dáta vo vizuálne forme a poskytovať interakciu s týmito dátami robí rviz neoddeliteľnou súčasťou nástrojov využívaných pri vývoji pokročilých aplikácií v ROS 2.

Rosbag

Rosbag je nástroj pre záznam a prehrávanie dátových vstupov a výstupov. Prakticky sa jedná o špeciálny typ súboru v ktorom sa ukladajú všetky informácie o témach a servisoch v čase. Je užitočný pri analýze dát, ladení, testovaní rôznych aplikácií. Hlavnou výhodou je, že na zaznamenaných dátach je možné pracovať na ladení programu bez toho aby sme museli mať fyzický hardvér.

Gazebo

Gazebo je simulátor robotických systémov, ktorý umožňuje vývojárom testovať a ladiť svoje aplikácie v simulovanom prostredí. Poskytuje podporu pre rôzne robotické platformy a modely, vrátane mobilných robotov, bezpilotných prostriedkov, manipulátorov a iných rôznych zariadení. Na 3.5 je znázornená simulácia bezpilotného prostriedku v reálnom prostredí. Taktiež umožňuje simulovať rôzne podmienky, ako



Obr. 3.5: Simulačné prostrenie Gazebo

sú rôzne terény, poveternostné podmienky alebo svetelné podmienky, čo je užitočné pre testovanie algoritmov v rôznych prostrediach. Tak ako aj pri rvize hlavnou výhodou, je možný vývoj aplikácií bez hardvéru.

3.6 Distribúcie ROS-u

Každá verzia ROS-u predstavuje štruktúrovaný súbor knižníc, nástrojov a balíkov. Rôzne verzie postupom času vylepšovali funkčnosť a opravovali chyby z predošlých distribúcií. Jednotlivé distribúcie ROS-u majú rôzne špecifikácie a podporované platformy a operačné systémy. V tabuľke 3.1 sú popísané dostupné verzie ROS 2. Pre najnovší operačný systém Ubuntu 22.04 sú dostupné momentálne dve verzie Iron a Humble.

Tab. 3.1: Distribúcie ROSu

Distribúcia	Predstavenie	Podpora (roky)	OS
Iron Irwini	23.5.2023	1,5	Ubuntu 22.04
Humble Hawksbill	23.5.2022	5	Ubuntu 22.04
Galactic Geochelone	23.5.2021	1,5	Ubuntu 20.04
Foxy Fitzroy	5.6.2020	3	Ubuntu 20.04
Eloquent Elusor	22.11.2019	1	Ubuntu 18.04
Dashing Diademata	31.12.2019	2	Ubuntu 18.04
Crystal Clemmys	14.12.2018	1	Ubuntu 18.04
Bouncy Bolson	2.7.2018	1	Ubuntu 18.04
Ardent Apalone	8.12.2017	1	Ubuntu 18.04

Avšak verzia Humble má dlhšie časové obdobie podpory. Z tohto dôvodu je pre súčasné aplikácie vhodné zvoliť práve túto distribúciu. Na obrázku 3.6 je znázornený poster distribúcie Humble.

Obr. 3.6: ROS2 Humble poster [11]

4 Výber hardvéru

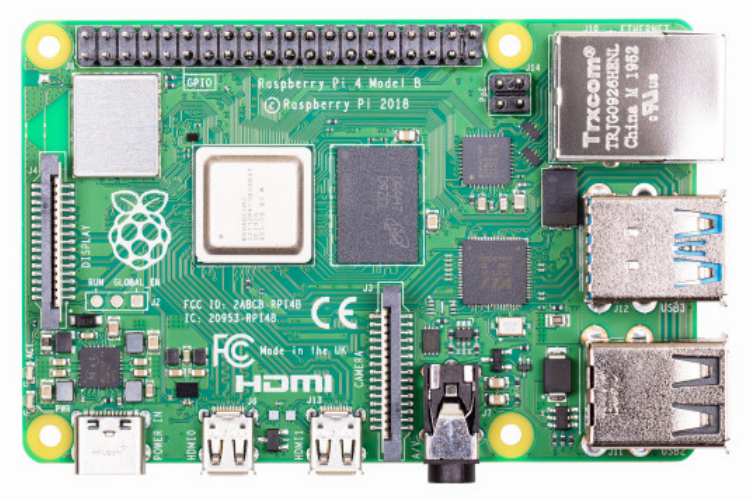
V tejto kapitole je popísaný výber vhodného počítača a kamery za účelom použitia pre vizuálnu odometriu.

4.1 Výber počítaču

V nasledujúcej kapitole sú opísané vhodné platformy PC, ktoré je možné použiť pre spracovanie obrazových a inerciálnych dát na bezpilotnom prostriedku. Pri výbere je potrebné myslieť na vhodné rozmery, hmotnosť a dostatočný výpočetný výkon.

4.1.1 Raspberry Pi 4

Raspberry Pi 4 je jedným s najnovších verzií populárnej série jednodoskových počítačov vyvinutých firmou Raspberry Pi. Tento mini počítač je známy svojou kompaktnou veľkosťou a nízkou cenou. Raspberry Pi 4 je poháňaný štvorjadrovým procesorom ARM Cortex-A72 s frekvenciou 1,5 GHz. Ponúka možnosti s 2 GB, 4 GB alebo až 8 GB LPDDR4 operačnej pamäte. Nedisponuje dedikovaným grafickým procesorom, avšak má integrovanú grafiku VideoCore v procesore. Disponuje rôz-

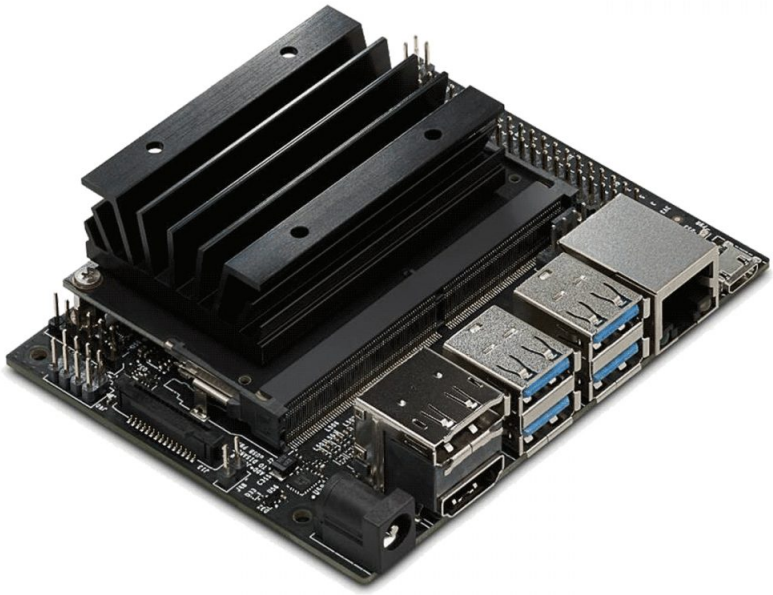


Obr. 4.1: Počítač Rasberry Pi 4

nymi perifériami ako HDMI, Ethernet, USB 3.0, USB 2.0, Wi-Fi, Bluetooth, porty pre pripojenie kamery a displeju. Nedisponuje žiadnou vnútornou pamäťou. Úložný priestor je limitovaný iba na SD kartu. Podporuje rôzne operačné systémy ako Raspbian, Ubuntu a mnoho ďalších distribúcií Linuxu.[15]

4.1.2 NVIDIA Jetson Nano

NVIDIA Jetson Nano je jednodoskový počítač vyvinutý spoločnosťou NVIDIA špeciálne pre aplikácie s umelou inteligenciou a hlbokým učením. Užívateľom poskytuje kompaktnú a cenovo dostupnú platformu. Disponuje štvorjadrovým procesorom ARM Cortex-A57 a grafickým procesorom s 128 NVIDIA CUDA jadrami. Grafický procesor podporuje hardvérovú akceleráciu pre aplikácie strojového učenia. Súčasťou tohto počítača je široká škála zberníc ako USB3.0, HDMI, DisplayPort,



Obr. 4.2: Počitač NVIDIA Jetson Nano

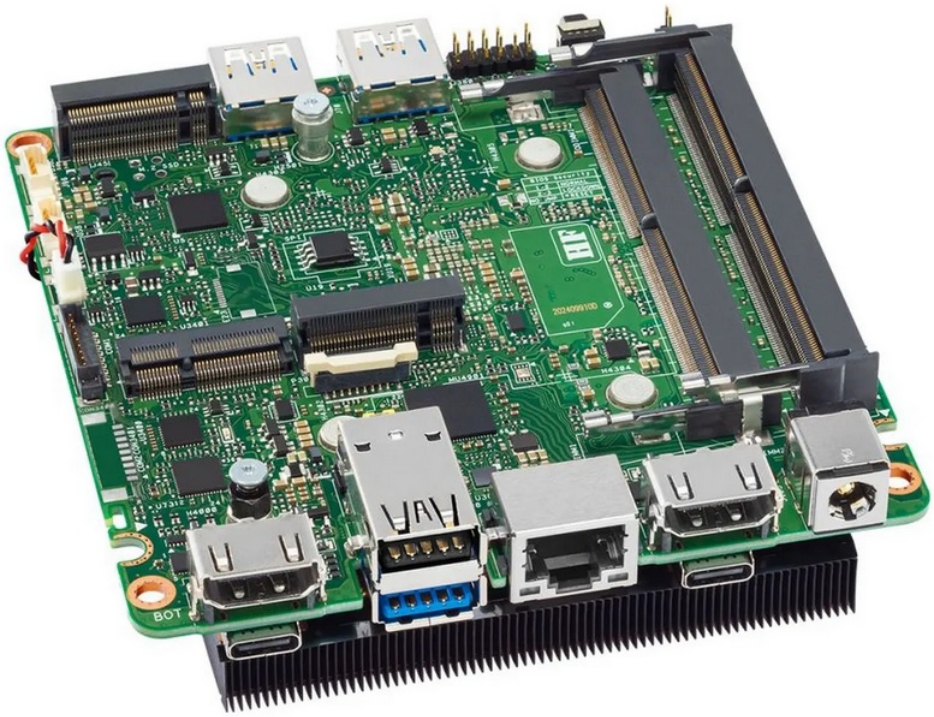
GPIO, I2C, SPI. NVIDIA taktiež poskytuje potrebné knižnice pre vývoj aplikácií. Táto platforma je podporovaná operačným systémom Ubuntu upraveným pre túto architektúru.[16]

4.1.3 Intel NUC

Intel NUC (Next Unit of Computing) je séria mini PC vyvinutých spoločnosťou Intel. V súčasnosti je možné kúpiť pod označením ASUS NUC, nakoľko od roku 2024 séria týchto mini počítačov prešla k firme ASUS. Tieto kompaktné zariadenia ponúkajú vysoký výkon v malom a elegantnom dizajne, čo ich robí ideálnymi pre použitie v rôznych aplikáciách. Intel ponúka široké spektrum konfigurácií komponentov, prípadne je možné si zakúpiť Intel NUC iba s procesorom. Takže je možné

tieto počítače konfigurovať s rôznymi typmi procesorov, pamätí RAM a veľkosťou úložného priestoru, čo umožňuje používateľom prispôsobiť výkon podľa potreby aplikácie.

Jedným z významných benefitov sú malé rozmery a možnosť rozširovania pamäte RAM a úložiska. Výhodou je taktiež pripojenie SSD disku cez M.2 port. Tak ako aj konkurenčné dosky Intel NUC taktiež disponuje širokou škálou portov vrátane USB, HDMI, Ethernet a mnoho ďalších. Výhodou je taktiež možnosť inštalovať štandardný operačný systém Ubuntu.[17]



Obr. 4.3: Doska počitaču Intel NUC

4.1.4 Porovnanie

Na základe zistených informácií a predošlých skúseností s jednotlivými doskami bolo rozhodnuté, že pre vývoj bude použitá doska od firmy Intel. Intel NUC disponuje najvyšším výkonom spomedzi vybraných dosiek. Výhodou je možnosť rozširovania pamäte RAM až do 64GB a pripojenie SSD disku cez M.2 port. Ďalšou výhodou je taktiež kompatibilita operačných systémov. Jedinou možnou nevýhodou oproti doske NVIDIA Jetson a Rasberry PI 4 je vyššia hmotnosť. Ďalšou nevýhodou Intel NUC-u je vyššia cena oproti konkurenčným doskám.

4.1.5 Parametre zvoleného Intel NUC

V tabuľke 4.1 sú parametre zvoleného počítaču Intel NUC. Na obrázku 4.4 je znázornený použitý počítač Intel NUC 11 Pro Kit Slim (NUC11TNKi7) v originálnom ochrannom kryte.

Tab. 4.1: Parametre zvoleného Intel NUC [19]

Model procesora	Intel Core i7 1165G7 Tiger Lake
Frekvencia procesora	1,2 GHz (1 200 MHz)
Počet jadier procesora	4 ×
Cache procesora	12 MB
Core Boost Frekvencia	4,7 GHz (4 700 MHz)
Pamäť RAM	2x16GB (64GB max.)
Počet slotov RAM	2 ×
Model grafickej karty	Iris Xe Graphics
Sloty pre prídavné karty	M.2
Použitý SSD disk	WD Blue SN570 500 GB
Základná výbava	Bluetooth, Wi-Fi
Typ WiFi	802.11a/b/g/n/ac/ax
Grafické výstupy	HDMI, USB-C
Ďalšie výstupy	RJ-45 (LAN) 2.5Gbps

Obr. 4.4: Počitač Intel NUC [19]

4.2 Výber kamery

Na základe rešerše v kapitole 2 bolo rozhodnuté, že najvhodnejším typom kamery pre použitie s vizuálnou odometriou bude stereo hĺbková kamera.

4.2.1 Hĺbkové kamery Intel

Firma Intel ponúka širokú škálu stereo hĺbkových kamier. Jednotlivé modely sa odlišujú použitým senzorom, podporovaným rozlíšením a frekvenciou snímok. Hlavnou výhodou je taktiež podpora pre ROS. Firma Intel podporuje vývoj balíčku pre získavanie obrazových dát z kamery.

Medzi najpoužívanejšie hĺbkové kamery patrí rada D400. Tieto kamery využívajú infračervené projektory a senzory pre snímanie hĺbky v obraze, čo umožňuje 3D mapovanie priestoru. Zároveň tieto kamery disponujú farebným senzorom s vysokým rozlíšením a širokým zorným poľom. Vďaka širokému zornému poľu, je možné v jednom snímku zachytiť veľkú oblasť.

Obr. 4.5: Intel Realsense D415 [10]

Obr. 4.6: Intel Realsense D455 [10]

V tabuľke 4.2 sú popísané jednotlivé parametre modelov kamier z rady Realsense D400. Na obrázku 4.5 a 4.6 sú zvolené modely D415 a D455. Tieto modely sa

odlišujú typom hĺbkového a farebného senzoru, šírkou zorného poľa a použitím IR projektorom. Model D455 má oproti modelu D415 vstavanú IMU jednotku.[10][18]

Tab. 4.2: Parametre modelov hĺbových kamier rady D400 [18]

D400 series Depth Cameras	**Intel® RealSense™ Depth Camera D415**	**Intel® RealSense™ Depth Camera D435/D435i/ D435f/D435if**	**Intel® RealSense™ Depth Camera D455/ D455f/D456**	**Intel® RealSense™ Depth Camera D405**
Hĺbkový modul	Intel® RealSense™ Depth module D415	Intel® RealSense™ Depth module D430	Intel® RealSense™ Depth module D450	Intel® RealSense™ Depth module D401
Typ obrazu	Štandartný	Širokouhlý	Širokouhlý	Širokouhlý
Zorné pole HD (16:9) (°)	H:65 / V:40 / D:72	H:87 / V:58 / D:95	H:87 / V:58 / D:95	H:84 / V:58 / D:92
Zorné pole VGA (4:3) (°)	H:50/ V:40 / D:61	H:75 / V:62 / D:89	H:75 / V:62 / D:89	-
IR Projector	Štandartný	Širokouhlý	Širokouhlý	-
Zorné pole IR Projektoru	H:67 / V:41 / D:75	H:90 / V:63 / D:99	H:90 / V:63 / D:99	-
Farebný senzor	OV2740	OV2740	OV9782	OV9782
Zorné pole farebného senzoru	H:69 /V:42 /D:77	H:69 /V:42 /D:77	H:90 /V:65 /D:84	H:84 /V:58 /D:92
IMU	-	-/6DoF/-/6DoF	6DoF	-
IMU Model	-	Bosch BMI055	Bosch BMI055	-

Poznámka: H- horizontálne, V - vertikálne, D - diagonálne

4.2.2 Mapovacie kamery

Jediným modelom v tejto kategórií je kamera IntelRealse T265. Jej súčasťou je stereo pár kamier a vstavaná IMU jednotka. Oproti ostatným kamerám sa odlišuje vstavaným algoritmom V-SLAM, ktorý beží priamo na kamere. Tento algoritmus určí polohu kamery v priestore a poskytuje taktiež odometrické dáta kamery. Kamera má uzavretú konfiguráciu, ktorú nie je možné meniť. Kamera pracuje s rozlíšením 848x480 s frekvenciou 30 snímkov za sekundu. Hlavnou výhodou sú znížené nároky na výpočtový výkon pri spracovaní obrazu, nakoľko výpočet polohy prebieha priamo v kamere. Na rozdiel od hĺbkových kamier z rodiny D400, výstupom kamery

Obr. 4.7: Intel Realsense T265 [10]

T265 nie je hĺbka obrazu. K výpočtu polohy priestore využíva stereo pár kamier pre detekciu príznakov v obraze.

4.2.3 Lidar kamery

Model kamery Intel Realsense L515 je jedinou kamerou od firmy Intel, ktorá k určeniu hĺbky v obraze využíva Lidar technológiu. Oproti kamerám z rodiny D400

Obr. 4.8: Intel Realsense L515 [10]

má menší rozsah hĺbky v obrazu a nižšiu snímkovaciu frekvenciu. Avšak tento typ kamery disponuje vyšším rozlíšením hĺbkového obrazu.

4.2.4 Vstavaná IMU jednotka

Vybrané modely kamier Realsense majú v sebe integrovanú IMU jednotku. Vo všetkých vybraných modeloch je IMU jednotka od firmy Bosch. BMI085 je veľmi miniatúrna 6 osá inerciálna jednotka, ktorej súčasťou je 3-osí 16-bit akcelerometer a 3-osí 16-bit gyroskop. V tabuľke 4.3 sú popísané parametre IMU jednotky Bosch BMI085.[20]

Tab. 4.3: Parametre IMU jednotky BMI085 [20]

Počet stupňov voľnosti	6
Rozsah akcelerometru	± 4 g
Vzorkovacia frekvencia akc.	100, 200 (Hz
Rozsah gyroskopu	± 1000 deg/s
Vzorkovacia frekvecia gyr.	200, 400(Hz)

Poznámka: Parametre vzhľadom na možnosti získavania dát pomocou librealsense

4.3 Zvolená kamera

Z dostupných a otestovaných modelov kamier D415 a D455, bol zvolený model kamery D455. Hlavnou výhodou oproti modelu D415 je podstatne väčšie zorné pole a vstavaná IMU jednotka. Jedinou menšou nevýhodou je mierne vyššia hmotnosť a rozmery.

5 Inštalácia

V rámci tejto kapitole je popísaná inštalácia operačného systému, ROS-u a ovládačov potrebných pre komunikáciu s hĺbkovou kamerou RealSense. Taktiež je popísaná možnosť kalibrácie kamery. Všetky potrebné príkazy pre inštaláciu sú súčasťou prílohy B

5.1 Operačný systém

Vzhľadom na zvolenú distribúciu systému ROS2 Humble bolo rozhodnuté, že bude potrebné kvôli kompatibilite zvoliť operačný systém Ubuntu 22.04. Pre počítač Intel NUC je možné inštalovať systém napr. pomocou bootovateľného USB. Link pre stiahnutie obrazu a softvéru pre tvorbu bootovateľného USB je v prílohe B.1.

5.2 ROS Humble

Celý postup inštalácie ROS Humble je popísaná v prílohe B.2. Prvým krokom je aktualizácia operačného systému a inštalácia potrebných nástrojov. Ďalej je potrebné pridanie repozitáru ROS2 do systému. Následne je možné inštalovať ROS pomocou príkazu **sudo apt install -y ros-humble-desktop**.

5.3 Ovládače kamery

Pre prácu s kamery Realsense je potrebná inštalácia Intel Realsense SDK 2.0. Inštalácia je možná pomocou predkompilovaných binárných súborov. Príkaz pre inštaláciu je v prílohe B.3. Po inštalácií bolo pri testovaní zistené, že nebolo možné čítať dáta z IMU jednotky. Pomocou postupu popísaného v prílohe B.3.1 je možné doinštalovať potrebné ovládače pre komunikáciu s IMU. Po doinštalovaní **librealsense2-dkms** a **librealsense2-utils** bolo zaznamenané taktiež zlepšenie v načítavaní obrazu z kamery, nakoľko predtým dochádzalo k strate niektorých framov.

5.4 RealSense Wrapper

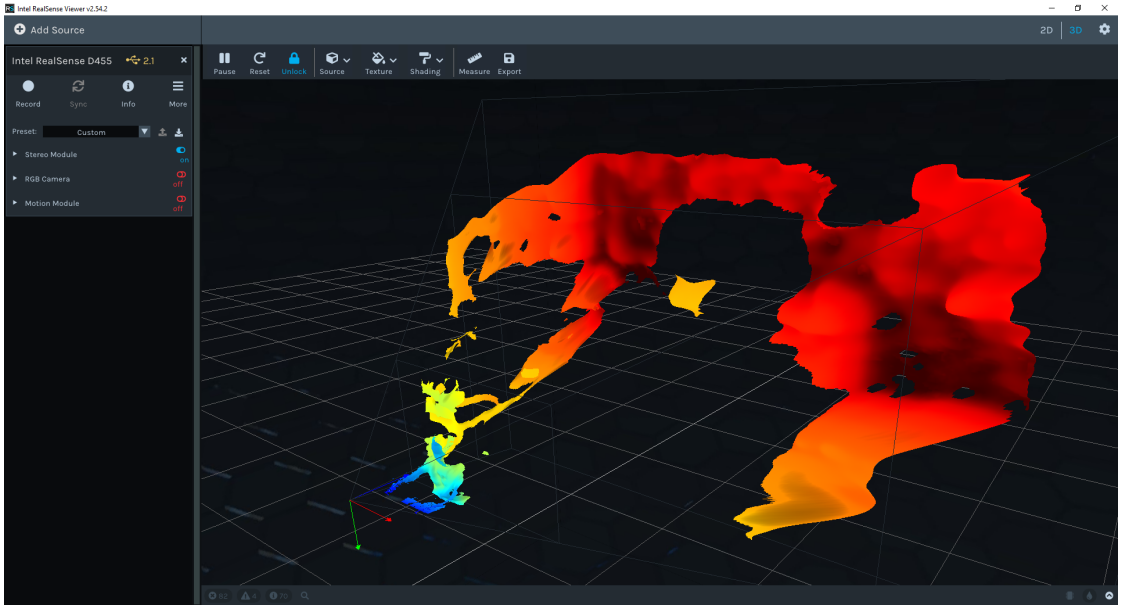
Realsense wrapper je ROS knižnica pomocou ktorej je možné integrovať zariadenia RealSense do systému ROS. Pomocou tejto knižnice je možné získavať všetky typy obrazov a dáta z inerciálnej jednotky. Príkaz pre inštaláciu a príklad spustenia je v prílohe B.4. Pomocou vlastných launch files je možné spustiť jednotlivé komponenty potrebné pre danú aplikáciu.

5.5 RealSense Viewer

RealSense Viewer je aplikácia od spoločnosti Intel pomocou ktorej je možné overiť funkčnosť kamery. Aplikácia taktiež umožňuje konfiguráciu parametrov kamery ako počet snímkov za sekundu, rozlíšenie a mnoho ďalších parametrov. Pomocou aplikácie je napr. taktiež vytvárať 3D modely.

5.6 Kalibrácia kamery

Prostredie Intel RealSenseViewer umožňuje automatickú kalibráciu kamery. Kalibrácia zahŕňa kalibráciu ohniskovej vzdialenosti a skreslenia šošovky. Po vykonaní kalibrácie je možné konfiguráciu uložiť priamo v kamere. Na obrázku 5.1 je znázornené prostredie Intel RealSenseViewer po vykonaní automatickej kalibrácie. [21]



Obr. 5.1: Prostredie Intel RealsenseViewer

6 Vyčítavanie obrazu z kamery a IMU

Pre komunikáciu s kamerami Realsense bol použitý ROS *wrapper* od firmy Intel. Intel RealSense ROS2 *wrapper* je softvérový balíček, ktorý umožňuje integráciu Intel RealSense kamier do ROS2. Tento ROS node zdieľa témy s jednotlivými obrazmi z kamery a dátami z IMU jednotky. Tento uzol taktiež ponúka rôzne možnosti *post-processingu* získaných dát ako napr. zrovnanie hĺbkového obrazu k RGB obrazu, prípadne zafarbenie hĺbkového obrazu k vzdialenosti. Následne je možné zo získanými dátami ďalej pracovať.

6.1 RealSense kamera ROS node

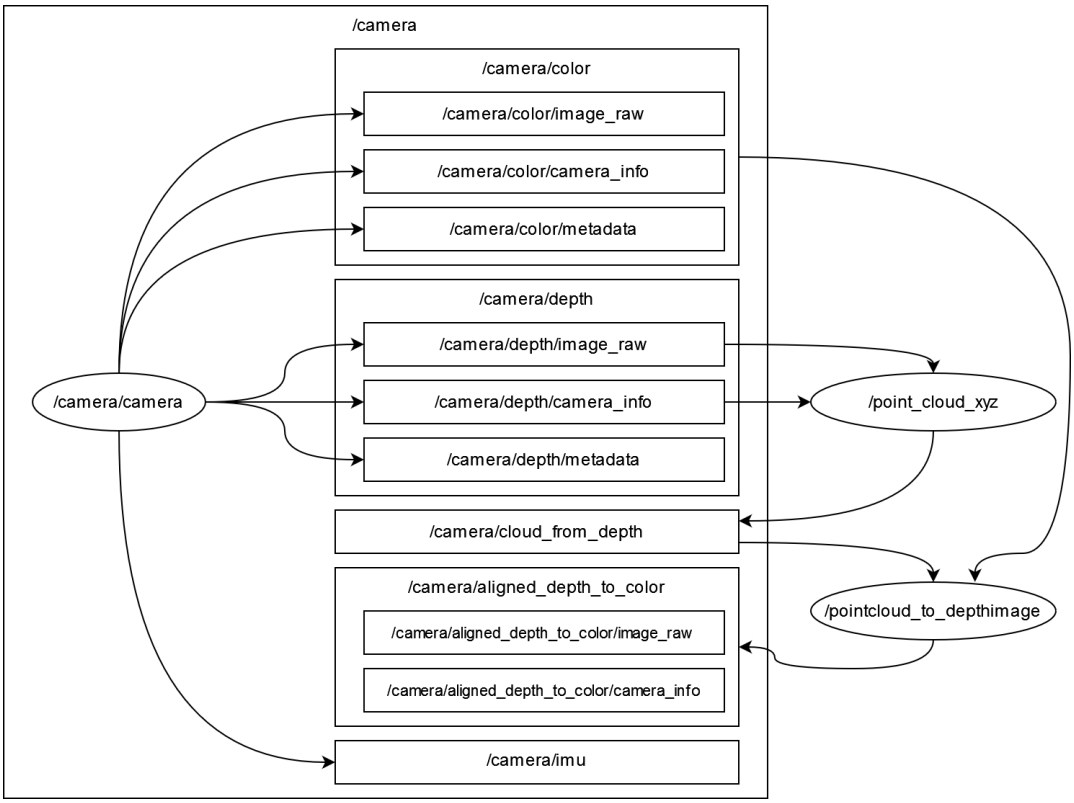
Jednotlivé súčasti uzlu kamery sú rozdelené do logických celkov, ku ktorým je možné pristupovať. K obrazovým dátam RGB kamery je možné pristupovať pomocou topicu **/camera/color**. Takisto k dátam hĺbkovej kamery je možné pristupovať pomocou témy **/camera/depth**. Súčasťou oboch kamier sú taktiež ďalšie súčasti ako **camera_info**, **image_raw**, **metadata**. Obsahom témy **camera_info** sú informácie o kalibrácií danej kamery. K obrazu z kamery je možné pristupovať pomocou témy **image_raw**. Pomocou témy **metadata** je možné pristupovať k metadátam kamery a ukladať ich vo formáte **json**. Obsahom metadát sú napr. model kamery, sériové číslo, kalibračné dáta, časová pečiatka, jednotky v ktorých kamera meria vzdialenosť a teplota.

K inerciálnym dátam z akcelerometru a gyroskopu je možné pristupovať pomocou témy **/camera/imu**. Prípadne je možné pristupovať k jednotlivým komponentom samostatne pomocou tém **/camera/gyro** a **/camera/accel**. Tak ako aj pri obrazových dátach súčasťou je info téma **imu_info**, téma s metadátami **metadata** a téma s meranými dátami **sample**.

Taktiež je možné vyčítavať hĺbkový obraz zrovnaný k RGB obrazu. Tento obraz je zdielaný pomocou **/camera/aligned_depth_to_color/image_raw**. Tento obraz sa v praxi používa pri aplikáciach kde dochádza k skenovaniu, rekonštrukcií 3D priestoru nakoľko je potrebné ku každému bodu z hĺbkovej kamery priradiť farbu.

Na obrázku 6.1 je znázornené prepojenie jednotlivých tém v ROS uzle pre kameru. Ako môžeme vidieť uzol vyčítava obrazové a inerciálne dáta z kamery a zdieľa ich na príslušné témy. K získavaniu zrovnaného hĺbkového obrazu sa používajú dva uzly **/point_cloud_xyz** a **/pointcloud_to_depthimage**. Uzol **/point_cloud_xyz** načítava hĺbkový obraz a transformuje jednotlivé body do karteziánskych súradníc. Výsledný *point cloud* je uložený do témy **/camera/cloud_from_depth**. Následne je k jednotlivým bodom z *pointcloudu* priradená farba z RGB

obrazu pomocou komponenty **/pointcloud_to_depthimage**. Komponent zároveň opäť prekonvertuje *point cloud* na hĺbkový obraz, ktorý má ku každému bodu priradenú farbu. Výsledný zrovnaný hĺbkový obraz je uložený do témy **/camera/aligned_depth_to_color**. V prílohe B.5 sú vypísané všetky témy, ktoré sú zdieľané ROS2 RealSense wrapperom.[22]



Obr. 6.1: Štruktúra uzlu RealSense kamery

6.1.1 Spustenie nodu

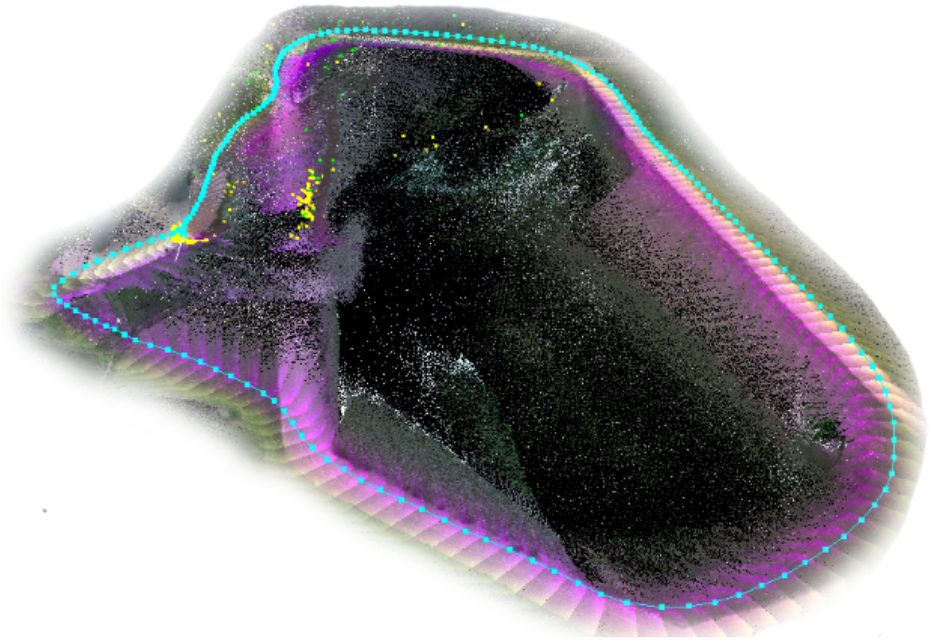
RealSense kamera node je možné spustiť pomocou *launch* súboru **rs_launch.py**, ktorý je súčasťou *wrapperu*. Pomocou vstupných parametrov je možné konfigurovať výstupné obrazy, IMU a iné potrebné parametre. V prílohe B.4.1 je príklad spustenia kamery aj IMU jednotky. Medzi parametrami je povolenie zrovnaného obrazu, povolenie gyra a akcelerometru a nastavenie ako budú ukladané dáta IMU jednotky.

7 Výber vhodného algoritmu

V rámci tejto kapitoly sa budeme zaoberať výberom vhodného algoritmu pre fúziu vizuálnej a inerciálnej odometrie.

7.1 RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) je SLAM algoritmus založený na spracovaní dát z RGB-D, stereo kamier a prípadne lidaru, určený na získavanie trajektórie. Tento algoritmus je založený na inkrementálnom detektore zhody obrazu s detekciou predošlých príznakov. Tento princíp je prakticky založený na relatívnej vizuálnej odometrií s tým rozdielom, že dochádza k detekcií zhody s predošlými snímkami. Týmto spôsobom je eliminovaný problém akumulovania chyby estimácie polohy v čase. *Loop closure detection* využíva metódu BoW (bag-of-words) na určenie pravdepodobnosti aká je miera zhody medzi novým obrazom a predchádzajúcou lokalitou alebo či prípadne predstavuje novú lokalitu. V prípade, že dôjde k *loop*



Obr. 7.1: Príklad získanej trajektórie pomocou algoritmu RTAB-Map[27]

closure detection, čiže k detekcií predošlého miesta na základe rozpoznania vzorov vo vizuálnych prvkov, algoritmus spätne opraví akumulovanú chybu v trajektórií. K tejto detekcií je potrebné ukladať väčšie množstvo predošlých snímkov. V prípade, že by nedochádzalo k obmedzovaniu počtu ukladaných snímkov by s časom narastala výpočtová náročnosť, nakoľko by bolo potrebné prechádzať väčšie množstvo dát. K eliminácií tohoto problému RTAB-Map implementuje taktiež správu pamäte, ktorá

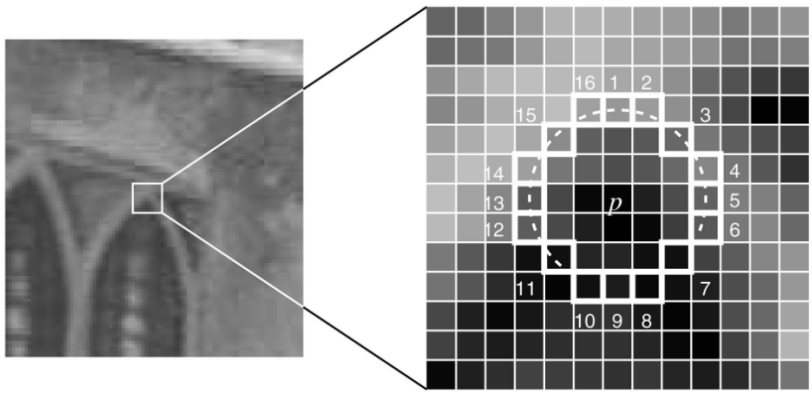
obmedzuje počet lokalít určených na *loop closure detection*. Týmto je zabezpečené, že výpočtový výkon bude dostatočný na porovnávanie v reálnom čase. RTAB-Map zároveň umožňuje aj implementáciu fúzie s inerciálnou odometriou. Výhodou je taktiež kompatibilita s ROS 1 a ROS 2. Na obrázku 7.1 je znázornený príklad získanej trajektórie pri chôdzi v okolí kopca pri budove T12 Fakulty elektrotechniky a komunikačních technologií.[23][24]

7.1.1 Detekčné algoritmy príznakov

RTAB-Map používa k detekcií príznakov dva prístupy ORB(Oriented FAST and Rotated BRIEF) and SURF(Speeded-Up Robust Features). Oba prístupy majú svoje pozitívne a negatívne vlastnosti.

ORB(Oriented FAST and Rotated BRIEF)

FAST (Features from accelerated segment test) je jedným z najrýchlejším detektorov príznakov. Je vhodný pri použitie v *real-time* aplikáciach nakoľko je schopný efektívne identifikovať významné body v obraze. Princíp detekcie príznakového bodu je založený na identifikácií kruhovej oblasti v obraze v okolí vyhodnocovaného pixelu. Hodnota intenzity pixelov v kruhovej oblasti je porovnávaná s intenzitou stredného pixelu. V praxi sa používa okolie od 1 pixelu až po 16 pixelov. V prípade, že jasový rozdiel medzi stredným pixelom a okolím je väčší ako nastavený prah pixel je považovaný za kľúčový bod. Na obrázku 7.2 je znázornený príklad detekcie príznaku algoritmom FAST. Najväčšou nevýhodou tohto algoritmu je nízka odolnosť oproti

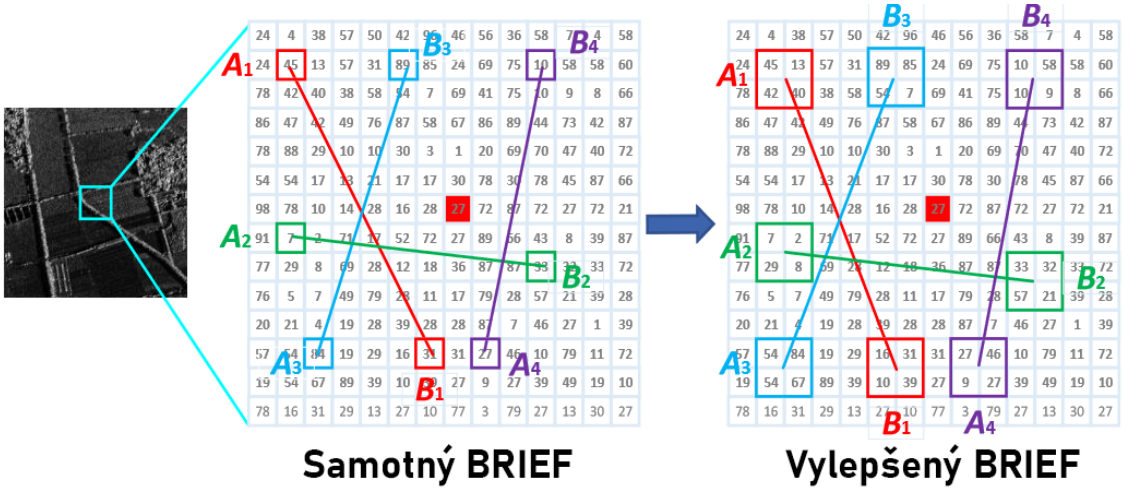


Obr. 7.2: Príklad detekcie algoritmom FAST

šumu a rozmazaniu obrazu. Táto metóda má nízku opakovateľnosť, nakoľko produkuje veľké množstvo príznakov, ktoré sa veľmi rýchlo objavujú a zanikajú medzi jednotlivými snímkami.[25][26]

BRIEF

BRIEF (Binary Robust Independent Elementary Features) je algoritmus pre popis príznakov z algoritmu FAST. Jeho výstupom je binárny deskriptor príznakov, ktorý vyjadruje vzťah príznaku k jeho okoliu. Binárna hodnota deskriptoru je zadávaná na základe porovnania intenzity dvojice bodov. Pred aplikáciou algoritmu dochádza k predspracovaniu obrazu pomocou aplikácie Gaussového filtru aby došlo zvýšeniu odolnosti oproti šumu. Porovnávané dvojice sa vyberajú na základe Gaussového rozloženia, prípadne náhodným výberom. V praxi má výsledný deskriptor tvar reťazca z dĺžkou **n**, kde **n** má hodnotu 128, 256, prípadne 512. Výhodou tohto algoritmu je predovšetkým rýchlosť spracovania, nakoľko dochádza iba k porovnaniu intenzity bez počítania komplexných výpočtov. Nevýhodu je nedostatočná rotačná invariantnosť algoritmu, nakoľko pri zmene rotácie objektu môže dochádzať k chybným detekciám. Tento problém rieši upravená verzia algoritmu **Rotated BRIEF**. Pred popisom vzťahov dochádza k odhadu orientácií rotácie. Okolie príznaku je rozšírené, a následne sa generujú páry v odhadovanej orientácií rotácie. Týmto je zabezpečené, že popisy príznakov zostávajú invariantné oproti rotácií. Na obrázku 7.3 sú znázornené príklady spájania príznakov deskriptormi. Na ľavej strane je znázornený samotný BRIEF algoritmus, na pravej strane je znázornený vylepšený algoritmus **Rotated BRIEF**.[27]



Obr. 7.3: Príklad spájania príznakov desktriptormi BRIEF[27]

SURF

SURF (Speeded-Up Robust Features) je populárny algoritmus pre popis a extrakciu príznakov z obrazu. Na rozdiel od algoritmu BRIEF, tento algoritmus k opisu príznaku nepoužíva iba binárne hodnoty ale používa hodnoty gradientu kruhového

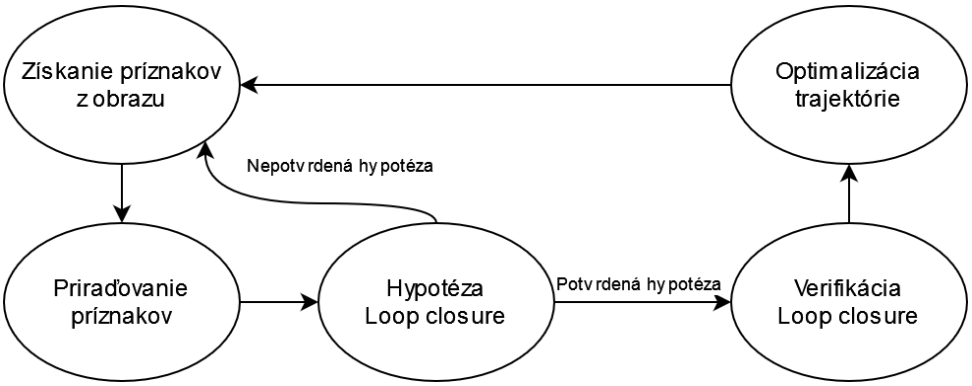
okolia bodu, smer gradientu, intenzitu, váhy deskriptoru, prípadne priemernú hodnotu pixelov vo vybranej oblasti. K detekcií príznakov je použitá technika, ktorá porovnáva rozdiely obrazov v rôznych mierkach, ktoré sú vyhľadené Gaussovským filtrom. Gradientné extrémy sú považované za kľúčové body. Následne je pre každý kľúčový bod priradená orientácia analýzou gradientov v blízkom okolí kľúčového bodu. Táto informácia zabezpečuje rotačnú invariantnosť, takže algoritmus dokáže rozpoznávať príznaky v obraze bez ohľadu na ich orientáciu v obraze. Výhodou tohto algoritmu je rýchlosť, invariancia oproti zmene mierky a šumu a zmene osvetlenia. Nevýhodou v porovnaní s algoritmom BRIEF je vyššia pamäťová náročnosť, nakoľko ku každému kľúčovému bodu je potrebné ukladať väčšie množstvo dát. Na obrázku 7.4 je znázornený príklad použitia algoritmu SURF. Ako môžeme vidieť na obrázku algoritmus môže na prvý dojem pripomínať nejaký kruhový detektor.[28][29]

Obr. 7.4: Príklad použitia algoritmu SURF[29]

7.1.2 Loop closure detection

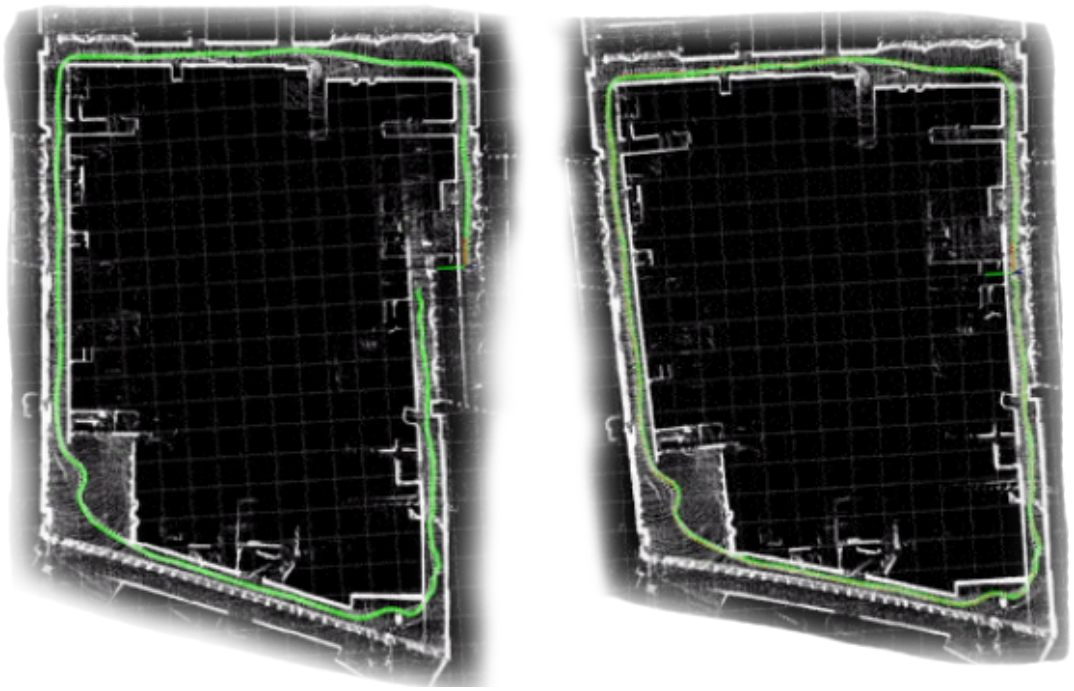
Je technika, ktorá sa zaoberá detekciou predošlých navštívených miest s cieľom optimalizovať presnosť určenia polohy v priestore. K detekcií sa používajú vizuálne príznaky, prípadne deskriptory. Na obrázku 7.5 je znázornený diagram priebehu detekcie *loop closure*. Ako prvé sú z obrazu získané vizuálne prvky a deskriptory pomocou algoritmov ORB a SURF. V ďalšom kroku dochádza k porovnávaniu získaných informácií z obrazu s obrazmi v databáze. Následne sa vyhodnocuje potenciálna hypotéza, či už bolo aktuálne miesto navštívené. Ak dôjde k zhode vizuálnych prvkov tak dôjde k ďalšej úrovni verifikácie zhody z predošlým obrazom na základe geometrickej

konzistencie obrazu alebo pravdepodobnostnými metódami. V prípade, že hypotéza nebola potvrdená, proces sa opäť opakuje od začiatku. Po verifikácii detekcie dochádza k optimalizácií trajektórie. Na obrázku 7.6 je znázornený príklad detekcie *loop*



Obr. 7.5: Diagram priebehu detekcie loop closure

closure na trajektórii. Ako môžeme vidieť na ľavej strane obrázku, bez aplikácie *loop closure* by došlo k chybnému určeniu trajektórie. Na pravej strane obrázku môžeme vidieť upravenú trajektóriu pomocou *loop closure*. Táto technika eliminuje chybu,



Obr. 7.6: Príklad detekcie loop closure [30]

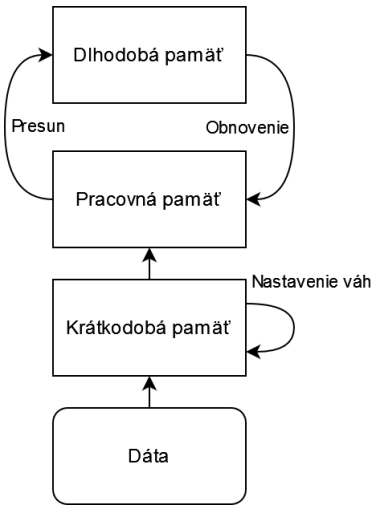
ktorá vzniká pri tvorbe trajektórie pomocou vizuálno-inerciálnej odometrie, nakoľko obidva prístupy postupne akumulujú chybu merania zmeny. Jedinou podmienkou je, že musí dôjsť k návratu na miesto, ktoré už bol predtým navštívené.[23][30]

7.1.3 Bag of words

Je metóda pre určenie pravdepodobnosti zhody predošlého obrazu s aktuálnym. Táto metóda sa používa pre verifikáciu hypotézy o možnom *loop closure*. V prvej fáze je potrebné vytvoriť slová (words), ktoré budeme následne porovnávať. V tomto prípade sa jedná o vizuálne rysy, čiže detekované príznaky, prípadne detektory. Následne dochádza k extrakcií vizuálnych prvkov z aktuálneho obrazu a porovnávaniu s databázou slov. V ďalšom kroku je vyhodnocovaná štatistika výskytu jednotlivých slov v danom snímku. Výsledkom je histogram výskytu jednotlivých vizuálnych prvkov, ktorý je porovnávaný s kľúčovými snímkami v databáze. Histogram aktuálneho obrazu je porovnávaný s predošlými. V prípade, že dôjde k zhode, ktorá bude presahovať preddefinovanú hodnotu, obraz bude považovaný za zhodný.[23][31]

7.1.4 Správa pamäte

Je technika, ktorá v spojení s RTAB-Map používa pre obmedzenie aktuálne porovnávaných lokácií pri detekcií *loop closure*. Tento komponent RTAB-Mapu je kľúčovou súčasťou pre možnú aplikáciu detekcie *loop closure* v reálnom čase. Na obrázku 7.7 je znázornený diagram správy pamäte. V rámci krátkodobej pamäte prebieha extrakcia, spracovanie príznakov, deskriptorov a tvorba *bag-of-words*. Na základe stráveného času na mieste je zvyšovaná váha danej lokácií. Po vytvorení lokácie sú



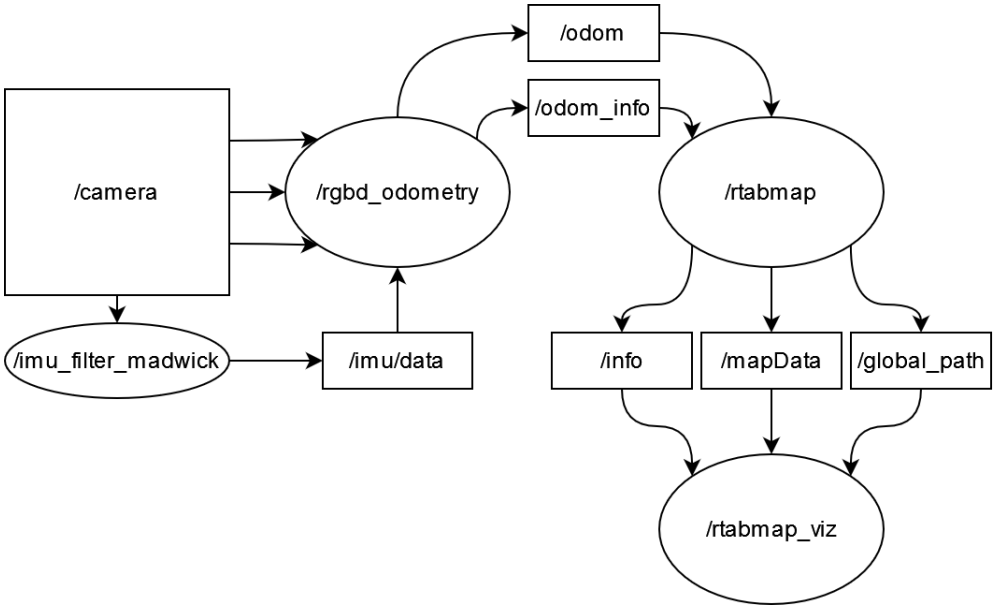
Obr. 7.7: Diagram správy pamäte

dáta presunuté do pracovnej pamäte. V tejto časti pamäte prebieha detekcia *loop closure*. Cieľom je uržiavať rýchlosť spracovania údajov v reálnom čase. Pokým výpočet prebieha do určitého času, nové lokácie budú pribúdať do pracovnej pamäte. V prípade, že čas spracovania nebude dostatočný, najstaršie dáta a dáta s najnižšími

váhami budú presunuté do dlhodobej pamäte. V prípade detekcie *loop closure* môžu byť najpodobnejšie lokácie opäť presunuté do pracovnej pamäte. [31]

7.1.5 ROS štruktúra RTAB-Map

Na obrázku 7.8 je znázornené prepojenie jednotlivých uzlov RTAB-Mapu. Uzol **/rgbd_odometry** spracováva dáta z kamery a IMU jednotky. Výstupom tohto uzlu je topic **/odom**, ktorého obsahom je poloha a orientácia v priestore. Taktiež je zdieľaný topic **odom_info**, ktorého obsahom sú doplňujúce informácie k topicu **/odom**. Tieto dáta sú následne spracované v uzle **/rtabmap**, ktorého výstupom je mapa a trajektória. Pre vizualizáciu je možné použiť uzol **rtabmap_viz**, prípadne vstavaný nastroj ROS-u **rviz**. K spusteniu je potrebný spúšťací súbor, pomocou



Obr. 7.8: Prepojenie RTAB-Map v ROSe

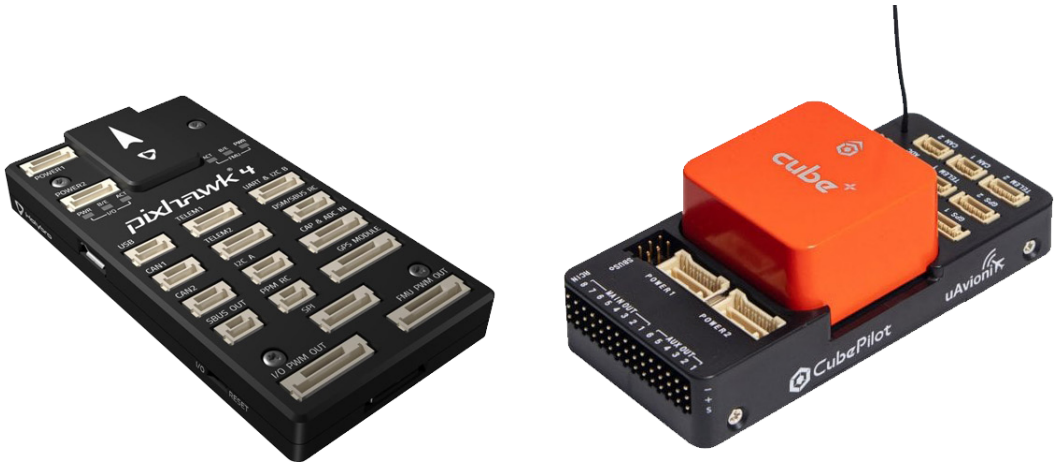
ktorého sa spustia všetky potrebné uzly s správnym mapovaním tém a potrebné konfiguračné parametre. Príklad spustenia vlastného spúšťacieho súboru je v prílohe B.6.2.

8 Ovládanie bezpilotného prostriedku

V tejto kapitole sú popísané možnosti ovládania bezpilotného prostriedku, vrátane rôznych letových režimov. Je diskutované možné použitie daných letových režimov v spojení s vizuálnou odometriou. Taktiež je popísaná použitá letová jednotka.

8.1 Letová jednotka

Na testovacej platforme bezpilotného prostriedku Holybro X500 je inštalovaná letová jednotka ***(FC - Flight Controller)*** Holybro Pixhawk 4. V letovej jednotke je inštalovaný PX4 firmware. Tento typ letovej jednotky sa v súčasnosti už nevyrába a často je nahradený novšími modelmi od firmy CubePilot. Na obrázku 8.2 sú znázornené spomínané letové jednotky.



Obr. 8.1: Letové jednotky Pixhawk 4 a Cubepilot Orange[33]

8.2 Letové režimy

Súčasťou firmwaru PX4 sú rôzne letové režimy bezpilotného prostriedku, ktoré sú popísané v tabuľke 8.1. Letové režimy ako **Manual**, **Altctl**, **Posctl**, **Acro** a **Stabilized** sú predovšetkým určené pre ovládanie bezpilotného prostriedku pomocou RC (Radio Control) vysielača pilotom. Tieto letové režimy sa odlišujú stupňom stabilizácie bezpilotného prostriedku. Letový režim **Altctl** sa používa pre udržiavanie konštatnej výšky letu pomocou barometru. Najviac stabilizovaným režimom je **Posctl**, ktorý využíva k stabilizácií aj GPS polohu. Ostatné spomínané letové režimy využívajú iba horizontálnu a vertikálnu stabilizáciu, prípadne sú bez akejkoľvek stabilizácie ako napr. **Acro** a **Manual**.

Letový režim **Mission** umožňuje programovanie a vykonávanie automatizovaných misií. V tomto režime môžu byť definované rôzne úlohy a trajektórie, ktoré má bezpilotný prostriedok vykonávať podľa preddefinovaných GPS súradníc a príkazov. Režim **Hold** umožňuje bezpilotnému prostriedku udržiavať svoju pozíciu v priestore bez pohybu. Tento letový režim sa často používa v situáciách, keď je potrebné stabilizovať bezpilotný prostriedok na určitom mieste, napríklad pri čakaní na ďalšie pokyny alebo pri zachovaní poslednej pozície.

Letové režimy **Takeoff** a **Land** sa využívajú pre vykonanie automatického štartu a pristátia bezpilotného prostriedku. Režim **ReturnToLaunch** umožňuje bezpilotnému lietadlu automatický návrat na štartovaciu pozíciu.

Mode	**Popis**
Unknown	Stav nie je známy.
Ready	UAV je pripravený na vzlet.
Takeoff	Režim pre automatický vzlet
Hold	Režim pre stabilizáciu vo všetkých osiach s GPS na jednom mieste
Mission	Režim misie
ReturnToLaunch	Režim pre návrat na štartovaciu pozíciu.
Land	Režim pre automatické pristátie
Offboard	Režim pre kontrolu externým zdrojom
FollowMe	Režim pre nasledovania
Manual	Režim manuálneho ovládania s RC vysielačom
Altctl	Režim pre udržiavanie konštatnej výšky
Posctl	Režim pre stabilizáciu vo všetkých osiach s GPS
Acro	Režim bez stabilizácie
Stabilized	Režime stabilizície

Tab. 8.1: Letové režimy pre PX4 autopilot[33]

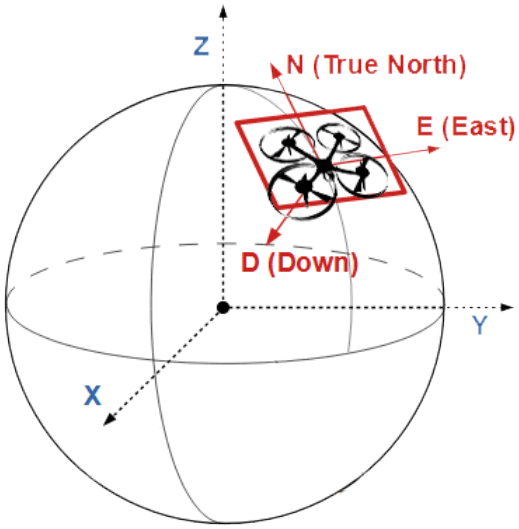
Letový režim **Offboard** patrí k pokročilým režimom riadenia bezpilotných prostriedkov, ktorý umožňuje externému riadiacemu systému riadiť pohyb bezpilotného prostriedku prostredníctvom počítača alebo iného externého zariadenia. Tento režim je najvhodnejší pre riadenie bezpilotného prostriedku externým riadiacim systémom nakoľko umožňuje riadiť pohyb UAV vo všetkých osiach.[33]

8.2.1 Letový režim offboard

Tento letový režim umožňuje riadenie polohy bezpilotného prostriedku pomocou rôznych súradnicových systémov.

NED (North-East-Down)

NED je súradnicový systém, ktorý sa v doslovnom preklade označuje ako Sever-Východ-Dole. Je to lokálny súradnicový systém často používaný v leteckých a vesmírnych aplikáciách na popis polohy a pohybu objektu vzhľadom k určitému referenčnému bodu. Smer severu v súradniciach NED je pozitívny smer pozdĺž lokálnej osi severu. Smer východu v súradniciach NED je pozitívny smer pozdĺž lokálnej osi východu. Je kolmý na smer severu a smeruje k východu. Smer dolu v súradniciach NED je pozitívny smer pozdĺž lokálnej osi dole. Je kolmý na smer severu a východu a smeruje smerom k stredu Zeme. Podobne ako aj ECEF je tento súradnicový sys-



Obr. 8.2: Súradnicový systém NED

tém kartétzsky. Narozdiel od ECEF, ktorý má referenčný bod v geografickom strede Zeme, má NED súradnicový systém referenčný bod vzhľadom k aktuálnej polohe UAV. Tento súradnicový systém je obľúbený aj vzhľadom na jeho intuitívnosť pri riadení.[34]

Global

Offboard riadenie pomocou **Global** príkazov umožňuje externému systému riadiť bezpilotný prostriedok a určovať jeho cieľové body pomocou súradníc GPS alebo iného globálneho súradnicového systému. Avšak nakoľko pre riadenie pomocou týchto príkazov je potrebná dostupnosť GPS dát, nie je vhodné tento spôsob riadenia použiť pre riadenie bezpilotného prostriedku pomocou vizuálnej odometrie.

Body

Tento súradnicový systém sa používa na určenie polohy a orientácie bezpilotného prostriedku v priestore. Referenčný bod tohto súradnicového systému je zvyčajne umiestnený v ťažisku bezpilotného prostriedku. Jedná sa svetový súradnicový systém, ktorý je rotovaný okolo osi ***yaw***. To znamená, že pri otočení po osi ***yaw*** bude vždy smer dopredu kladná osa x. Tým je zabezpečené, že aj keď bezpilotný prostriedok otáča okolo osi ***yaw***, smer pohybu dopredu zostáva konzistentný a nezávislý od zmeny ***yaw***.

Attitude

Je spôsob ovládania, ktorým je možné pomocou príkazov priamo ovládať náklon vo všetkých osiach. Okrem náklonu umožňuje aj kontrolu nad uhlovou rýchlosťou okolo všetkých osí. Pri obidvoch spôsoboch ovládania je taktiež potrebné kontrolovať základný ťah motorov, ktorým je zároveň možné meniť výšku bezpilotného prostriedku.

8.3 Implementácia komunikácie s FC

Pre komunikáciu s letovou jednotkou je použitá knižnica MAVSDK. Táto knižnica implementuje MAVLink (Micro Air Vehicle Link) správy, pomocou ktorých je možné zasielať príkazy pre ovládanie bezpilotného prostriedku a taktiež umožňuje načítavanie telemetrických dát. Pomocou knižnice MAVSDK je možné komunikovať s letovou jednotkou pomocou rôznych programovacích jazykov ako C++, Python, Java, Javascript, Rust. Pomocou tejto knižnice bol vytvorený ROS node, pomocou ktorého je možné riadiť činnosti bezpilotného prostriedku ako napr. štart, pristátie a kontrola polohy.

8.4 Protokol MAVLink

MAVLink je protokol používaný na komunikáciu, prenos riadiacich signálov, informácií o stave bezpilotného prostriedku a ďalších konfiguračných správ medzi bezpilotným prostriedkom a pozemnou riadiacou stanicou. Zdieľanie informácií prebieha rôznymi spôsobmi, čiže prostredníctvom publish-subscribe (pre odosielanie dátových tokov) a bod-ku-bodu (pre konfiguráciu, misie, alebo parametre). Na prenos konkrétnych informácií sú definované typy správ podporované konkrétnym systémom MAVLink. [35]

8.4.1 Point to point

Tento typ správ sa využíva pre riadenie systému s konkrétnym ID v rámci jednej siete. Letová jednotka má svoje jedinečné ID v rámci siete a príkazy riadenia sú určené pre konkrétne ID. Tento spôsob komunikácie zabezpečuje garantované doručenie parametrov, príkazov riadenia a misií ku konkrétnemu systému.[35]

8.4.2 Publish-subscribe

V tomto režime sa správy neposielajú pre konkrétny cieľový systém a ani ID komponenty. Týmto spôsob sa zdieľajú napr. telemetrické dáta o polohe, výške, stavu batérie. Tento typ správy môže prijímať väčší počet užívateľov. K danému typu správy je možné pridať odber a spracovávať daný typ správy.[35]

8.5 Použité triedy MAVSDK

V nasledujúcej podkapitole sú popísané použité triedy s knižnice MAVSDK.

8.5.1 Action

Pomocou tejto triedy je možné nastavovať jednoduché príkazy ako je napr. odistenie motorov (arm), reštartovanie letovej jednotky, vzlet, pristátie, prípadne návrat na štartovaciu pozíciu. Taktiež je možné ovládať polohu bezpilotného prostriedku v globálnom súradnicovom systéme, prípadne udržiavať aktuálnu polohu pomocou letového režimu **hold**.

8.5.2 Telemetry

Táto trieda umožňuje načítavanie telemetrie a stavových informácií o bezpilotnom prostriedku ako napr. aktuálna GPS poloha, pripojenie RC ovládaču, letový režim. K danému typu správy je možné pridať odber, a následne v prípade prijatia správy spracovávať pomocou *callback* funkcií.

8.5.3 Offboard

Trieda offboard prakticky implementuje všetky potrebné príkazy pre ovládanie bezpilotného prostriedku v režime **offboard**. Knižnica implementuje všetky typy ovládanie popísané v kapitole 8.2.1.

8.6 Implementácia riadiaceho uzlu v ROS

V rámci riadiaceho ROS uzlu je implementované komunikácia s letovou jednotkou pomocou knižnice **MAVSDK**. Uzol zároveň odoberá určenú relatívnu polohu v priestore z príslušnej témy. Pomocou určenej relatívnej polohy v priestore je implementované riadenie dronu v priestore pomocou servisov s parametrami. Rôzne typy servisov poskytujú napr. vzlet, pristátie, let na určený bod, prípadne návrat na miesto vzletu. Uzol taktiež poskytuje telemetriu s UAV ako napr. letový režim, stav batérie, aktuálne natočenie, polohu.

8.7 Implementované servisy

V nasledujúcich podkapitolách sú popísané jednotlivé implementované servisy pre ovládanie bezpilotného prostriedku.

V tabuľke 8.2 sú popísané jednotlivé typy použitých služieb pre implementované servisy.

Service	Typ servisu
Vzlet	std_srvs::srv::Trigger
Pristátie	std_srvs::srv::Trigger
Let na bod	visual_interfaces::srv::Position
Návrat na štart	std_srvs::srv::Trigger

Tab. 8.2: Implementované services

8.7.1 Vzlet

Pred samotným vzletom je potrebné odistiť motory pomocou takz. **arm**. Následne je možné poslať letovej jednotke príkaz pre vzlet. Obidva príkazy je možné realizovať pomocou triedy **Action** v **MAVSDK**. Následne sa čaká na dokončenie vzletu počas ktorého je monitorovaný stav. Pre tento servis bol použitý vstavaný servis **std_srvs::srv::Trigger**. Po dokončení servisu je na základe štádia dokončenia nastavená hodnota premenných **success** a **message**.

8.7.2 Pristátie

Tento servis je prakticky dosť podobný servisu pre vzlet, s tým rozdielom, že pri pristátí nie je potrebné posielať príkaz na **arm**. Je potrebné iba posielať príkaz na pristátie a počkať a monitorovať stav, kým bezpilotný prostriedok nepristane. Následne dôjde k **disarm**-u, čiže k vypnutiu motorov.

8.7.3 Let na bod

Pre let na určený bod je na základe aktuálnej relatívnej polohy v priestore vypočítaný uhol medzi požadovaným bodom a aktuálnou polohou pomocou nasledujúcich vzťahov:

$$\Delta x = \text{target}_\text{x} - \text{uav}_\text{x} \tag{8.1}$$

$$\Delta y = \text{target}_\text{y} - \text{uav}_\text{y} \tag{8.2}$$

$$\text{angle\_rad} = \arctan\left(\frac{\Delta y}{\Delta x}\right) \tag{8.3}$$

$$\text{angle\_deg} = \text{angle\_rad} \times \frac{180.0}{\pi} \tag{8.4}$$

Následne je potrebné uhol prepočítať na uhol kompasu vzľadom na to, že kladná osa y bude predstavovať sever. Preto je potrebné upraviť vzťah 8.3 na

$$\text{angle\_degrees} = 360.0 - \text{angle\_radians} \times \frac{180.0}{\pi} + 90.0 \tag{8.5}$$

Následne je počas letu k danému bodu uhol k danému bodu priebežne prepočítavaný kvôli možnej deviácií bezpilotného prostriedku dôsledkom poveternostných podmienok. Taktiež je konštantne počítaná euklidovská vzdialenosť medzi aktuálnou polohou a požadovaným bodom pomocou nasledujúcich vzťahov:

$$\text{dist2D}(x_1, y_1, x_2, y_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{8.6}$$

$$\text{dist3D}(x_1, y_1, z_1, x_2, y_2, z_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{8.7}$$

Po dosiahnutí daného bodu je ešte vyslaný príkaz, pomocou ktorého sa nastaví zastavenie pohybu vo všetkých osiach. Tento typ servisu je implementovaný pomocou vlastného typu servisu, kde má požiadavka (**request**) dve vstupné premenné **x**, **y**, ktoré reprezentujú požadované súradnice bodu.

8.7.4 Návrat na štart

Prakticky pri návrate na štart sa jedná o kombináciu dvoch predošlých servisov pre let na bod a pristátie. Ako miesto štartu môžeme uvažovať počiatok súradnicového systému, teda bod 0,0,0. Prípadne je možné bod štartu definovať ako iný bod v priestore. Avšak bezpilotný prostriedok vrátane vizuálno-inerciálnej odometrie je zvyčajne inicializovaný v mieste štartu.

8.8 Exekútor

Exekútor využíva jedno prípadne väčší počet vlákien operačného systému pre vyvolávanie jednotlivých callback funkcií ako odoberanie, servis, časovače, spracovanie správ a mnoho ďalších. V ROS 2 sú k dispozícií tri typy exekútorov. Najviac používaným je **SingleThreadedExecutor** (jednovláknový), ktorý vykonáva všetky činnosti sekvenčne pomocou jedného vlákna. Tento typ je vhodný pre jednoduché aplikácie kde nie je potrebné vykonávať väčšie množstvo činností súčasne.

Druhým v poradí je **MultiThreadedExecutor** (viac-vláknový), tento typ exekútora využíva väčší počet vlákien súčasne na spracovanie udalostí, čo umožňuje vykonávanie úloh paralelne. Posledný typ exekútora je **StaticSingleThreadedExecutor** (statický jedno-vláknový), ktorý má vlákno vyhradené len pre spracovanie udalostí a nie je dynamicky priraďované alebo uvoľňované na rozdiel od viac-vláknového. Statické priraďovanie vlákien pomáha zabezpečiť konzistentné správanie pri spracovaní udalostí.[38]

8.8.1 Použitý exekútor

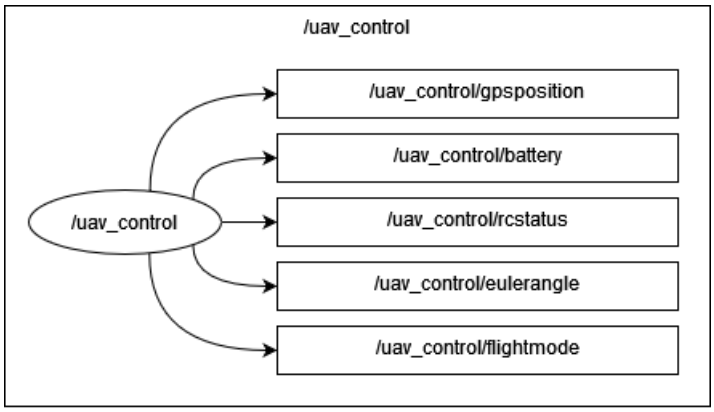
Pre spustenie riadiaceho uzlu je použitý viac-vláknový exekútor, nakoľko uzol poskytuje služby a zároveň odoberá témy z ostatných uzlov. Taktiež je vhodný pre použitie s väčším počtom uzlov, ktoré môže byť vhodné pri budúcom vývoji.

8.9 Implementácia telemetrie

Implementovaný riadiaci uzol zabezpečuje aj poskytovanie telemetrických údajov z bezpilotného prostriedku prostredníctvom odberu údajov z letovej jednotky. Ak sú telemetrické údaje k dispozícii, spracovávajú sa pomocou *callback* funkcií a následne sa zdieľajú v rámci ROS-u prostredníctvom príslušných tém, ktoré je možné vidieť na obrázku 8.3.

ROS 2 umožňuje vytváranie skupín *callback* funkcií v rámci jedného uzlu. Táto schopnosť umožňuje organizovať a riadiť vykonávanie *callback* funkcií, čo je veľmi užitočné pri spracovaní viacerých asynchrónnych udalostí s rôznymi prioritami. K dispozícií sú dve skupiny *callback* funkcií. V skupine **MutuallyExclusive** *callback* funkcie nesmú byť vykonávané paralelne. Na rozdiel od skupiny **Reentrant**, kde funkcie môžu byť vykonávané paralelne. V prípade, že *callback* funkcia nemá priradenú žiadnu skupinu je automaticky priradené do *default*. Funkcie v tejto skupine sa nemôžu vykonávať zároveň napríklad s volaním servisu.[38]

V implementovanom riadiacom uzle sú *callback* funkcie rozdelené do troch skupín. Jednou skupinou sú *callback* funckie, ktoré spracovávajú prijatú telemetriu s



Obr. 8.3: Zdielané témy z riadiaceho uzlu

letovej jednotky. Táto skupina *callback* funkcií nie je priradená k žiadnej skupine v rámci ROS-u, pretože funkcie sú súčasťou knižnice **MAVSDK**. Z tohto dôvodu im nemôže byť pridelená skupina v rámci ROSu. Súčasťou druhej skupiny sú funkcie pre spracovanie dát z iných uzlov. V poslednej skupine sú zahrnuté funkcie pre obsluhu volaných servisov.

8.10 Simulácia v prostredí Gazebo

Pre otestovanie implementovaných funkcií, ovládanie a načítavanie telemetrie bezpilotného prostriedku som využil simuláciu bezpilotného prostriedku v simulačnom prostredí Gazebo v spojení s PX4-SITL. Na obrázku 8.4 je bezpilotný prostriedok počas testovania. Pre testovanie bola zvolená mapa **baylands**.

Obr. 8.4: Bezpilotný prostriedok v prostredí Gazebo

8.10.1 Použitý model

Počas testovania v simulačom prostredí Gazebo bol použitý model **x500_depth**, ktorý je znázornený na obrázku 8.5. Prakticky sa jedná o model reálneho bezpilotného prostriedku Holybro X500 použitého pri prvotných testoch určovania polohy. Súčasťou tohto modelu je taktiež hĺbková kamera, ktorej obraz je možné pomocou **ros_gz_bridge** zdielať ako ROS topic.

Obr. 8.5: Bezpilotný prostriedok X500

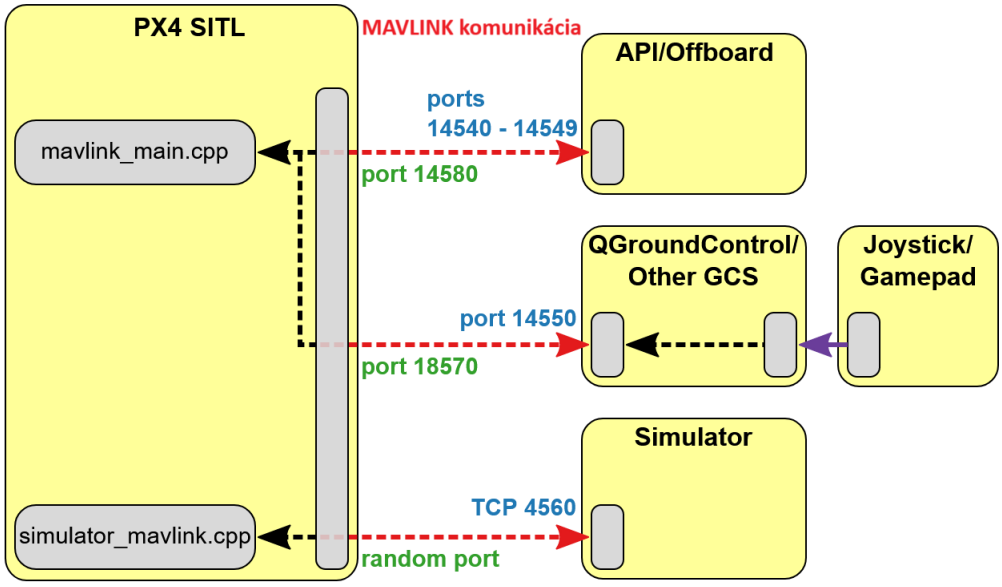
8.10.2 PX4-SITL

Je simulačné prostredie používané pre testovanie a vývoj bez nutnosti fyzického hardvéru. SITL (Software In the Loop) znamená v preklade softvér v slučke, je spôsob simulácie letovej jednotky na počítači. Komunikácia s simulovanou letovou jednotkou je možná pomocou UDP portu 14541 pre komunikáciu s **Offboard API**. Na obrázku 8.6 je znázornená komunikácia v simulačnom prostredí. Ako môžeme na obrázku vidieť pre jednotlivé časti systému sú vyhradené jedinečné porty pomocou, ktorých je možné komunikovať.

Pri spustení simulácie je potrebné spustiť vhodný model s ktorým budeme pracovať. Po správnom načítaní sa model bezpilotného prostriedku zobrazí v spustenom simulačnom prostredí Gazebo.[37]

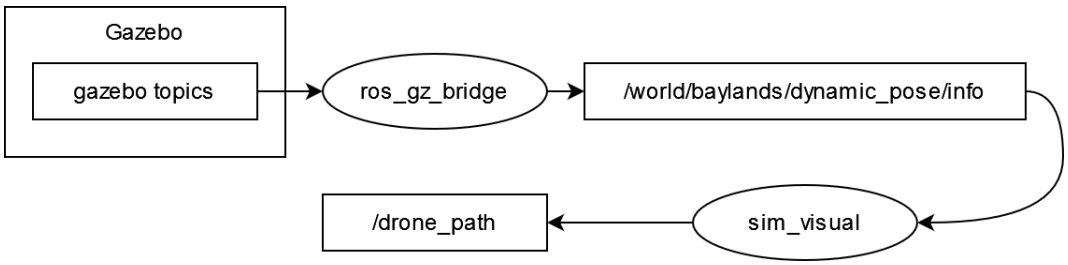
8.10.3 Prevod tém z Gazebo do ROSu

Tak ako aj v rámci systému ROS, súčasťou simulačného prostredia Gazebo sú témy, ktorými sa propagujú informácie s jednotlivých modelov. Pomocou ROS balíčku **ros_gz_bridge** je možné vytvoriť premostenie medzi témami v prostredí Gazebo a ROS. Pre simuláciu je využitá poloha modelu bezpilotného prostriedku v



Obr. 8.6: Komunikácia v simulačnom prostredí PX4 SITL

prostredí Gazebo na určenie relatívnej polohy v priestore. Poloha je následne prevedená pomocou uzlu **ros_gz_bridge** z prostredia Gazebo do ROSu do formátu **geometry_msgs::msg::PoseArray**. Následne je z tejto témy pomocou implementovaného uzlu **sim_visual** vytváraná simulovaná trajektória, ktorá odpovedá formátu výstupu RTAB-Mapu. Výstupom implementovaného uzlu je trajektória typu **nav_msgs::msg::Path**. Tieto simulované dáta odpovedajú výstupu RTAB-Mapu a tak bolo možné odladiť a implementovať riadenie polohy bezpilotného prostriedku v priestore. Na obrázku 8.7 je znázornené prepojenie tém medzi Gazebom a ROS-om



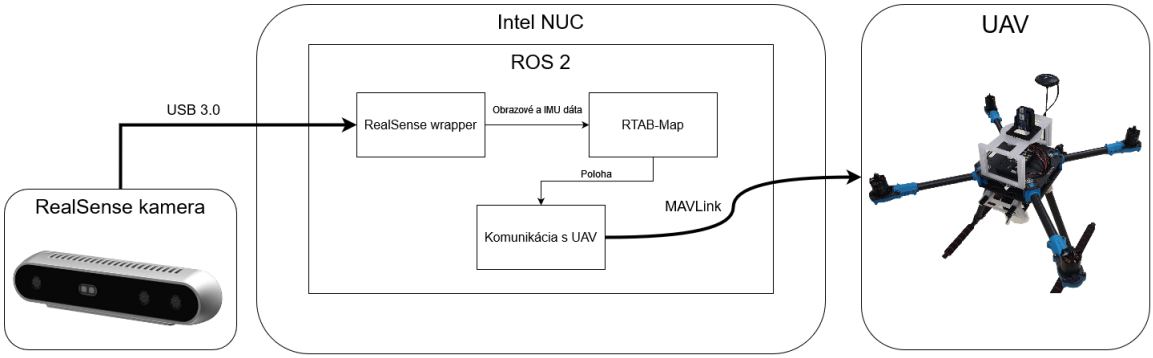
Obr. 8.7: Prevod tém medzi Gazebom a ROSom

9 Softvérová štruktúra

V tejto kapitole je popísané prepojenie jednotlivých komponentov v rámci ROS-u. Taktiež sú popísané použité rozhrania pre prepojenie komponentov.

9.1 Prepojenie jednotlivých komponentov

Na obrázku 9.1 je znázornené prepojenie jednotlivých komponentov. RealSense kamera je k riadiacemu počítaču Intel NUC pripojená pomocou vysokorýchlostného USB 3.0 . Pri pripojení kamery cez USB 2.0 bol zaznamenaný značný pokles v obnovovacej frekvencií snímkov. Z tohto dôvodu je potrebné použiť vysokorýchlostný



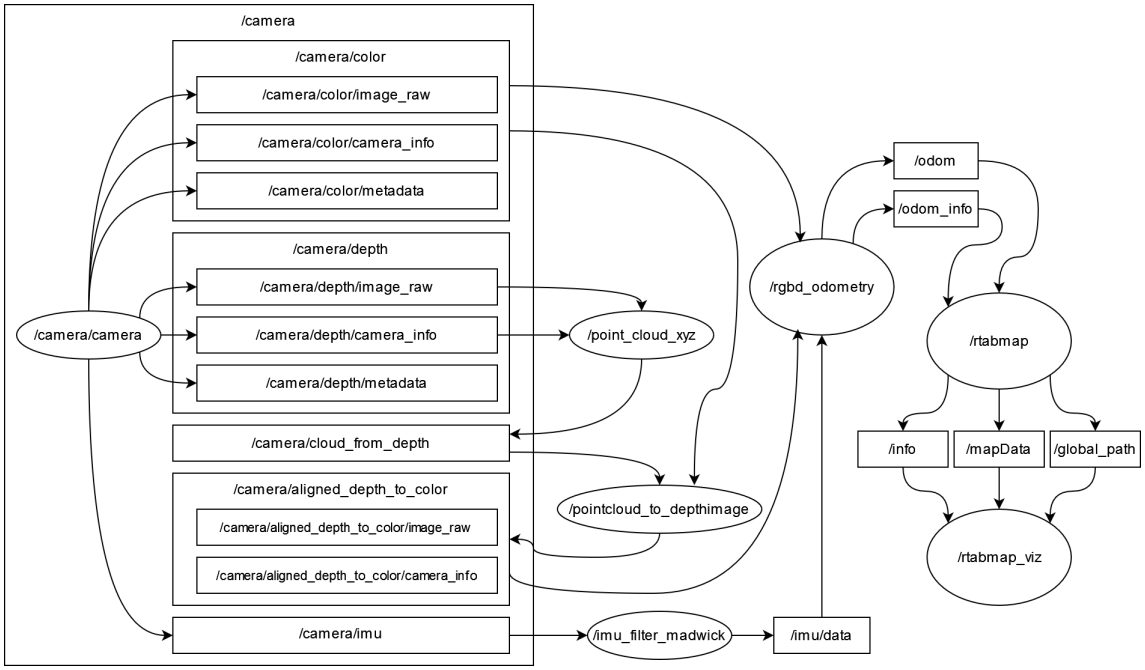
Obr. 9.1: Prepojenie jednotlivých komponentov

USB kábel. Obraz je vyčítaný a upravovaný pomocou RealSense ROS wrapperu. Estimáciu polohy na základe vizuálnych a inerciálnych dát zabezpečuje RTAB-Map, ktorého výstupom je relatívna poloha v priestore. Pomocou zistenej polohy je možné poskytovať relatívnu polohu bezpilotného prostriedku oproti miestu štartu pri výpadku GPS, prípadne je bezpilotný prostriedok taktiež možné riadiť. Komunikácia s letovým kontrolérom bezpilotného prostriedku je zabezpečená pomocou protokolu MAVLink. K prepojeniu počítaču a letového kontroléru je potrebné použiť USB sériový prevodník. Na letovom jednotke je potrebné nakonfigurovať Mavlink výstup na telemetrický port.

9.2 Prepojenie kamery a RTAB-Map

Na obrázku 9.2 je znázornená štruktúra prepojenia jednotlivých uzlov v rámci ROS-u. Jednotlivé obrazy a dáta z kamery sú distribuované cez príslušné témy. Vstupom do nodu **/rqbd_odometry** je rgb obraz (**/camera/color/image_raw, /camera/color/camera_info**), prispôsobený hĺbkový obraz (**/camera/realig-**

ned__depth __to__color/image__raw) a inerciálne dáta (**/imu/data**). Bližší opis prepojenia jednotlivých komponentov je popísaný v kapitolách 6.1 a 7.1.5.



Obr. 9.2: Prepojenie kamery a RTAB-Map v ROS-e

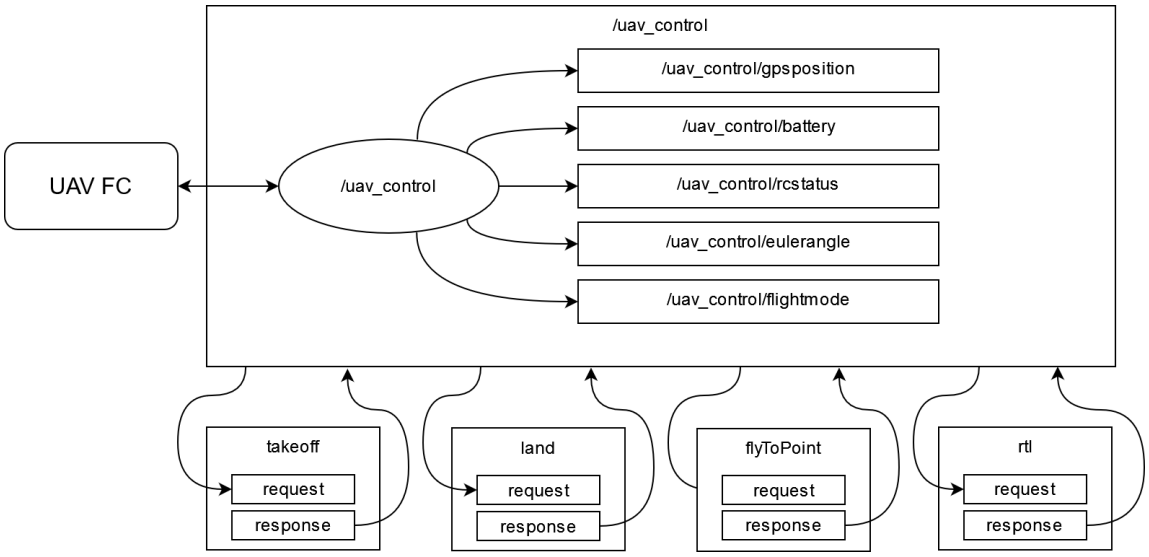
9.3 Komunikácia s letovým kontrolérom

Ku komunikácií s letovým kontrolérom bola pri prvotných testoch použitá knižnica **Mission__flier**, ktorá implementuje integráciu **MAVSDK C++** do ROS-u. Pomocou tejto knižnice je možné načítavať dáta ako napr. stav batérie, letový režim, pozíciu GPS, informácie o GPS a mnoho iných ďalších informácii. Následne boli funkcie pre načítavanie telemetrie implementované do vlastnej knižnice **drone__control**.

V riadiacom uzle **uav__control** je implementované riadenie a načítavanie telemetrie z bezpilotného prostriedku. Uzol poskytuje rôzne typy správ, ktoré možno vidieť na obrázku 9.3. V týchto poskytovaných správach sú zahrnuté informácie o polohe a dostupnosti GPS v správe **gpsposition**. Ďalšie správy **battery**, **rcstatus** a **flightmode** poskytujú informácie o stave batérie, spojení s RC ovládačom a aktuálnom letovom režime. Všetky implementované správy sú súčasťou knižnice **visual__interfaces**. Obsah jednotlivých správ je v prílohe D.

Súčasťou riadiaceho uzlu sú servisy pomocou ktorých je možné základné ovládanie bezpilotného prostriedku. Medzi základné funkcie patrí servis pre vzlet **takeoff**

a servis pre pristátie **land**. Najdôležitejším je servis pre let na bod v priestore **flyToPoint** pomocou, ktorého je možné riadiť bezpilotný prostriedok na požadovaný bod v priestore pomocou vizuálno-inerciálnej odometrie. Uzol má odber na danú tému, ktorá poskytuje informáciu o polohe v priestore typu **nav_msgs::msg::Path**, pomocou ktorej je realizované riadenie na bod v priestore. Posledným implementovaným servisom je návrat na štart **rtl**, v tomto prípade sa jedná o návrat na počiatok súradnicového systému kde bola inicializovaná vizuálna odometria.



Obr. 9.3: Riadiaci uzol uav_control

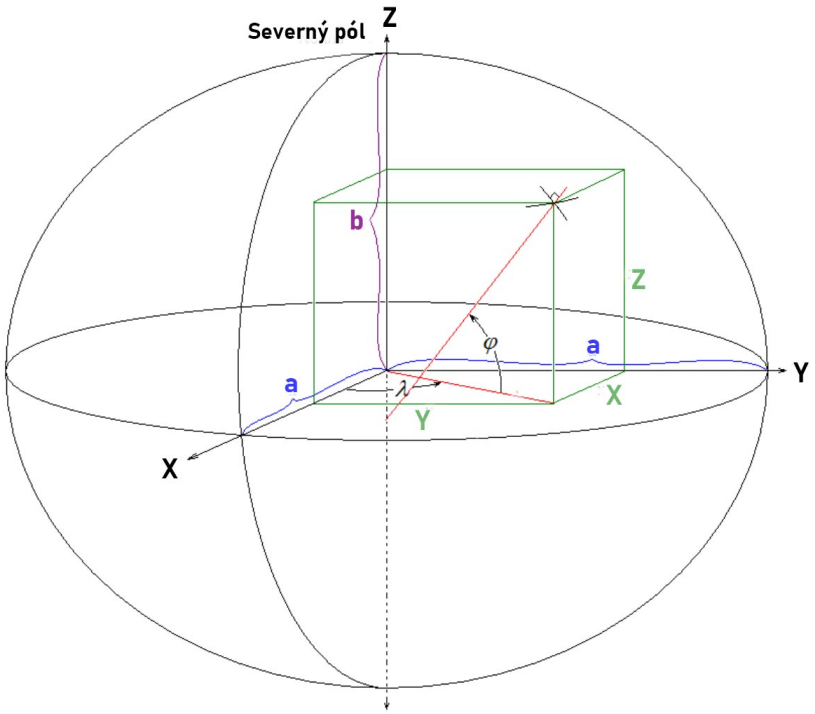
10 Porovnanie s GPS

V tejto kapitole sa budeme venovať porovnaniu presnosti vizuálnej odometrie s lokalizačným systémom GPS.

10.1 Prepočet GPS súradníc na relatívne

Nakoľko výstupom vizuálnej odometrie je relatívna poloha v priestore, je potrebné geodetické súradnice prepočítať do relatívneho súradného systému. Štandardné GPS súradnice sú udávané pomocou *latitude* (zemepisnej šírky), *longtitude* (zemepisnej dĺžky) a *altitude* (výšky). Tento formát má označenie **WGS84** (World Geodetic System 1984). Aby sme mohli určovať relatívne súradnice bezpilotného prostriedku, je potrebné geodetické súradnice prepočítať do geocentrických, ktoré môžme následne použiť ako relatívne.

ECEF (Earth-Centered, Earth-Fixed) je geocentrický systém založený na stredovom bode Zeme. Počiatok súradnicového systému je v strede Zeme, osi smerujú na zemepisný rovník a osa z smeruje na severný pól. Na obrázku 10.1 je znázornený



Obr. 10.1: Súradnicový systém ECEF[32]

súradnicový systém ECEF, kde ρ je zemepisná šírka, λ je zemepisná dĺžka, **a** je

polomer zeme v horizontálnej osi, **b** je polomer zeme v vertikálnej osi. Po prepočte do tohto súradnicového systému je možné jednoducho pracovať s polohou **x,y,z**.[32]

10.1.1 Vzťahy pre prepočet WGS84 na ECEF

Geodetické súradnice WGS84 je možné prepočítať na ECEF súradnice pomocou nasledujúcich vzťahov:

$$x = (N(\phi) + h)\cos(\phi)\cos(\lambda) \tag{10.1}$$

$$y = (N(\phi) + h)\cos(\phi)\sin(\lambda) \tag{10.2}$$

$$z = \left(\frac{a^2}{b^2}N(\phi) + h\right)\sin(\phi) \tag{10.3}$$

$$N(\phi) = \frac{a^2}{\sqrt{a^2\cos^2(\phi) + b^2\sin^2(\phi)}} \tag{10.4}$$

kde:

- a: horizontálny polomer elipsoidu Zeme,
- b: vertikálny polomer elipsoidu Zeme,
- ϕ: zemepisná šírka,
- λ: zemepisná dĺžka,
- h: výška nad elipsoidom,
- N: polomer zakrivenia v hlavnej vertikále.

Následne je možné vypočítať relatívnu polohu v priestore od prvého bodu pomocou týchto vzťahov:

$$\Delta x = x - x_{\text{ref}} \tag{10.5}$$

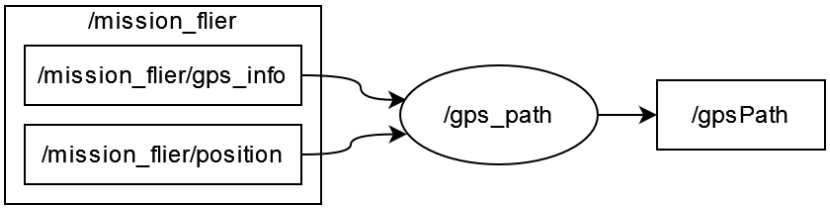
$$\Delta y = y - y_{\text{ref}} \tag{10.6}$$

$$\Delta z = z - z_{\text{ref}} \tag{10.7}$$

Výsledkom je relatívna poloha v priestore oproti referenčným súradniciam.[32]

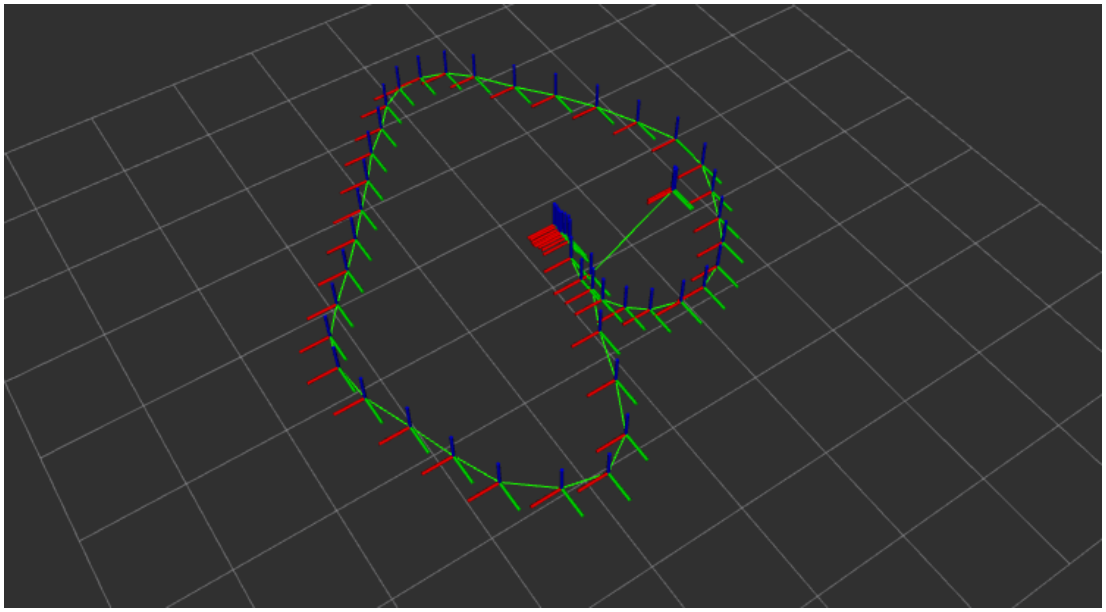
10.2 Implementácia prepočtu

Pre prepočet súradníc bol vytvorený ROS node, ktorý získava GPS dáta z letového kontroléru dronu a následne prepočítava GPS súradnice na relatívne súradnice. Zároveň umožňuje vizualizáciu trajektórie v prostredí rviz. Na obrázku 10.2 je znázornené prepojenie tém pre ROS node **/gps_path**. Node odoberá témy **/mission_flier/position** a **/mission_flier/gps_info** z nodu **/mission_flier**. Následne sú GPS súradnice prepočítané pomocou vzťahov v kapitole 10.1.1. Výsledné súradnice sú zdieľané na tému **gpsPath**. Táto správa je typu **nav_msgs/msg/Path**,



Obr. 10.2: Prepojenie tém k ROS node na prepočet súradníc

ktorú je možno vizualizovať pomocou prostredia rviz. Na obrázku 10.3 je znázornená vizualizácia trajektórie prepočítaných GPS súradníc pomocou prostredia rviz. Pre názornosť bol zvýraznený každý jeden bod trajektórie.



Obr. 10.3: Príklad vizualizácie trajektórie v prostredí rviz

Samotný prepočet medzi medzi WGS84 súradnicami na ECEF súradnice je implementovaný v triede **WGS84ToECEFConverter**. V tejto triede sú zadané potrebné konštanty pre prepočet a funkcia **convertToECEF**, ktorá má návratovú hodnotu **ECEF_Point**. Súčasťou štruktúry **ECEF_Point** sú XYZ súradnice.

10.3 Meranie odchýlky medzi trajektóriami

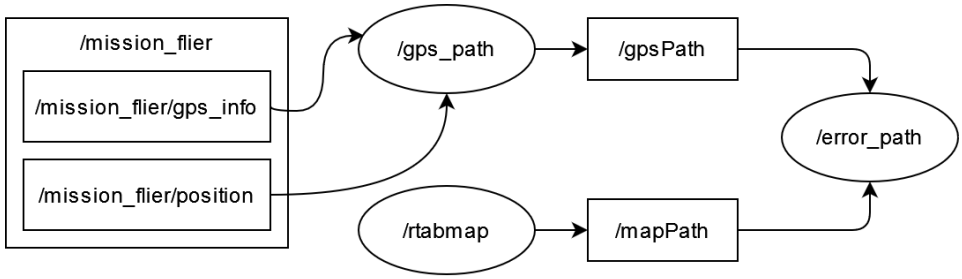
Pre meranie odchýlky medzi vizuálnou odometriou a GPS bol implementovaný ROS node, ktorý odoberá aktuálnu polohu oboch trajektórií. Následne vyhodnocuje rozdiel medzi aktuálnou oboch trajektórií pomocou nasledujúcich vzťahov:

$$\Delta x = x_{\text{gps}} - x_{\text{visual}} \tag{10.8}$$

$$\Delta y = y_{\text{gps}} - y_{\text{visual}} \tag{10.9}$$

$$\Delta z = z_{\text{gps}} - z_{\text{visual}} \tag{10.10}$$

Výsledné údaje je následne možné spracovať a vyhodnotiť napr. pomocou tabuľkového editoru Excel. Na obrázku 10.4 je znázornené prepojenie tém pre výpočet odchýlky medzi jednotlivými trajektóriami.



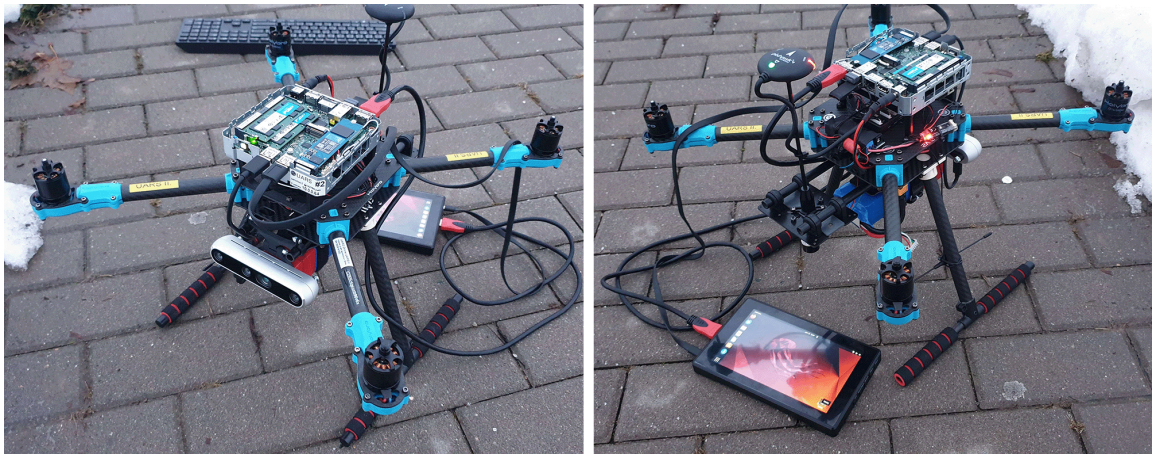
Obr. 10.4: Prepojenie tém k ROS node pre výpočet odchýlky

11 Experiment porovnania presnosti

V nasledujúcej kapitole sú popísané vykonané experimenty porovnania presnosti. S vytvorenými uzlami boli vykonané experimenty pri ktorom boli zaznamenávané všetky údaje na bezpilotnom prostriedku. Počas experimentu bol vytvorený **rosbag**, pomocou ktorého je možné následne rekonštruovať let. Pomocou tohto **rosbagu** je možné neskôr vyvíjať iné rôzne aplikácie s dátami z bezpilotného prostriedku ale napr. taktiež evaluovať namerané dáta.

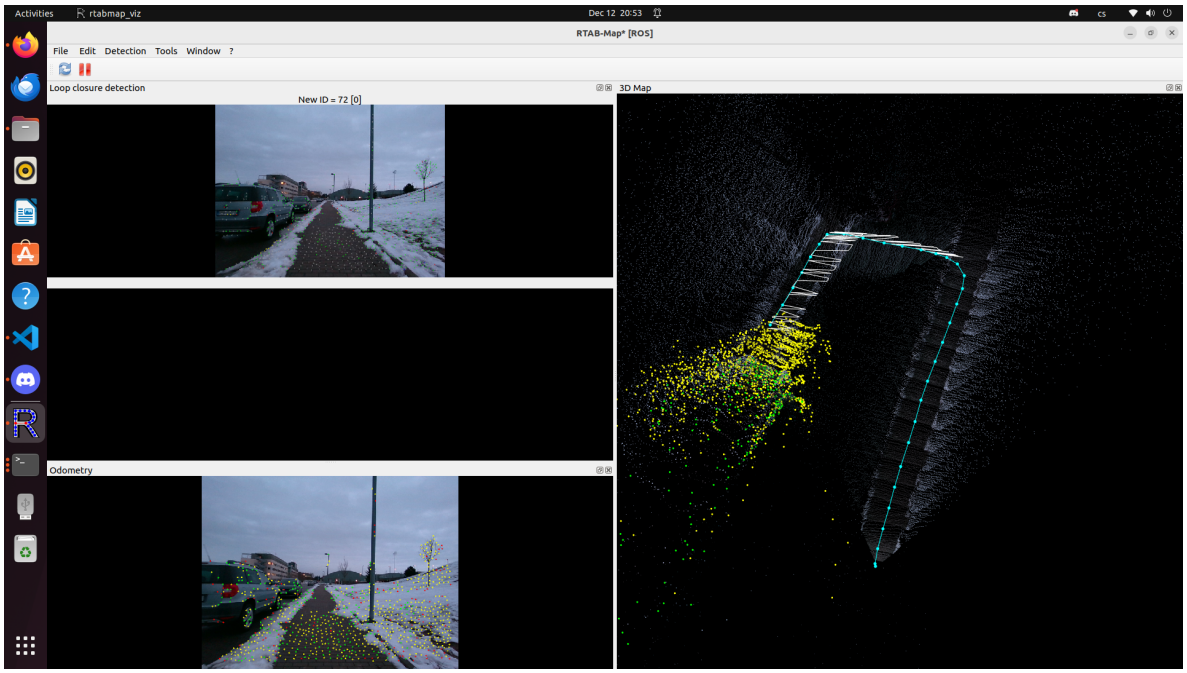
11.1 Priebeh prvého experimentu

Prvý experiment bol vykonaný v priestoroch pri parkoviskách za budovou T12. Po spustení dronu bolo potrebné počkať na dostatočný počet GPS satelitov pre zisťovanie polohy GPS. Následne bolo potrebné spustiť všetky uzly pre komunikáciu s letovou jednotkou, RealSense kamerou a vizuálno-inerciálnou odometriou. Po spustení všetkých potrebných uzlov bolo možné spustiť záznam všetkých vysielaných tém do **rosbag**. Následne bola ručne vykonaná trajektória letu dronu. Na obrázku 11.1 je dron pripravený na experiment. Ako môžeme vidieť na obrázku dron je vybavený



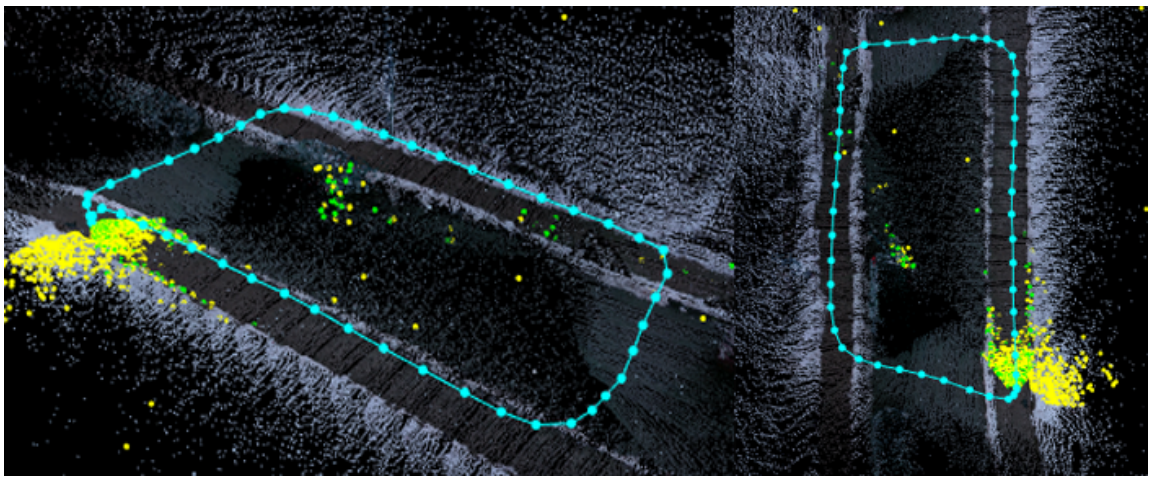
Obr. 11.1: Pripravený drone na experiment

počítačom Intel NUC, letovým kontrolérom a kamerou Intel RealSense D455. Pre experiment bol pripojený taktiež externý monitor pre sledovanie vizualizácie počas experimentu. Na obrázku 11.2 môžeme vidieť snímok obrazovky počas vykonávania experimentu. Počas experimentu bolo možné sledovať priebeh tvorby letovej trajektórie v **rtabmap_viz**. V aplikácií je taktiež možné sledovať aktuálny snímok z kamery s detekovanými príznakmi v obraze, a taktiež párovanie predošlých snímkov pri detekcií *loop closure*.



Obr. 11.2: Priebeh experimentu v prostredí RTAB-Map viz

Na obrázku 11.3 môžeme vidieť dokončenú výslednú trajektóriu. Ako môžeme na obrázku pozorovať bezpilotný prostriedok dosiahol opätovne na miesto odkiaľ bol experiment započatý. Po ukončení experimentu bolo zastavené nahrávanie do **ros-**

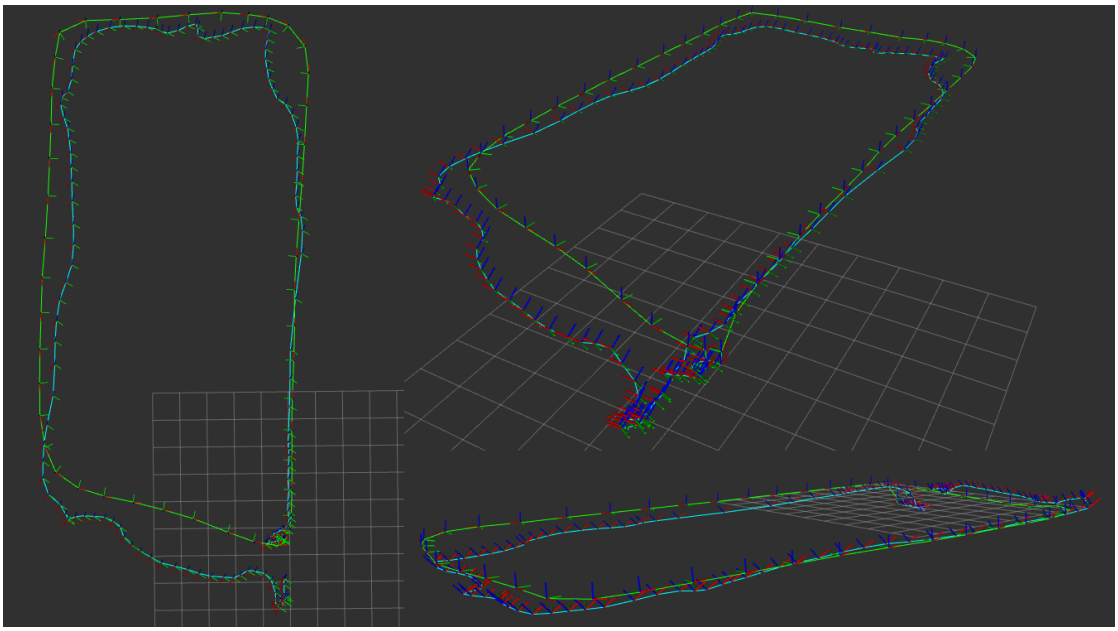


Obr. 11.3: Výsledná trajektória RTAB-Map

bagu. Následne bolo možné namerané dáta kdekoľvek neskôr spracovať. Vytvorený **rosbag** môže byť taktiež použitý pri budúcom vývoji, prípadne testovaní. Výhodou je možnosť pracovania na vývoji bez potreby vykonávania nových meraní.

11.1.1 Vyhodnotenie experimentu

Namerané dáta boli spracované pomocou vytvorených ROS uzlov **/gps_path** a **/error_path**. Výslednú trajektóriu z RTAB-Map-u a prepočítanú z GPS je možné vizualizovať pomocou vizualizačného nástroju **rviz**. V prostredí **rviz** je iba potrebné pridať príslušné témy, prípadne nastaviť štýl zobrazovania. Na obrázku 11.4 je znázornená vizualizácia trajektórií v prostredí **rviz**. Pre predstavu preletenej vzdialenosti predstavuje mriežka v prostredí **rviz** 1m. Modrá trajektória odpovedá polohe

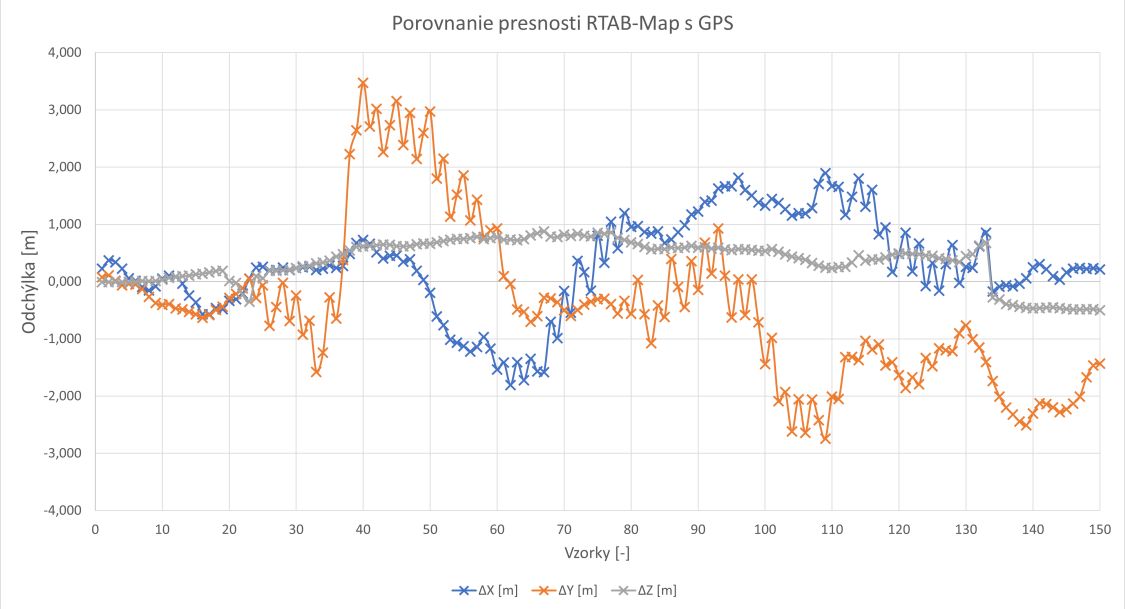


Obr. 11.4: Vizualizované trajektórie v prostredí rviz

zaznamenanej pomocou GPS, zelená trajektória odpovedá trajektórií určenej pomocou vizuálno-inerciálnej odometrie. Ako môžeme vidieť na obrázku trajektórie majú s určitými odchýlkami približne rovnaký tvar. Avšak v trajektórií GPS sa vyskytujú rôzne oscilácie. Pri experimente mal GPS systém k dispozícií približne 8 satelitov a nebola použitá GPS s RTK (Real Time Kinematic). Pri budúcich experimentoch by bolo vhodné trajektórie referencovať k nejakým referenčným bodom v teréne. Avšak cieľom tohto experimentu bolo vyhodnotiť odchýlku medzi GPS na bezpilotnom prostriedku s vizuálno inerciálnou odometriou.

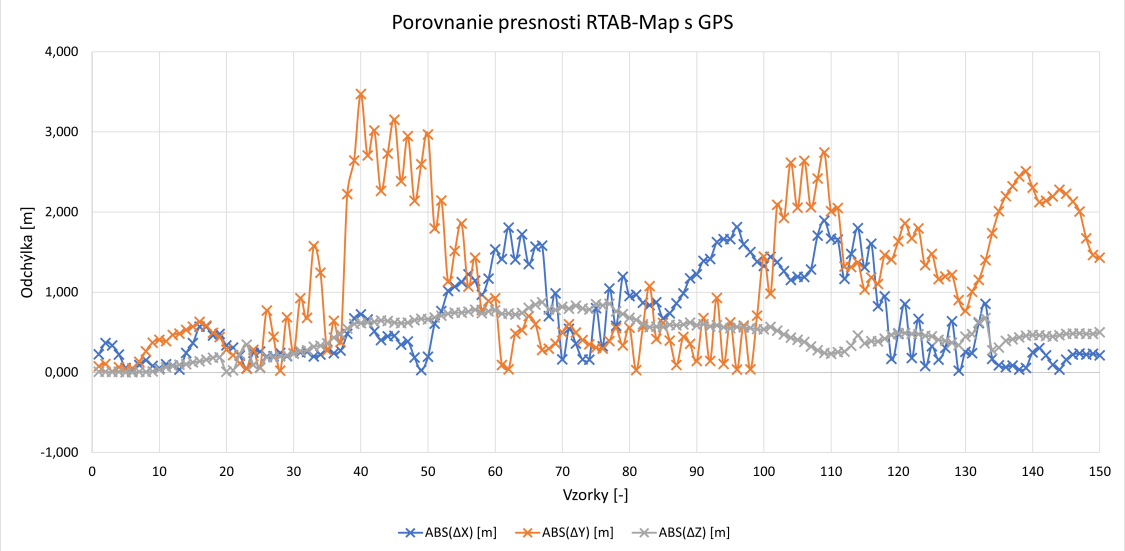
Výstupné dáta boli následne spracované pomocou tabuľkového editoru Excel. Tabuľka s nameranými hodnotami je v prílohe E. Bola vyhodnotená odchýlka vo všetkých osiach, absolútna chyba a taktiež euklidovská vzdialenosť medzi jednotlivými bodmi trajektórie. V nasledujúcich grafoch sú znázornené priebehy odchýlok medzi GPS a vizuálno-inerciálnou odometriou.

V grafe 11.5 je znázornená odchýlka v jednotlivých osiach. Priemerná odchýlka v ose **x** bola 0,294m, v ose **y** bola -0,431m a v ose **z** 0,357m. Preletená trajektória predstavovala približne 50m.



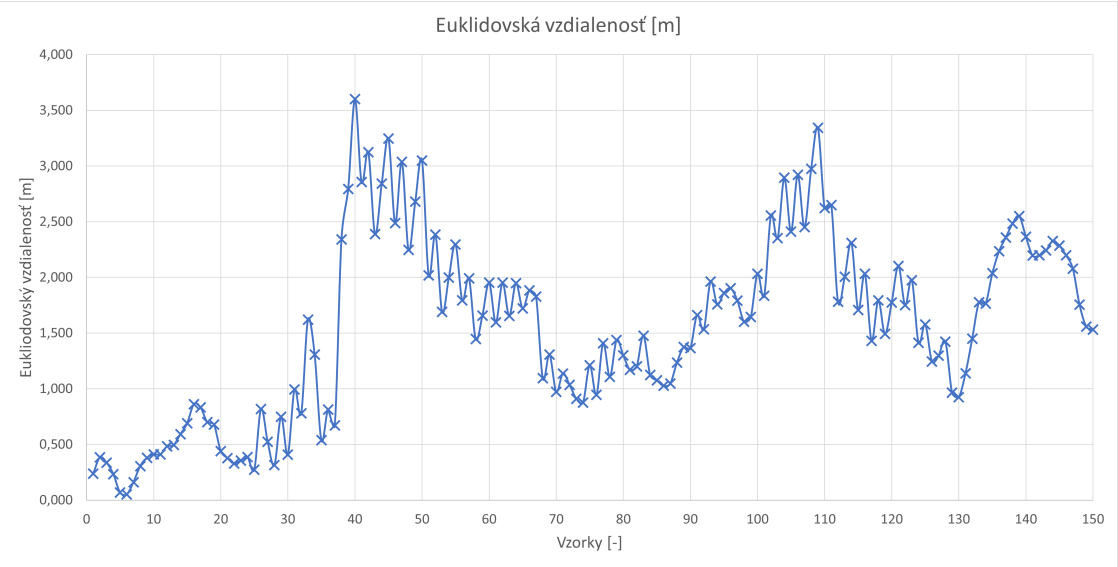
Obr. 11.5: Graf porovnania odchýlky presnosti v jednotlivých osiach

V grafe 11.6 je znázornená absolútna odchýlka v jednotlivých osiach. Priemerná absolútna odchýlka v ose **x** bola 0,657m, v ose **y** bola 1,127m a v ose z 0,465m.



Obr. 11.6: Graf porovnania absolútnej odchýlky presnosti v jednotlivých osiach

V grafe 11.7 je znázornená odchýlka pomocou euklidovskej vzdialenosti medzi polohou určenou vizuálnou odometriou a GNSS. Priemerná odchýlka bola 1,558m.



Obr. 11.7: Graf porovnania euklidovskej vzdialenosti

Histogram chýb v jednotlivých osiach je v prílohe E.1. Na základe experimentu môžeme konštatovať, že vizuálno-inerciálna odometria dosahuje veľmi dobrých výsledkov v porovnaní s GPS. Ako už bolo spomenuté vyššie v ďalších experimentoch by bolo vhodné obidve merané trajektórie referencovať k referenčným bodom v teréne.

11.1.2 Klimatické podmienky

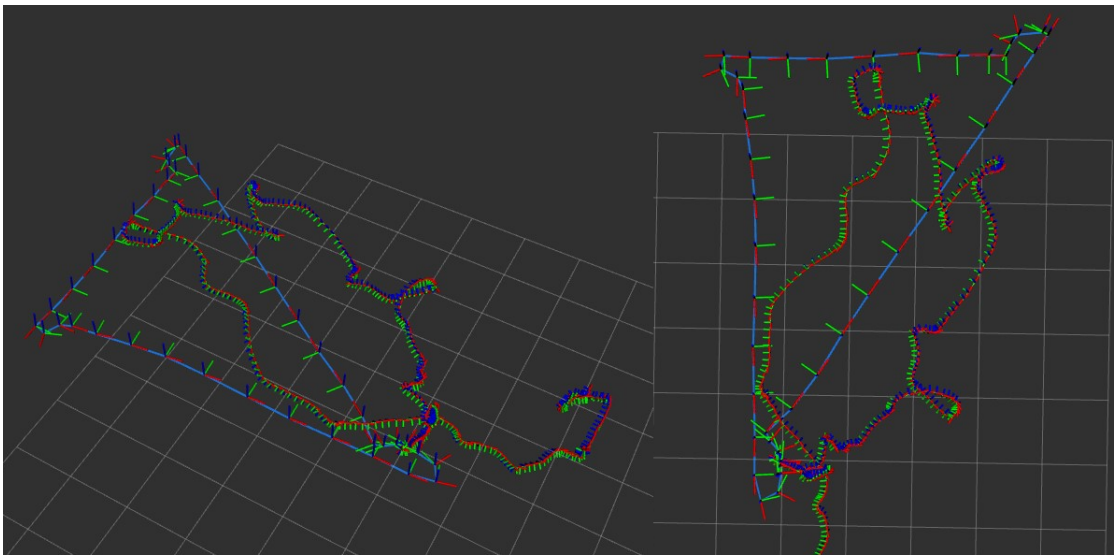
Počas vykonávania experimentu bola vonkajšia teplota približne 4°C. Vlhkosť nebola počas experimentu zaznamenaná. Terén bol počas experimentu s výnimkou ciest pokrytý súvislou vrstvou snehu. Počas experimentu bola vysoká oblačnosť.

11.2 Ďalšie merania

V nasledujúcej podkapitole sú popísané výsledky ďalších meraní presnosti oproti GPS. Opakované merania boli vykonávané rovnakým spôsobom ako pri prvom experimente. Lety boli zaznamenané pomocou **rosbagov** a následne boli vyhodnotené pomocou implementovaných uzlov pre vyhodnotenie chyby.

11.2.1 Prvé meranie

Počas týchto meraní bol cieľ tvoriť trajektóriou letu určité geometrické tvary aby bolo možné následne evaluovať presnosť. Ako môžeme vidieť na obrázku 11.8 z trajektórie vizuálnej odometrie (modrá) vidieť jasne trojuholníkový tvar. Na trajektórii GNSS môžeme vidieť výrazné odchýlky oproti vizuálnej odometrii. Ako môžeme vi-

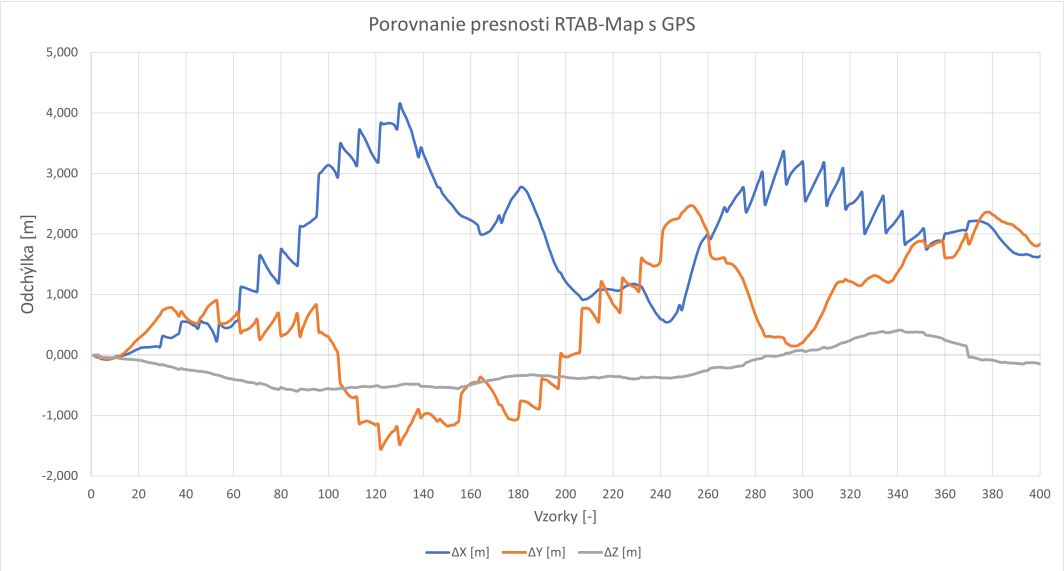


Obr. 11.8: Vizualizované trajektórie v prostredí rviz

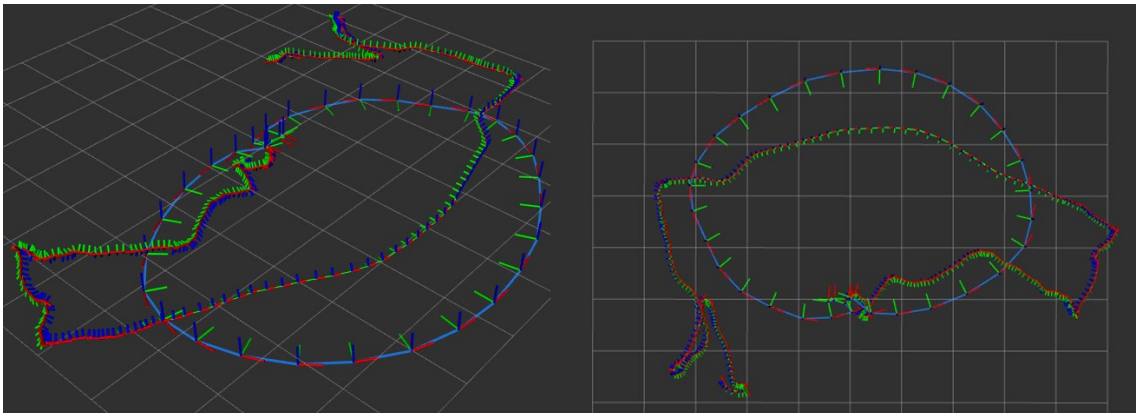
dieť na obrázku 11.9 maximálna odchýlka bola zaznamenaná približne 4 metre na osi x. Priemerná absolútna chyba na osi x bola 1,679 metra, na osi y 0,638 metra a na osi z 0,385 metra. Samotná preletená trajektória bola približne 19m. Počas merania boli k dispozícii 7 až 10 satelitov. Ďalšie grafy sú súčasťou prílohy F.1.

11.2.2 Druhé meranie

Ako môžeme vidieť na obrázku 11.10 z trajektórie vizuálnej odometrie (modrá) vidieť jasne elipsovitý tvar. Tak ako aj pri prvom meraní na trajektórii GNSS môžeme vidieť výrazné odchýlky oproti vizuálnej odometrii. Ako môžeme vidieť na obrázku 11.11 maximálna odchýlka bola zaznamenaná približne 4,5 metra na osi x. Priemerná absolútna chyba na osi x bola 0,698 metra, na osi y 1,160 metra a na osi z 0,311 metra. Preletená trajektória predstavovala približne 17m. Počas merania boli k dispozícii 8 až 10 satelitov. Ďalšie grafy sú súčasťou prílohy F.2.



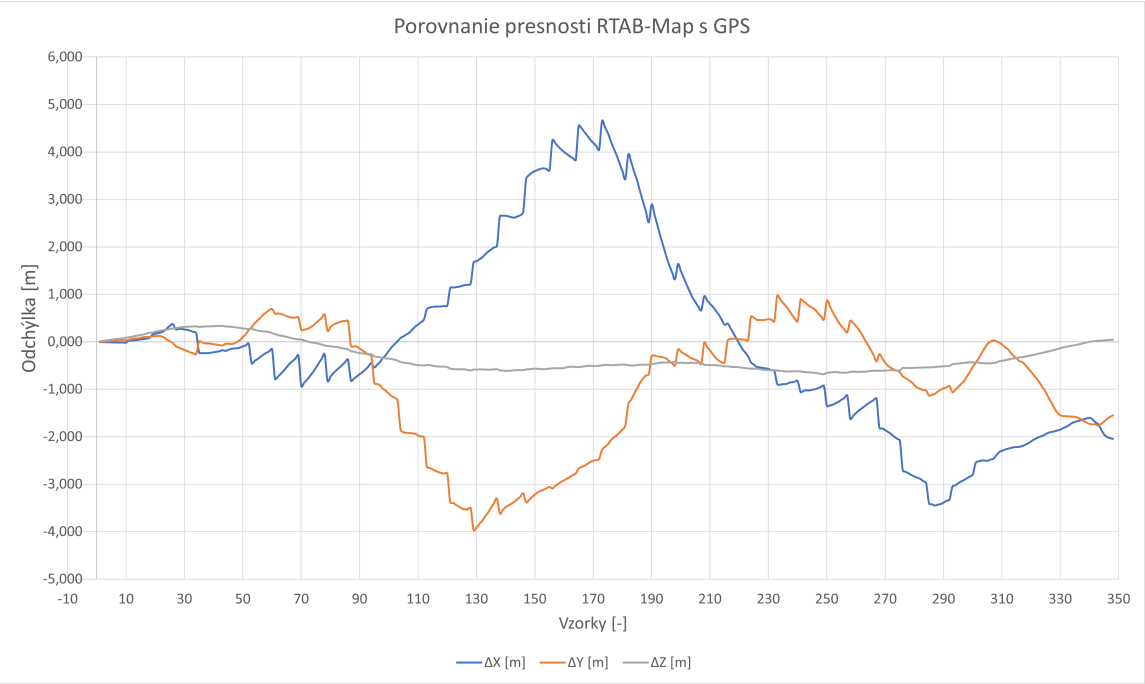
Obr. 11.9: Graf porovnania odchýlky presnosti v jednotlivých osiach



Obr. 11.10: Vizualizované trajektórie v prostredí rviz

11.2.3 Výsledky merania

Druhý experiment potvrdil výsledky dosiahnuté počas prvého experimentu. Vizuálno-inerciálna odometria dosahuje veľmi dobré výsledky v porovnaní s globálnym pozičným systémom GNSS. Počas druhého experimentu nastal problém s hĺbkovou kamerou, nakoľko bolo počas experimentu slnečno, bol obraz z hĺbkovej kamery preexponovaný. Tento fakt spôsobil prakticky nefunkčnosť hĺbkovej kamery. Avšak problém bol vyriešený zmenou nastavenia expozície hĺbkovej kamery. Značné nepresnosti v trajektórií GNSS boli spôsobené nízkym počtom satelitov pre určenie polohy, nakoľko bolo meranie vykonávané v zastavanej oblasti. Tento experiment potvrdil značnú výhodu vizuálnej odometrie oproti GNSS v zastavaných oblastiach,



Obr. 11.11: Graf porovnania odchýlky presnosti v jednotlivých osiach

nakoľko nie je závislá od žiadneho zdroju signálu.

11.2.4 Klimatické podmienky

Počas vykonávania experimentu bola vonkajšia teplota približne 20°C. Vlhkosť nebola počas experimentu zaznamenaná. Terén tvorila prevažne trávnatá plocha s budovami na pozadí v zastavanej oblasti. Počas experimentu bolo slnečno s vyjasnenou oblohou.

Záver

V rámci tejto diplomovej práce sme sa zoznámili s dostupnými technológiami a metódami pre určovanie polohy bezpilotného prostriedku za pomoci vizuálnych a inerciálnych dát. Boli zhodnotené výhody, nevýhody jednotlivých metód a podmienky ich použitia. V úvode práce sú popísané rôzne typy kamerových senzorov, ich parametre a možnosti použitia v spojení s vizuálnou odometriou.

V ďalšej časti práce je popísaný systém ROS, jeho rôzne distribúcie, verzie, súčasti a spôsob komunikácie. Na základe použitého operačného systému bola zvolená vhodná distribúcia ROS Humble. Ďalšia časť práce je venovaná výberu vhodného palubného počítaču a hĺbkovej kamery. Vzhľadom na rozmery a potrebný výpočtový výkon je najvhodnejšie použiť počítač Intel NUC. Vzhľadom na súčasnú dostupnosť hĺbkových kamier Realsense bol zvolený dostupný model D455.

Nasledujúca časť práce je venovaná inštalácií operačného systému, potrebných ovládačov pre kameru a knižniciam do ROSu pre komunikáciu s kamerou Realsense. K vyčítavaniu údajov z kamery bola použitá knižnica pre ROS od firmy Intel.

Pre spracovanie obrazových a inerciálnych dát bol zvolený algoritmus RTAB-Map. V práci sú popísané detekčné algoritmy príznakov a spôsob určovania polohy, ktoré algoritmus využíva. Taktiež je popísané prepojenie jednotlivých uzlov v rámci systému ROS.

Nasledujúca časť práce bola venovaná ovládaniu bezpilotného prostriedku. Boli diskutované rôzne typy letových režimov a ich vhodnosť v spojení s vizuálnou odometriou. Ďalej boli opísané možnosti riadenia bezpilotného prostriedku v režime offboard. Taktiež je popísaná implementácia navrhnutého ROS node, implementované správy pre telemetriu a implementované servisy. Funkčnosť implementovaného uzlu bola následne otestovaná v simulačnom prostredí Gazebo v spojení s PX4 SITL.

Ďalšia časť práce je venovaná popisu prepojenia jednotlivých použitých komponentov. Je popísané použité rozhranie pre pripojenie kamery a letového kontroléru k palubnému počítaču. Zároveň je popísané prepojenie implementovaných uzlov v rámci systému ROS.

V nasledujúcej časti práce bolo vykonané porovnanie presnosti určovania polohy voči systému GPS. V rámci experimentov boli vytvorené rosbagy, pomocou ktorých bolo následne možné vyhodnotiť presnosť, ale taktiež neskôr umožňujú vyvíjať rôzne iné aplikácie so zaznamenanými dátami. Na prepočet GPS súradníc na relatívne bol vytvorený ROS node /gps_path. K meraniu odchýlky medzi trajektóriami bol implementovaný ROS node /error_path.

Na základe vykonaných experimentov je možné konštatovať, že vizuálno-inerciálna odometria dosahuje veľmi dobré výsledky v porovnaní s GNSS. Pri prvom experimente bola maximálna absolútna odchýlka v osi y približne 3,5m, avšak priemerná

odchýlka bola 1,127m. V osi x bola priemerná absolútna odchýlka 0,657m a v osi z 0,465m. Počas ďalších experimentov nepresiahla maximálna absolútna odchýlka vo všetkých osiach hodnotu 5m, avšak priemerná odchýlka nepresiahla hodnotu 1,7m. Ďalšie experimenty potvrdili vhodnosť použitia vizuálno-inerciálnej odometrie hlavne v zastavaných oblastiach. V budúcnosti by bolo vhodné vykonať meranie voči pevným referenčným bodom v teréne, nakoľko vykonané meranie vyjadruje presnosť voči použitej GPS.

Literatúra

[1] Visual-odometry. *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/topics/computer-science/visual-odometry>. [cit. 2023-09-25]

[2] N., Poponak, Gettinger D., Carr S.S. KEANE J.F., et al. A review on absolute visual localization for UAV. In: *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0921889020305066>. [cit. 2023-09-29].

[3] *Inertial Navigation System*. Online. In: ScienceDirect. Dostupné z: <https://www.sciencedirect.com/topics/engineering/inertial-navigation-system>. [cit. 2023-09-29].

[4] EL-RABBANI, A., M.M. Mitchell C.E. EWING, L. Weill M.S. GREWAL, et al. Chapter 1 - Introduction to Navigation. In: *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/B9780127999494000014>. [cit. 2023-10-03].

[5] TAHERI, Hamid a Zhao Chun XIA. SLAM; definition and evolution. *Engineering Applications of Artificial Intelligence* [online]. Dostupné z : <https://www.sciencedirect.com/science/article/pii/S0952197620303092>. [cit. 2023-09-29].

[6] Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges. In: *MDPI* [online]. Dostupné z : <https://www.mdpi.com/2504-446X/7/2/89>. [cit. 2023-10-04].

[7] ABDELMOGHIT ZAARANE, Ibtissam Slimani, Abdelmoghit Zaarane IBTISSAM SLIMANI, Yuslena Sari PUGUH BUDI PRAKOSO, et al. Distance measurement system for autonomous vehicles using stereo camera. In: *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S2590005620300011>. [cit. 2023-10-04].

[8] Li Z. BIAN J.-W., Elvira R. CAMPOS C., et al. Robust self-supervised monocular visual odometry based on prediction-update pose estimation network. In: *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0952197622004717>. [cit. 2023-10-04].

[9] K. MIKOLAJCZYK, C. Schmid., R. Mohr C. SCHMID, T. Tuytelaars K. MIKOLAJCZYK, et al. Performance evaluation of feature detection and matching in stereo visual odometry. In: *ScienceDirect* [online].

Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0925231213002774>. [cit. 2023-10-05].

[10] REALSENSE, Intel. Beginner's guide to depth (Updated). In: *Intel® RealSense™ Depth and Tracking Cameras* [online]. Dostupné z: <https://www.intelrealsense.com/beginners-guide-to-depth/>. [cit. 2023-10-05].

[11] BLOSS, Richard, Brian Gerkey MORGAN QUIGLEY, M. Crosby F. ROVIDA, et al. A ROS2 based communication architecture for control in collaborative and intelligent automation systems. In: *ScienceDirect* [online]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S2351978920300469>. [cit. 2023-10-12].

[12] ROS Documentation. [online]. Dostupné z: <https://docs.ros.org/>. [cit. 2023-10-12]

[13] PINTÉR, Marco. Lokalizace a navigace dronu ve známém prostředí bez využití GNSS. Dostupné z: <https://www.vut.cz/studenti/zav-prace/detail/142055>. [cit. 2023-10-12].

[14] Intra-process Communications in ROS 2. [online]. Dostupné z: <https://design.ros2.org/articles/intraprocess_communications.html>. [cit. 2023-10-16]

[15] RASPBERRY PI LTD. *Raspberry Pi 4 Tech Specs*. Online. In: Raspberry Pi. Dostupné z: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>. [cit. 2023-10-18].

[16] *Jetson Nano*. Online. In: NVIDIA Developer. Dostupné z: <https://developer.nvidia.com/embedded/jetson-nano>. [cit. 2023-10-18].

[17] *Intel® NUC Mini PCs, Elements and Laptops*. Online. In: Intel. Dostupné z: <https://www.intel.com/content/www/us/en/products/details/nuc.html>. [cit. 2023-10-24].

[18] *Intel RealSense D400 Series Product Family Datasheet*. Online. In: Intel® RealSense™ Developer Documentation. Dostupné z: <https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet>. [cit. 2023-10-26].

[19] A.S., Alza.*Intel NUC 11 Pro Kit Slim (NUC11TNKi7)*. Online. In: Alza.sk. Dostupné z: <https://www.alza.sk/intel-nuc-11-pro-kit-slim-nuc11tnki7-d6318264.htm>.[cit. 2023-10-28].

[20] *IMU: BMI055*. Online. In: Bosch Sensortec. Dostupné z: <https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi085/>.[cit. 2023-10-28].

[21] *Intel® RealSense™ Self-Calibration for D400 Series Depth Cameras*. Online. In: Intel® RealSense™ Developer Documentation. Dostupné z: <https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras>.[cit. 2023-10-31].

[22] *Intel® RealSense™ ROS2 packages for using Intel RealSense D400 cameras*. Online. GitHub. Dostupné z: <https://github.com/IntelRealSense/realsense-ros>.[cit. 2023-11-05].

[23] *RTAB-Map*. Online. RTAB-Map. Dostupné z: <http://introlab.github.io/rtabmap/>.[cit. 2023-11-07].

[24] LABBÉ, Mathieu a MICHAUD, François. *Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation*. Online. IEEE Xplore. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/6459608>.[cit. 2023-11-07].

[25] S. ASADZADEH, W.J. de Oliveira; M.F. ASLAN, A. Durdu; D. BARATH, J. Matas; D. BARATH, J. Matas; BATURU, C. et al. *Unmanned aerial vehicle remote sensing image registration based on an improved oriented FAST and rotated BRIEF- random sample consensus algorithm*. Online. ScienceDirect. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S0952197623011284>.[cit. 2023-11-08].

[26] ROSTEN, Edward; PORTER, Reid a DRUMMOND, Tom. *Faster and Better: A Machine Learning Approach to Corner Detection*. Online. IEEE Xplore. Dostupné z: <https://ieeexplore.ieee.org/document/4674368>.[cit. 2023-11-08].

[27] MA, Chaoqun; HU, Xiaoguang; XIAO, Jin; DU, Huanchao a ZHANG, Geofeng. *Improved ORB Algorithm Using Three-Patch Method and Local Gray Difference*. Online. MDPI. Dostupné z: <https://www.mdpi.com/1424-8220/20/4/975>.[cit. 2023-11-09].

[28] BAY Herbert, ESS Andreas, TUYTELAARS Tinne, VAN GOOL Luc. *Speeded-Up Robust Features (SURF)*. Online. ScienceDirect. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S1077314207001555>.[cit. 2023-11-09].

[29] OpenCV. *Introduction to SURF (Speeded-Up Robust Features)*. Online. ScienceDirect. Dostupné z: <https://docs.opencv.org/3.4/df/dd2/tutorial_py_surf_intro.html>.[cit. 2023-11-10].

[30] *Introduction to Loop Closure Detection in SLAM*. Online. Dostupné z: <https://www.thinkautonomous.ai/blog/loop-closure/>.[cit. 2023-11-10].

[31] CHANDRACHARY, Shiva.*Introduction to 3D SLAM with RTAB-Map*. Online. Dostupné z: <https://shivachandrachary.medium.com/introduction-to-3d-slam-with-rtab-map-8df39da2d293>.[cit. 2023-11-13].

[32] *Geographic coordinate conversion*. Online. In: Wikipedia, The Free Encyclopedia. Dostupné z: <https://en.wikipedia.org/wiki/Geographic_coordinate_conversion>.[cit. 2023-11-18].

[33] *PX4 Autopilot User Guide*. Online. In: PX4 Autopilot User Guide. Dostupné z: <https://docs.px4.io/main/en/>.[cit. 2024-02-24].

[34] *Local tangent plane coordinates*. Online. In: Wikipedia, The Free Encyclopedia. Dostupné z: <https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates>.[cit. 2024-02-25].

[35] *MAVLink protocol*. Online. In: Mavlink. Dostupné z: <https://mavlink.io/en/about/overview.html>.[cit. 2024-02-25].

[36] *MAVSDK C++ library documentation*. Online. In: Mavsdk. Dostupné z: <https://mavsdk.mavlink.io/main/en/cpp/api_reference>.[cit. 2024-03-08].

[37] *PX4 simulation*. Online. In: docs.px4 . Dostupné z: <https://docs.px4.io/main/en/simulation/>.[cit. 2024-03-26].

[38] ROS Documentation: Executors. [online]. Dostupné z: <https://docs.ros.org/en/foxy/Concepts/About-Executors.html>. [cit. 2023-10-12]

Zoznam symbolov a skratiek

VO	Visual Odometry
GPS	Global Position System
GNSS	Global navigation satellite system
RVL	Relative visual localization
AVL	Absolute visual localization
UAV	Unmanned aerial vehicle
SLAM	Simultaneous localization and mapping
V-SLAM	Visual Simultaneous Localization and Mapping
IMU	Inertial Measurement Unit
IMS	Inertial Navigation System)
ToF	Time of Flight
DDS	Data Distibution Service
RMW	ROS Middleware Inteface
QoS	Quality of Service
SDK	Software Development Kit
ROS	Robot Operating System
RTAB-Map	Real-Time Appearance-Based Mapping
BoW	Bag of Words
RTK	Real Time Kinematic

Zoznam príloh

A Obsah priloženého média **87**
A.1 ros2_ws . . . 87
A.2 namerane_hodnoty . . . 87

B Inštalácia **88**
B.1 Ubuntu 22.04 . . . 88
B.2 ROS Humble . . . 88
B.3 RealSense SDK 2.0 . . . 88
B.3.1 Inštalácia dalších potrebných ovládačov . . . 89
B.4 RealSense ROS wrapper . . . 89
B.4.1 Príklady spustenia . . . 89
B.5 Zoznam topicov z Realsense wrapperu . . . 89
B.6 RTAB-Map . . . 90
B.6.1 Prílad spustenia . . . 90
B.6.2 Príklad spustenia s vlastným launch file . . . 90

C Simulácia **91**
C.1 Spustenie Gazebo . . . 91
C.2 Spustenie PX4-SITL . . . 91
C.3 Spustenie simulovaných pozičných dát . . . 91
C.4 Spustenie riadiaceho uzlu . . . 91

D Implementované ROS správy a servisy **92**
D.1 Implementované ROS správy . . . 92
D.1.1 Battery.msg . . . 92
D.1.2 Eulerangle.msg . . . 92
D.1.3 Flightmode.msg . . . 92
D.1.4 Gpsposition.msg . . . 92
D.1.5 Rcstatus.msg . . . 92
D.2 Implementované servisy . . . 93
D.2.1 Position.srv . . . 93
D.3 Použité servisy . . . 93
D.3.1 std_srvs\Trigger.srv . . . 93

E Namerané dáta z prvého experimentu **94**
E.1 Histogram chyby v jednotlivých osiach . . . 98

F Namerané dáta z ďalších experimentov **100**
F.1 Prvý experiment 100
F.2 Druhy experiment 101

A Obsah priloženého média

A.1 ros2_ws

Obsahom tejto zložky sú všetky implementované ROS 2 knižnice.

A.2 namerane_hodnoty

V priloženom excel súbore sú všetky namerané hodnoty a grafy s vykonaných experimentov.

```
.....................................................koreňový adresár priloženého archívu
├── ros2_ws ..................................... pracovná zložka pre ROS
│   └── src
│       ├── drone_control
│       ├── sim_position
│       ├── visual_interfaces
│       └── GPS_Path
├── pdf ..................................... pdf diplomovej práce
│   └── diplomovaPraca.pdf
├── namerane_hodnoty
│   └── experimenty_namerane_hodnoty.xlsx
└── zdrojoveSuboryLatexPrace ..... LaTex repozitár diplomovej práce
```

B Inštalácia

B.1 Ubuntu 22.04

Inštalačný obraz je možné stiahnuť pomocou nasledujúceho odkazu.

`https://ubuntu.com/download/desktop/thank-you?version=22.04.3&architecture=amd64`

Bootovateľné USB je následne možné vytvoriž pomocou programu Rufus

`https://rufus.ie/en/`

B.2 ROS Humble

```
sudo apt update
sudo apt upgrade
sudo apt install -y curl gnupg2 lsb-release

sudo sh -c 'echo "deb http://packages.ros.org/ros2/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros2.list'

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/
ros.asc | sudo apt-key add -

sudo apt update
sudo apt install -y ros-humble-desktop

source /opt/ros/humble/setup.bash

sudo apt install -y python3-rosdep
sudo rosdep init
rosdep update

sudo apt install python3-colcon-common-extensions
```

B.3 RealSense SDK 2.0

Inštalácia pomocou predkompilovaných binárie z ROS serverov

```
sudo apt install ros-humble-librealsense2*
```

B.3.1 Inštalácia dalších potrebných ovládačov

```
sudo mkdir -p /etc/apt/keyrings

curl -sSf https://librealsense.intel.com/Debian/librealsense.pgp
| sudo tee /etc/apt/keyrings/librealsense.pgp > /dev/null

echo "deb [signed-by=/etc/apt/keyrings/librealsense.pgp]
https://librealsense.intel.com/Debian/apt-repo ‘lsb_release -cs‘ main" | \

sudo tee /etc/apt/sources.list.d/librealsense.list
sudo apt-get update

sudo apt-get install librealsense2-dkms

sudo apt-get install librealsense2-utils
```

Pri inštalácií je potrebné nastaviť Secure Boot heslo a reštartovať počítač následne zadať heslo. Je to potrebné z dôvodu inštalácie ovládačov.

B.4 RealSense ROS wrapper

Inštalácia pomocou predkompilovaných binárií z ROS serverov

```
sudo apt install ros-humble-realsense2-*
```

B.4.1 Príklady spustenia

```
príklad spustenia pre získavanie obrazu:
ros2 launch realsense2_camera rs_launch.py align_depth.enable:=true

príklad spustenia pre získavanie obrazu a IMU:
ros2 launch realsense2_camera rs_launch.py align_depth.enable:=true
enable_gyro:=true enable_accel:=true unite_imu_method:=2
```

B.5 Zoznam topicov z Realsense wrapperu

```
/camera/camera/accel/imu_info
/camera/camera/accel/metadata
/camera/camera/accel/sample
```

```
/camera/camera/aligned_depth_to_color/camera_info
/camera/camera/aligned_depth_to_color/image_raw
/camera/camera/color/camera_info
/camera/camera/color/image_raw
/camera/camera/color/metadata
/camera/camera/depth/camera_info
/camera/camera/depth/image_rect_raw
/camera/camera/depth/metadata
/camera/camera/extrinsics/depth_to_accel
/camera/camera/extrinsics/depth_to_color
/camera/camera/extrinsics/depth_to_depth
/camera/camera/extrinsics/depth_to_gyro
/camera/camera/gyro/imu_info
/camera/camera/gyro/metadata
/camera/camera/gyro/sample
/camera/camera/imu
```

B.6 RTAB-Map

```
Inštalácia pomocou predkompilovaných binárok z ROS serverov
sudo apt install ros-humble-rtabmap-ros
```

B.6.1 Prílad spustenia

```
prípadne je možné spustiť pomocou vstavaných spúštacích súbrov
ros2 launch rtabmap_launch realsense_d435i_color.launch.py
```

B.6.2 Príklad spustenia s vlastným launch file

```
pre spustenie je potrebné mať nainštalovaný balíček z príloh
ros2 launch rtabmap_launch_files realsense_d455_color_imu.launch.py
```

C Simulácia

C.1 Spustenie Gazebo

```
ros2 launch gazebo_sim x500_depthlaunch_withoutgz.launch.py
```

C.2 Spustenie PX4-SITL

```
PX4_SYS_AUTOSTART=4001 PX4_SIM_MODEL=x500_depth
./build/px4_sitl_default/bin/px4 -i 1
```

C.3 Spustenie simulovaných pozičných dát

```
ros2 run sim_visual sim_visual
```

C.4 Spustenie riadiaceho uzlu

```
ros2 run visual_odometry_control drone_control
```

D Implementované ROS správy a servisy

Všetky imlementované správy a servisy sú súčasťou knižnice **visual_interfaces**.

D.1 Implementované ROS správy

D.1.1 Battery.msg

```
float32 voltage_v
float32 current
float32 remaining_percent
```

D.1.2 Eulerangle.msg

```
float32 roll
float32 pitch
float32 yaw
```

D.1.3 Flightmode.msg

```
int32 flighmode
```

D.1.4 Gpsposition.msg

```
float64 latitude_deg
float64 longitude_deg
float32 absolute_altitude_m
float32 relative_altitude_m
int32 number_of_satellites
int32 fix_type
```

D.1.5 Rcstatus.msg

```
bool was_available_once
bool is_available
float32 signal_strength_percent
```

D.2 Implementované servisy

D.2.1 Position.srv

```
float32 x
float32 y
---
bool success          # Whether the operation was successful
int64 result          # Additional information, e.g., a result value
string message        # Additional message describing the result
```

D.3 Použité servisy

D.3.1 std_srvs\Trigger.srv

```
---
bool success        # indicate successful run of triggered service
string message      # informational, e.g. for error messages
```

E Namerané dáta z prvého experimentu

Tabuľka k nameraným dátam v experimente popísanom v kapitole 11.

Tab. E.1: Tabuľka k nameraným dátam v experimente

Vzor. [-]	ΔX **[m]**	ΔY **[m]**	ΔZ **[m]**	**ABS**(ΔX) **[m]**	**ABS**(ΔY) **[m]**	**ABS**(ΔZ) **[m]**	**Euklid. vzdial. [m]**
1	0,226	0,075	-0,010	0,226	0,075	0,010	0,238
2	0,369	0,107	-0,009	0,369	0,107	0,009	0,384
3	0,336	0,013	-0,005	0,336	0,013	0,005	0,336
4	0,224	-0,066	-0,008	0,224	0,066	0,008	0,233
5	0,055	-0,042	-0,003	0,055	0,042	0,003	0,069
6	0,013	-0,051	-0,002	0,013	0,051	0,002	0,053
7	-0,093	-0,132	0,008	0,093	0,132	0,008	0,162
8	-0,154	-0,264	0,007	0,154	0,264	0,007	0,306
9	-0,083	-0,372	0,011	0,083	0,372	0,011	0,381
10	0,052	-0,408	0,035	0,052	0,408	0,035	0,412
11	0,104	-0,392	0,065	0,104	0,392	0,065	0,410
12	0,085	-0,469	0,079	0,085	0,469	0,079	0,483
13	-0,036	-0,486	0,095	0,036	0,486	0,095	0,496
14	-0,244	-0,531	0,103	0,244	0,531	0,103	0,593
15	-0,366	-0,572	0,126	0,366	0,572	0,126	0,690
16	-0,571	-0,631	0,138	0,571	0,631	0,138	0,862
17	-0,570	-0,585	0,160	0,570	0,585	0,160	0,832
18	-0,461	-0,494	0,182	0,461	0,494	0,182	0,699
19	-0,481	-0,440	0,189	0,481	0,440	0,189	0,679
20	-0,338	-0,284	0,008	0,338	0,284	0,008	0,441
21	-0,309	-0,216	-0,025	0,309	0,216	0,025	0,378
22	-0,209	-0,114	-0,229	0,209	0,114	0,229	0,330
23	0,053	0,046	-0,349	0,053	0,046	0,349	0,356
24	0,246	-0,283	0,096	0,246	0,283	0,096	0,387
25	0,260	-0,063	0,056	0,260	0,063	0,056	0,273
26	0,198	-0,772	0,190	0,198	0,772	0,190	0,819
27	0,204	-0,444	0,190	0,204	0,444	0,190	0,524
28	0,243	-0,022	0,198	0,243	0,022	0,198	0,314
29	0,203	-0,688	0,214	0,203	0,688	0,214	0,749
30	0,235	-0,241	0,233	0,235	0,241	0,233	0,410

Vzor. [-]	ΔX [m]	ΔY [m]	ΔZ [m]	**ABS**(ΔX) [m]	**ABS**(ΔY) [m]	**ABS**(ΔZ) [m]	**Euklid. vzdial. [m]**
31	0,249	-0,926	0,266	0,249	0,926	0,266	0,996
32	0,257	-0,682	0,278	0,257	0,682	0,278	0,780
33	0,201	-1,576	0,320	0,201	1,576	0,320	1,621
34	0,225	-1,243	0,335	0,225	1,243	0,335	1,307
35	0,284	-0,275	0,365	0,284	0,275	0,365	0,538
36	0,239	-0,644	0,435	0,239	0,644	0,435	0,813
37	0,272	0,380	0,481	0,272	0,380	0,481	0,671
38	0,482	2,225	0,546	0,482	2,225	0,546	2,341
39	0,676	2,643	0,606	0,676	2,643	0,606	2,795
40	0,726	3,474	0,616	0,726	3,474	0,616	3,602
41	0,660	2,707	0,626	0,660	2,707	0,626	2,856
42	0,513	3,017	0,632	0,513	3,017	0,632	3,125
43	0,403	2,265	0,649	0,403	2,265	0,649	2,390
44	0,449	2,732	0,641	0,449	2,732	0,641	2,842
45	0,457	3,154	0,621	0,457	3,154	0,621	3,247
46	0,346	2,385	0,614	0,346	2,385	0,614	2,487
47	0,388	2,948	0,621	0,388	2,948	0,621	3,037
48	0,186	2,142	0,650	0,186	2,142	0,650	2,246
49	0,028	2,597	0,668	0,028	2,597	0,668	2,681
50	-0,197	2,970	0,662	0,197	2,970	0,662	3,049
51	-0,608	1,798	0,685	0,608	1,798	0,685	2,017
52	-0,764	2,146	0,710	0,764	2,146	0,710	2,386
53	-1,017	1,134	0,732	1,017	1,134	0,732	1,690
54	-1,069	1,517	0,746	1,069	1,517	0,746	2,000
55	-1,129	1,855	0,746	1,129	1,855	0,746	2,296
56	-1,223	1,070	0,760	1,223	1,070	0,760	1,794
57	-1,143	1,431	0,781	1,143	1,431	0,781	1,991
58	-0,971	0,778	0,740	0,971	0,778	0,740	1,448
59	-1,170	0,894	0,761	1,170	0,894	0,761	1,657
60	-1,536	0,927	0,775	1,536	0,927	0,775	1,954
61	-1,415	0,094	0,732	1,415	0,094	0,732	1,596
62	-1,808	-0,040	0,735	1,808	0,040	0,735	1,952
63	-1,408	-0,484	0,721	1,408	0,484	0,721	1,654
64	-1,723	-0,528	0,743	1,723	0,528	0,743	1,950
65	-1,351	-0,701	0,805	1,351	0,701	0,805	1,722
66	-1,570	-0,602	0,846	1,570	0,602	0,846	1,882

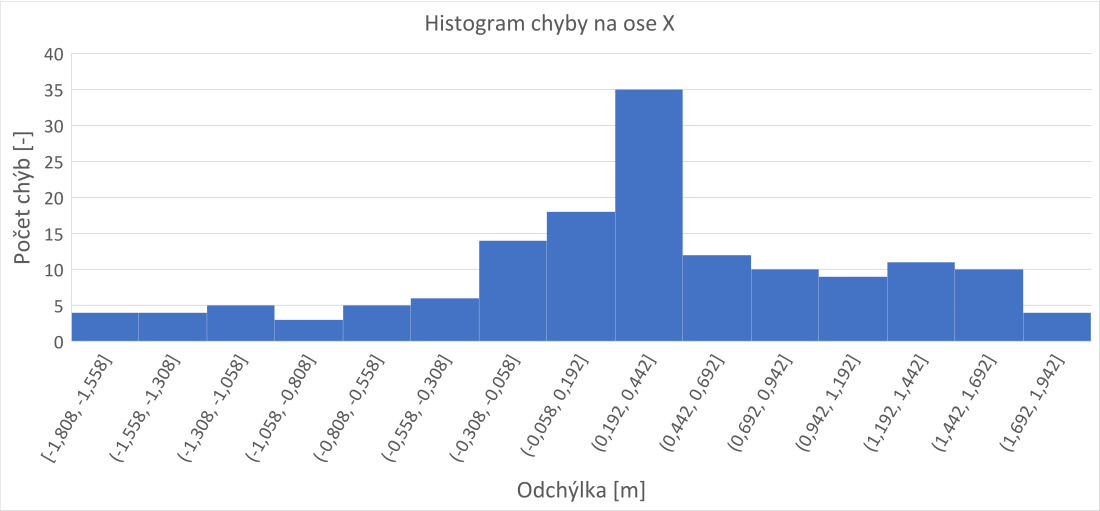
Vzor. [-]	ΔX [m]	ΔY [m]	ΔZ [m]	**ABS**(ΔX) [m]	**ABS**(ΔY) [m]	**ABS**(ΔZ) [m]	**Euklid. vzdial. [m]**
67	-1,581	-0,281	0,875	1,581	0,281	0,875	1,829
68	-0,701	-0,297	0,786	0,701	0,297	0,786	1,094
69	-0,987	-0,362	0,779	0,987	0,362	0,779	1,308
70	-0,162	-0,499	0,820	0,162	0,499	0,820	0,974
71	-0,544	-0,601	0,797	0,544	0,601	0,797	1,137
72	0,362	-0,498	0,833	0,362	0,498	0,833	1,036
73	0,162	-0,404	0,800	0,162	0,404	0,800	0,911
74	-0,159	-0,351	0,786	0,159	0,351	0,786	0,875
75	0,803	-0,310	0,852	0,803	0,310	0,852	1,211
76	0,329	-0,298	0,835	0,329	0,298	0,835	0,946
77	1,047	-0,391	0,858	1,047	0,391	0,858	1,409
78	0,580	-0,555	0,763	0,580	0,555	0,763	1,108
79	1,194	-0,338	0,728	1,194	0,338	0,728	1,439
80	0,955	-0,559	0,682	0,955	0,559	0,682	1,300
81	0,969	0,029	0,656	0,969	0,029	0,656	1,171
82	0,868	-0,571	0,603	0,868	0,571	0,603	1,201
83	0,838	-1,077	0,565	0,838	1,077	0,565	1,477
84	0,876	-0,416	0,571	0,876	0,416	0,571	1,125
85	0,675	-0,617	0,567	0,675	0,617	0,567	1,076
86	0,738	0,396	0,594	0,738	0,396	0,594	1,027
87	0,861	-0,094	0,587	0,861	0,094	0,587	1,046
88	0,987	-0,445	0,597	0,987	0,445	0,597	1,236
89	1,173	0,356	0,622	1,173	0,356	0,622	1,375
90	1,223	-0,144	0,589	1,223	0,144	0,589	1,365
91	1,393	0,678	0,603	1,393	0,678	0,603	1,663
92	1,415	0,141	0,575	1,415	0,141	0,575	1,534
93	1,626	0,930	0,590	1,626	0,930	0,590	1,964
94	1,663	0,105	0,565	1,663	0,105	0,565	1,759
95	1,664	-0,625	0,551	1,664	0,625	0,551	1,860
96	1,815	0,036	0,569	1,815	0,036	0,569	1,902
97	1,600	-0,584	0,561	1,600	0,584	0,561	1,793
98	1,503	0,038	0,554	1,503	0,038	0,554	1,602
99	1,382	-0,709	0,541	1,382	0,709	0,541	1,645
100	1,328	-1,444	0,537	1,328	1,444	0,537	2,034
101	1,443	-0,984	0,567	1,443	0,984	0,567	1,836
102	1,374	-2,090	0,522	1,374	2,090	0,522	2,555

Vzor. [-]	ΔX [m]	ΔY [m]	ΔZ [m]	**ABS**(ΔX) [m]	**ABS**(ΔY) [m]	**ABS**(ΔZ) [m]	**Euklid. vzdial. [m]**
103	1,263	-1,927	0,475	1,263	1,927	0,475	2,353
104	1,154	-2,618	0,435	1,154	2,618	0,435	2,894
105	1,193	-2,054	0,409	1,193	2,054	0,409	2,411
106	1,190	-2,641	0,380	1,190	2,641	0,380	2,921
107	1,285	-2,062	0,322	1,285	2,062	0,322	2,451
108	1,707	-2,420	0,275	1,707	2,420	0,275	2,974
109	1,895	-2,743	0,244	1,895	2,743	0,244	3,343
110	1,670	-2,010	0,234	1,670	2,010	0,234	2,624
111	1,656	-2,053	0,254	1,656	2,053	0,254	2,649
112	1,167	-1,321	0,260	1,167	1,321	0,260	1,782
113	1,479	-1,314	0,333	1,479	1,314	0,333	2,007
114	1,801	-1,372	0,462	1,801	1,372	0,462	2,311
115	1,310	-1,034	0,364	1,310	1,034	0,364	1,708
116	1,605	-1,185	0,392	1,605	1,185	0,392	2,033
117	0,827	-1,102	0,386	0,827	1,102	0,386	1,431
118	0,949	-1,464	0,421	0,949	1,464	0,421	1,795
119	0,168	-1,408	0,467	0,168	1,408	0,467	1,493
120	0,483	-1,636	0,491	0,483	1,636	0,491	1,775
121	0,853	-1,859	0,492	0,853	1,859	0,492	2,103
122	0,179	-1,673	0,482	0,179	1,673	0,482	1,750
123	0,668	-1,796	0,479	0,668	1,796	0,479	1,975
124	-0,081	-1,337	0,456	0,081	1,337	0,456	1,415
125	0,325	-1,478	0,450	0,325	1,478	0,450	1,579
126	-0,160	-1,166	0,409	0,160	1,166	0,409	1,246
127	0,307	-1,198	0,394	0,307	1,198	0,394	1,298
128	0,639	-1,218	0,369	0,639	1,218	0,369	1,424
129	-0,024	-0,902	0,341	0,024	0,902	0,341	0,965
130	0,257	-0,768	0,447	0,257	0,768	0,447	0,925
131	0,240	-1,005	0,477	0,240	1,005	0,477	1,139
132	0,616	-1,153	0,628	0,616	1,153	0,628	1,450
133	0,854	-1,402	0,678	0,854	1,402	0,678	1,776
134	-0,171	-1,736	-0,276	0,171	1,736	0,276	1,766
135	-0,089	-2,011	-0,311	0,089	2,011	0,311	2,037
136	-0,066	-2,199	-0,393	0,066	2,199	0,393	2,235
137	-0,086	-2,321	-0,413	0,086	2,321	0,413	2,359
138	-0,037	-2,443	-0,438	0,037	2,443	0,438	2,483

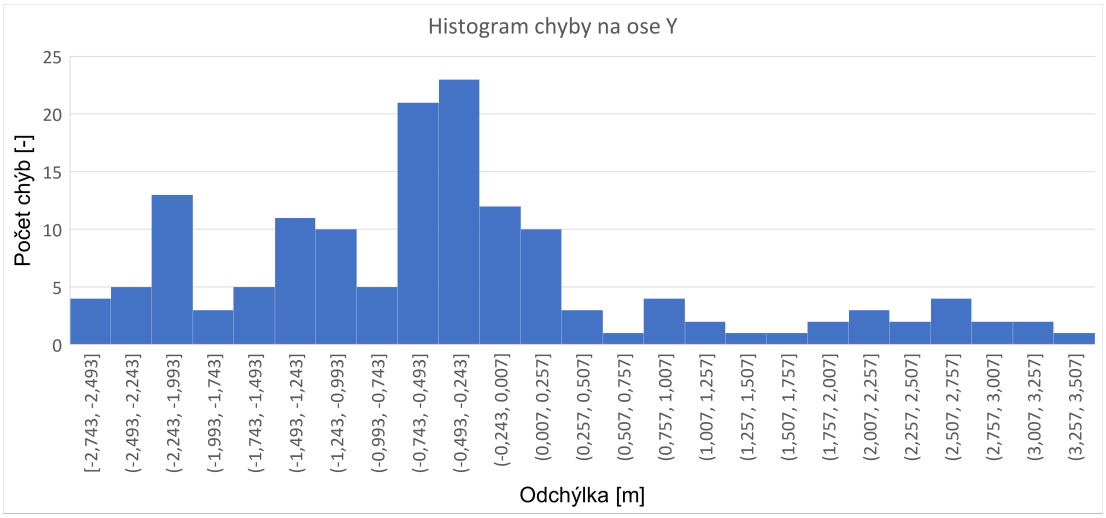
Vzor. [-]	ΔX [m]	ΔY [m]	ΔZ [m]	**ABS**(ΔX) [m]	**ABS**(ΔY) [m]	**ABS**(ΔZ) [m]	**Euklid. vzdial. [m]**
139	0,055	-2,510	-0,457	0,055	2,510	0,457	2,552
140	0,249	-2,306	-0,468	0,249	2,306	0,468	2,366
141	0,305	-2,126	-0,463	0,305	2,126	0,463	2,197
142	0,214	-2,141	-0,456	0,214	2,141	0,456	2,200
143	0,096	-2,196	-0,449	0,096	2,196	0,449	2,243
144	0,035	-2,280	-0,465	0,035	2,280	0,465	2,327
145	0,160	-2,228	-0,479	0,160	2,228	0,479	2,285
146	0,224	-2,132	-0,484	0,224	2,132	0,484	2,198
147	0,238	-2,007	-0,488	0,238	2,007	0,488	2,079
148	0,223	-1,672	-0,482	0,223	1,672	0,482	1,755
149	0,233	-1,464	-0,482	0,233	1,464	0,482	1,559
150	0,213	-1,433	-0,500	0,213	1,433	0,500	1,533

E.1 Histogram chyby v jednotlivých osiach

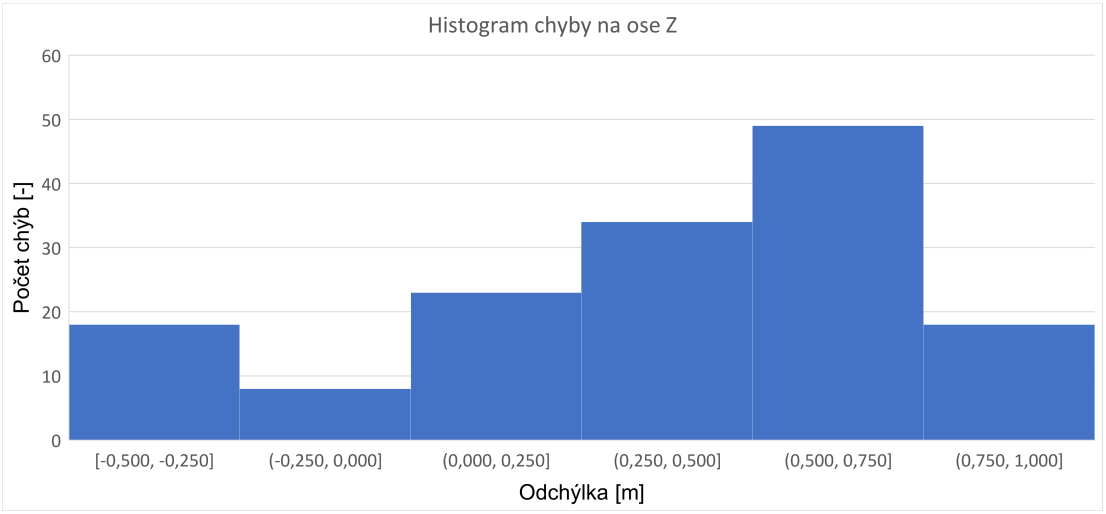
V nasledujúcich grafoch sú znázornené histogramy výskytu odchýlky v jednotlivých osiach.



Obr. E.1: Histogram chyby na ose X



Obr. E.2: Histogram chyby na ose Y

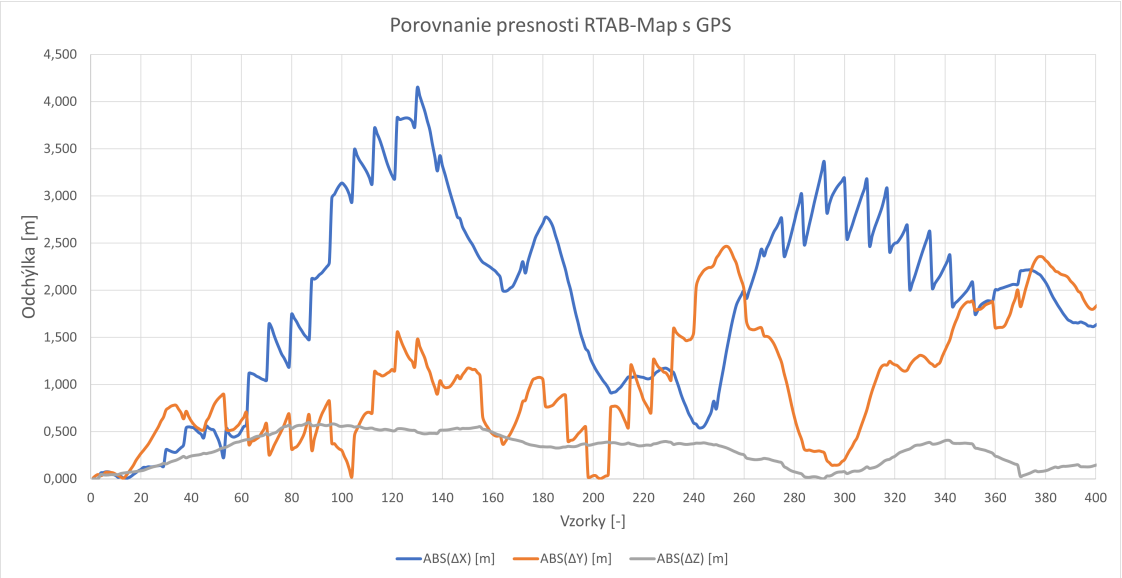


Obr. E.3: Histogram chyby na ose Z

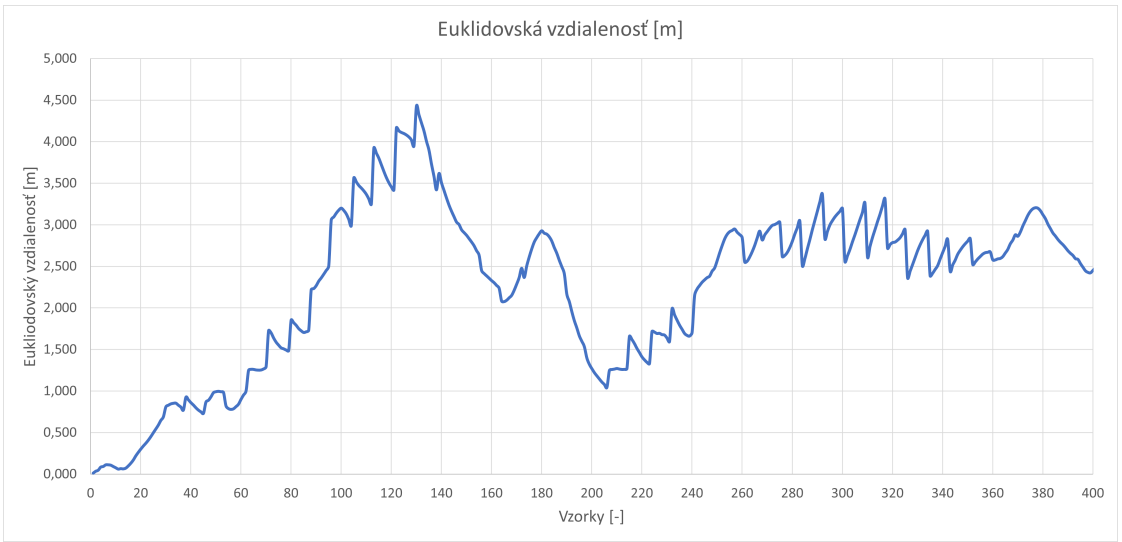
F Namerané dáta z ďalších experimentov

Grafy k druhému experimentu popísaného v kapitole 11.2.

F.1 Prvý experiment

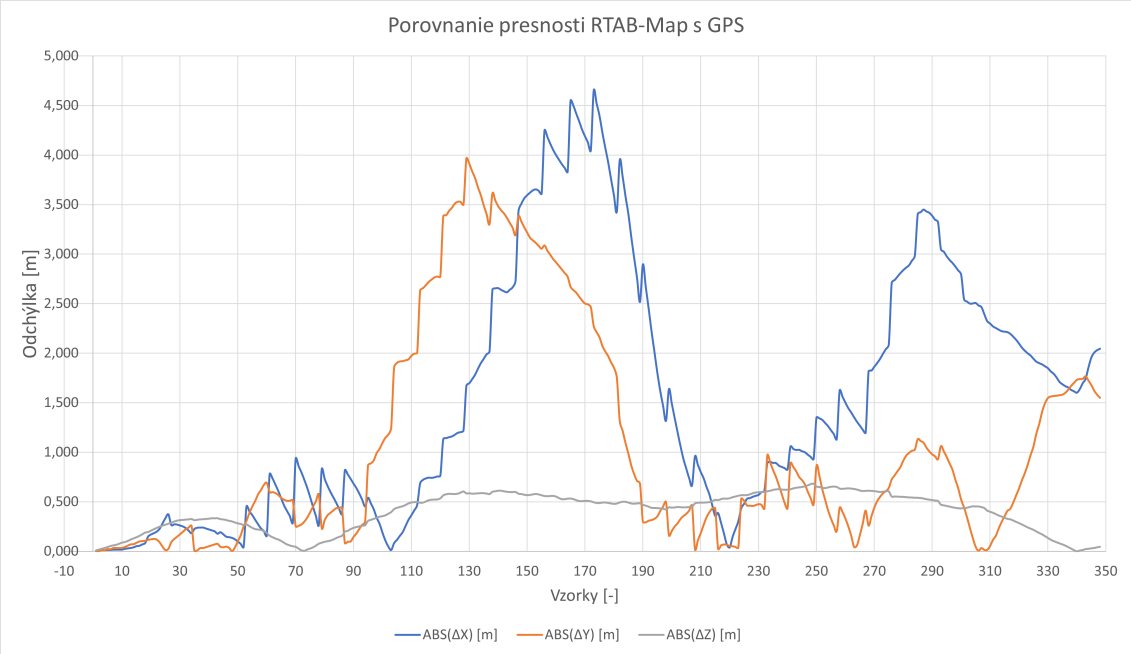


Obr. F.1: Graf porovnania absolútnej odchýlky

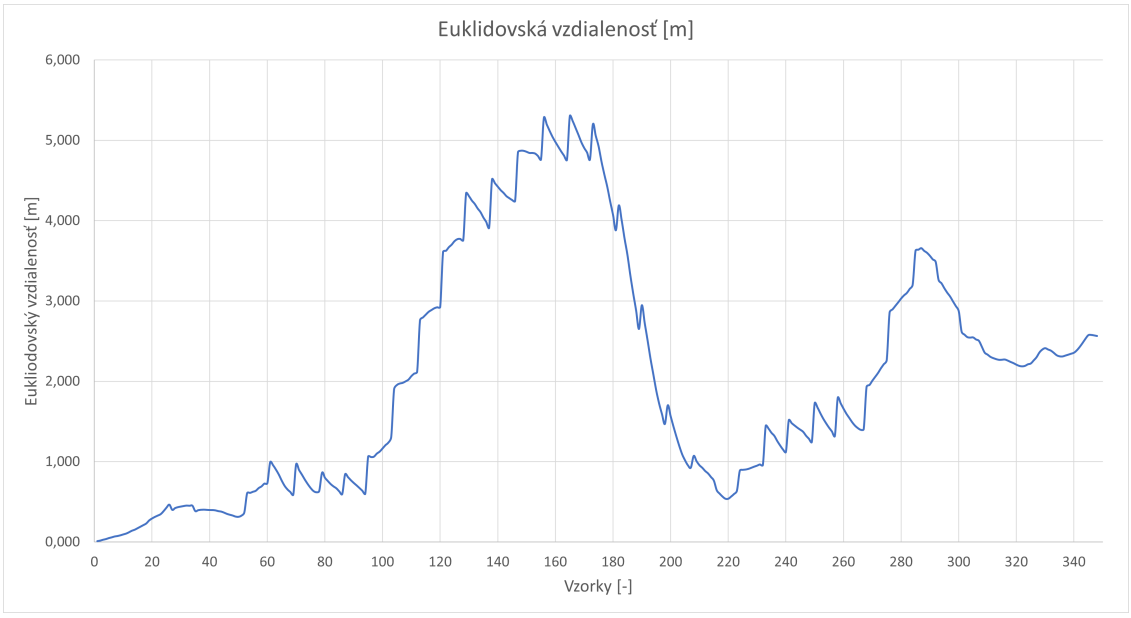


Obr. F.2: Graf porovnania euklidovskej vzdialenosti

F.2 Druhy experiment



Obr. F.3: Graf porovnania absolútnej odchýlky



Obr. F.4: Graf porovnania euklidovskej vzdialenosti