FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Sledování dronu

Jakub Herman

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Sledování dronu

Jakub Herman

Vedoucí práce
Ing. Zdeněk Bouček

PLZEŇ 2024

Citace v seznamu literatury:
HERMAN, Jakub. *Sledování dronu*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Zdeněk Bouček.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub HERMAN**
Osobní číslo:	**A21B0375P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Automatické řízení a robotika**
Téma práce:	**Sledování dronu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznámení se s úlohou sledování.
2. Návrh algoritmu pro sledování dronu jiným dronem.
3. Otestování algoritmu v simulačním prostředí.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

- V. Walter, N. Staub, A. Franchi and M. Saska, "UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs," in IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2637-2644, July 2019, doi: 10.1109/LRA.2019.2901683.
- J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok and A. P. Schoellig, "Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 7512-7519, doi: 10.1109/IROS51168.2021.9635857.
- Akhloufi, M.A.; Arola, S.; Bonnet, A. Drones Chasing Drones: Reinforcement Learning and Deep Search Area Proposal. Drones 2019, 3, 58.

Vedoucí bakalářské práce: **Ing. Zdeněk Bouček**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2023**
Termín odevzdání bakalářské práce: **20. května 2024**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

V Plzni dne 17. října 2023

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 31. prosince 2024

. . . . . .

Jakub Herman

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato bakalářská práce se zabývá vývojem a implementací algoritmu pro sledování dronů v konfiguraci leader-follower s využitím technologií ROS2 a PX4 Autopilot, včetně simulačního prostředí Gazebo garden. Cílem je navrhnout a experimentálně ověřit systém, který umožňuje sledujícímu dronu efektivně následovat vedoucí dron s vysokou přesností, i v dynamických nebo krizových situacích. V práci jsou popsány veškeré komponenty systému, včetně komunikace a metody detekce a lokalizace dronů. Výsledky experimentů ukázaly, že navržený systém dokáže přesně sledovat trajektorie a dynamicky reagovat na změny v prostředí. Práce v závěru diskutuje potenciál pro vylepšení v oblasti autonomních dronových technologií.

Abstract

This bachelor thesis deals with the development and implementation of a drone tracking algorithm in a leader-follower configuration using ROS2 and PX4 Autopilot technologies, including the Gazebo garden simulation environment. The aim is to design and experimentally validate a system that allows the tracking drone to effectively follow the leader drone with high accuracy, even in dynamic or crisis situations. The paper describes all components of the system including communication and methods for drone detection and localization. Experimental results show that the proposed system can accurately track trajectories and dynamically respond to environmental changes. The thesis concludes by discussing the potential for improvements in autonomous drone technology.

Klíčová slova

Drony • Gazebo Garden • Leader-follower • PX4 Autopilot • Regulace • ROS2 • Sledování dronů • Řízení dronu

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu práce, panu Ing. Zdeňku Boučkovi za odborné vedení, poskytnutí všech potřebných materiálů k dosažení požadovaných výsledků a poskytnutí potřebné pomoci se všemi problémy týkající se této bakalářské práce. Děkuji za trpělivost a ochotu pomoci mi překonat všechny výzvy, které se během práce objevily.

Obsah

Úvod 1

Drony, známé také jako bezpilotní letouny (Unmanned Aerial Vehicles – UAVs), se staly v posledních letech zásadní součástí moderní technologie, přičemž jejich aplikace sahají od rekreačních činností po klíčové průmyslové a vědecké využití [1]. Tyto bezpilotní letouny otevřely nové možnosti v mnoha oborech díky své schopnosti dosáhnout míst, která jsou těžce dostupná nebo nebezpečná. Využití dronů zahrnuje širokou škálu aplikací, včetně zemědělství, kde pomáhají monitorovat plodiny a spravovat zdroje, ve veřejné bezpečnosti, kde se využívají pro vyhledávací a záchranné mise, až po filmový průmysl, kde umožňují získávat záběry z výšky s ohledem na nízké náklady.

Sledování dronu v konfiguraci leader-follower je pokročilá technika řízení, která umožňuje několika dronům operovat koordinovaně, přičemž jeden dron (leader) určuje trajektorii a další drony (followers) automaticky sledují tento pohyb v předem definovaných formacích [2]. Tato technologie nachází uplatnění v řadě situací, od automatizovaného průzkumu a monitorování, kde více bezpilotních letounů může pokrýt větší plochu rychleji a efektivněji, až po vojenské a záchranné operace, kde koordinovaný pohyb může znamenat rozdíl mezi úspěchem a neúspěchem mise. Výzvy spojené se sledováním leader-follower zahrnují vývoj algoritmů pro přesné polohování, komunikaci mezi drony a adaptaci na dynamické změny v prostředí. Přes tyto technické překážky sledování otevírá nové možnosti pro automatizaci a efektivitu v operacích s drony, nabízí unikátní řešení pro komplexní výzvy a posouvá hranice toho, co je možné s moderními UAV systémy dosáhnout.

Tato práce se zaměřuje na návrh a implementaci systému pro sledování dronu v konfiguraci leader-follower s využitím technologie RGB-D a segmentační kamery. Cílem je vytvořit efektivní a spolehlivý algoritmus, který umožní sledujícímu dronu přesně sledovat vedoucí dron. Navržený systém využívá informace z kamer pro přesné určení polohy a vzdálenosti.

V kapitole 2 budou podrobněji přiblíženy a vysvětleny základní pojmy spojené s tímto tématem a jejich uplatnění v reálných aplikacích. Kapitola 3 představuje simulační prostředí a technologie využité v projektu, díky kterým bylo možné dosáhnout velmi realistických simulací a tak dron řídit jako reálný systém. Kapitola 4

detailně popisuje implementaci, funkci a komunikaci navrženého řídícího algoritmu sledování na základě dvou kamer, také je zde představena funkcionalita nalezení ztraceného dronu a popsáno uživatelské rozhraní. V kapitole 5 jsou prezentovány výsledky pečlivě zvolených experimentů tak, aby otestovali veškeré funkce algoritmu v krizových scénářích. Jsou zde také podrobně rozebrány jednotlivé situace v podobě grafů, na kterých je vysvětleno, jak dron reaguje v určitých situacích.

Sledování dronu 2

Tato kapitola představuje problematiku sledování dronu v reálných aplikacích. Jak již bylo zmíněno v předešlé kapitole, tento typ řízení je zásadní pro koordinaci letu dvou nebo více dronů. Dovoluje tedy vizuálně zpracovávat chování vedoucího dronu a efektivně na něj reagovat autonomně bez nutnosti zásahu operátora, což redukuje počet operátorů letky na pouze jednoho.

2.1 Seznámení s problematikou

V reálných aplikacích se s touto metodou můžeme setkat v mnoha odvětvích, například jako je zemědělství, kde tento systém umožňuje efektivně mapovat velké oblasti tím, že vedoucí dron stanoví trajektorii letu a mapovací vzor, zatímco sledující drony následují a zajišťují pokrytí širšího spektra oblasti. Tímto způsobem lze rychleji a efektivněji sbírat data o zdraví plodin, úrovních vlhkosti a dalších důležitých metrikách [3]. Dále jsou UAV stále častěji používány v záchranných misích, kde rychlá reakce může zachránit životy. V leader-follower konfiguraci může vedoucí dron identifikovat a prozkoumat oblasti zájmu, zatímco sledující drony můžou ve formaci detailněji skenovat oblast pro zvýšení úspěšnosti mise [4]. Využití najdeme i v armádních a obranných aplikacích, kde mohou v této konfiguraci plnit rozmanité role, od průzkumu a sledování až po taktické nasazení. Leader může řídit mise a poskytovat strategické informace, zatímco sledující drony provádějí specifické úkoly jako je zaznamenávání obrazu, skenování oblasti, nebo transport potřebných materiálů [5]. Kromě těchto tradičních aplikací, leader-follower konfigurace může být také využita v bezpečnostních aplikacích, kde sledující drony nejsou omezeny sledováním pouze známých vedoucích dronů, ale mohou autonomně sledovat i neznámé drony nebo objekty. Tyto aplikace však vyžadují koordinování a uspořádání více sledujících dronů najednou, což v práci není potřeba.

Řešení této problematiky představuje značnou výzvu, zejména kvůli složitosti detekce vedoucího dronu. Dále je potřeba se zaměřit na korekci vzdálenosti, které jsou založeny na analyzovaných datech, a rovněž na funkce, které umožňují lokalizaci ztraceného dronu.

Detekcí dronů se zabývá článek [6], který představuje systém UVDAR pro vizuální lokalizaci. Tento systém je navržen tak, aby umožnil efektivní sledování a lokalizaci v rámci formace leader-follower. Využívá speciálně upravené LED diody, které jsou umístěny na vedoucím dronu a emitují světlo v ultrafialovém spektru, ty jsou pak snímány kamerovým systémem vybaveným UV filtrem, který minimalizuje rušení způsobené okolním světlem a zlepšuje přesnost detekce vedoucího dronu. Tato funkce jde rozšířit o různé obnovovací frekvence diod a tak rozeznávat jednotlivé drony mezi sebou, nebo na tomto základě určovat orientaci sledovaného dronu. Dále se článek věnuje korekci vzdálenosti, na základě této detekce a prezentovány metodiky výpočtů parametrů nutné pro regulaci. Z tohoto článku byly obecně čerpány poznatky k této problematice a její řešení.

Další významný přístup k problematice představuje článek z časopisu Science Robotics [7], který popisuje plně autonomní navigaci roje dronů pomocí optimalizačního algoritmu. Každý dron v roji plánuje svou trajektorii a sdílí ji s ostatními členy roje. Tento systém umožňuje dronům létat ve formaci, autonomně se vyhýbat překážkám a eliminovat potřebu GPS signálu, což umožňuje provádění operace v hustě zalesněných nebo složitých prostředích. Algoritmus optimalizace trajektorií minimalizuje výpočetní nároky a zajišťuje efektivní a bezpečný pohyb dronů ve složitých prostředích, díky čemuž zajišťuje vysokou úroveň autonomie a koordinace.

2.2 Úloha sledování dronu

V této práci se zaměřujeme na úlohu sledování dronu v situacích, kde je kladen důraz na vysokou přesnost a spolehlivost, zejména v dynamických a krizových scénářích. Konkrétně je zde řešen problém sledování vedoucího dronu sledujícím dronem, kdy vedoucí dron určuje trajektorii letu a sledující dron musí tuto trajektorii autonomě následovat. Tato úloha zahrnuje vývoj a implementaci řídícího algoritmu, který se zaměřuje na přesnou detekci a lokalizaci vedoucího dronu pomocí vizuální a hloubkové kamery, na základě kterých se snaží regulovat svou polohu a orientaci. Řízení je pak podrobeno pečlivě zvoleným experimentům, které testují chování algoritmu.

Simulační prostředí a implementace — 3

V této sekci je podrobně rozebrán výběr a implementace technologií, které byly použity pro realizaci projektu. Každá zvolená technologie je zde představena a je vysvětlen její přínos pro dosažení stanovených cílů projektu. Dále je objasněn způsob její integrace do celkové architektury systému a popsán její vliv na funkčnost výsledného řešení.

3.1 Robot Operating System 2

Robot Operating System 2 [8], dále jen ROS2, je open-source middlewarový software, který je nástupcem ROS a slouží k vývoji komplexních robotických aplikací, poskytující nástroje a knihovny pro jejich tvorbu [9]. Modulárnost a efektivita hrají v projektu významnou roli z hlediska komunikace mezi autopilotem dronu, simulací nebo řídícím algoritmem. Jeho schopnosti v oblasti síťové komunikace zajišťují efektivní výměnu dat mezi komponentami systému a navíc, ROS2 podporuje real-time zpracování a je kompatibilní s různými operačními systémy, což rozšiřuje jeho použití a umožňuje integraci s řadou hardwarových a softwarových platforem včetně PX4.

3.1.1 Uzly, témata a služby

ROS2 využívá model publikování a odběru zpráv pomocí DDS a poskytování služeb, což umožňuje jednotlivým uzlům (anglicky "nodes") systému sdílet informace a žádat o služby. Data Distribution Service (DDS) je komunikační protokol a middleware standard, definován Object Management Group, navržený pro vysoce spolehlivou, škálovatelnou a efektivní distribuci dat v systémech. V kontextu ROS2 poskytuje základní infrastrukturu pro komunikaci mezi uzly, umožňuje real-time výměnu zpráv s vysokou úrovní spolehlivosti, flexibilitou a je široce využíván v různých aplikacích vyžadujících efektivní mezi systémovou komunikaci. Dále je uzel základním stavebním kamenem tohoto komplexního systému, fungující jako samostatný

proces zodpovědný za vykonávání specifických úkolů či algoritmů. Podstatou ROS2 je jeho schopnost nezávisle zpracovávat informace od ostatních uzlů prostřednictvím publikace a odběru zpráv, což umožňuje efektivní řešení specifických úkolů. Například, uzel může číst data z nějakého senzoru a následně tato data publikovat, což přináší významné výhody pro celý systém.

Každý uzel je vybaven funkcionalitou, která mu umožňuje veřejně sdílet informace na definovaném tématu (anglicky "topic"), jež je v systému identifikováno unikátním názvem. Toto téma, které uzel vytváří, se stává kanálem pro distribuci dat, jež jsou pak dostupná pro ostatní uzly, které mohou tyto informace přijímat a dále s nimi pracovat nebo je odesílat.

Vedle tohoto asynchronního modelu komunikace, ROS2 rovněž integruje služby (anglicky "services"), které umožňují synchronní a specifické interakce mezi jednotlivými uzly na základě modelu přijímání a odesílání zpráv. Model popisuje proces, kde jeden uzel (klient) požádá jiný uzel (server) o provedení operace nebo poskytnutí dat a čeká na jeho odpověď před pokračováním. Tento přístup zvyšuje efektivitu a přesnost distribuce zpráv v situacích, kde je vyžadována reakce v podobě odpovědi, ale musí se dbát na ošetření událostí když server klientovi neodpoví.

K tomuto jsou v ROS2 implementovány akce (anglicky "actions"), které představují rozšíření komunikačních možností . Dále umožňují uzlům vykonávat dlouhotrvající úkoly s možností poskytování průběžné zpětné vazby a dovolují klientům zrušit úkol před jeho dokončením, takže jsou zvláště užitečné v situacích, kde je vyžadováno nejen čekání na odpověď procesu, ale také potřeba monitorovat jeho průběh a následně reagovat na změny stavu během vykonávání.

Veškerá komunikace a prostředky použité v projektu s využitím ROS2 jsou podrobně vysvětleny v sekci 4.2.

3.1.2 Komunikace v ROS2

Komunikaci v ROS2 zajišťuje abstraktní middleware vrstva (RMW), která je kompatibilní mezi aplikacemi a různými implementacemi DDS, což umožňuje jej flexibilně využívat. DDS pak využívá protokoly TCP a UDP pro přenos dat, kde TCP zajišťuje spolehlivý a řazený přenos pro kritické komunikační úlohy a UDP nabízí rychlejší, ale potenciálně méně spolehlivý přenos pro aplikace vyžadující nízkou latenci [10]. Tato kombinace umožňuje optimalizovat komunikační vlastnosti podle specifických potřeb aplikací, čímž se zvyšuje již zmíněná efektivita, spolehlivost a škálovatelnost systémů.

3.2 PX4 autopilot

PX4 Autopilot je pokročilý open-source softwarový systém určený pro široké spektrum bezpilotních vozidel, včetně kvadrokoptér, středních letounů, VTOL (Vertical Take-Off and Landing) letadel a dokonce i podvodních plavidel. Jeho předností je vysoká modularita, která umožňuje jeho nakonfigurování s mnoha různými typy bezpilotních systémů. Jádrem tohoto systému jsou pokročilé algoritmy pro řízení letu, které umožňují manipulaci s bezpilotními vozidly v různých letových režimech a real-time operačním systémech, což je nezbytné pro bezpečný a stabilní let [11]. PX4 také disponuje nástroji pro plánování misí, které umožňují uživatelům předem definovat cesty letu a úkoly pro bezpilotní vozidlo.

Hlavní výhodou tohoto autopilotu, jsou sofistikované algoritmy řízení, které umožňují rozšíření funkcí dronu bez nutnosti implementace vlastních řešení. Tyto algoritmy zahrnují stabilizaci, řízení polohy a rychlosti, stejně jako odhad stavu dronu, což zjednodušuje celkový vývoj bez nutnosti vyvíjení a testování vlastních řešení.

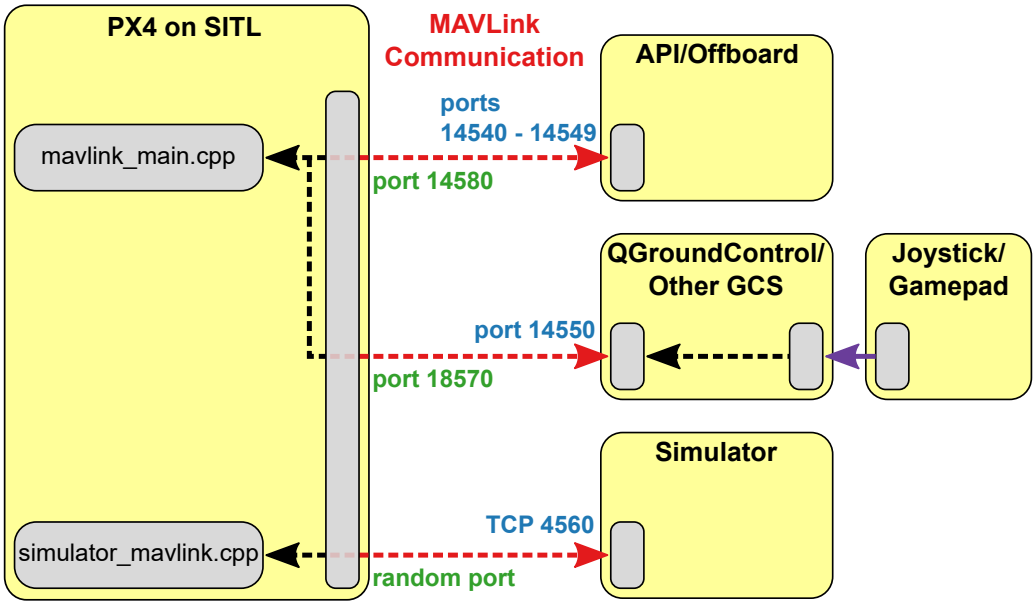
Veškerá komunikace probíhá přes MAVLink (Micro Air Vehicle Link) komunikační protokol, který je široce používán pro komunikaci v rámci malých a středně velkých bezpilotních letounů a slouží jako most pro telemetrii, řízení a konfiguraci mezi autopilotem a řídicí stanicí nebo jinými systémovými komponentami [12]. Protokol je navržen tak, aby byl co nejefektivnější s ohledem na šířku pásma a zároveň zachoval vysokou úroveň spolehlivosti, což umožňuje bezproblémovou komunikaci mezi různými částmi bezpilotního systému.

PX4 je také charakteristický vysokou míru bezpečnosti a spolehlivosti, na které aktivně pracuje rozsáhlá komunita a software udržuje jednou z nejmodernějších technologií pro autonomní let.

3.2.1 PX4 Autopilot SILT

Simulátory jsou velmi významnou částí jakéhokoliv vývoje prostředků nebo algoritmů, které jsou nějakým způsobem spojeny s realitou. V tomto případě nám umožňují ovládat simulované vozidlo skrze software PX4 a komunikovat s ním jako s vozidlem reálným, například pomocí aplikace QGroundControl, režimu offboard application programming interface (API) nebo rádiového ovladače. PX4 podporuje simulaci Software In the Loop (SITL), kde veškeré data jsou počítána z prostřednictví počítačového softwaru simulace, který může běžet na jiném počítači ve stejné síti [13]. Touto formou lze testovat jednotlivé typy vozidel anebo samotné algoritmy, než jsou nasazené do reálných funkcí. Tím se výrazně zmenšují náklady na vývoj, roste efektivita a je kladen velký důraz na bezpečnost. Další druh simulace nese název Hardware In the Loop (HIL), kde data běží na skutečné desce autopilotu a

pomocí simulačního firmwaru lze tuhle desku testovat úpravy PX4 Autopilot na reálném HW.



Obrázek 3.1: Přehled architektury PX4 SITL. Zdroj: [14]

Diagram na obrázku 3.1 popisuje detailní komunikaci simulačního prostředí PX4 SITL s ostatními komponentami a simulátory, které komunikují pomocí MAVLink zpráv. Jedinou výjimkou z podporovaných simulátorů je Gazebo, který běží v části ROS2 systému a tyto zprávy nepodporuje, proto musí být spuštěn balíček MAVROS, který tyto zprávy překládá do správného formátu a zajistí bezchybnou komunikaci s autopilotem.

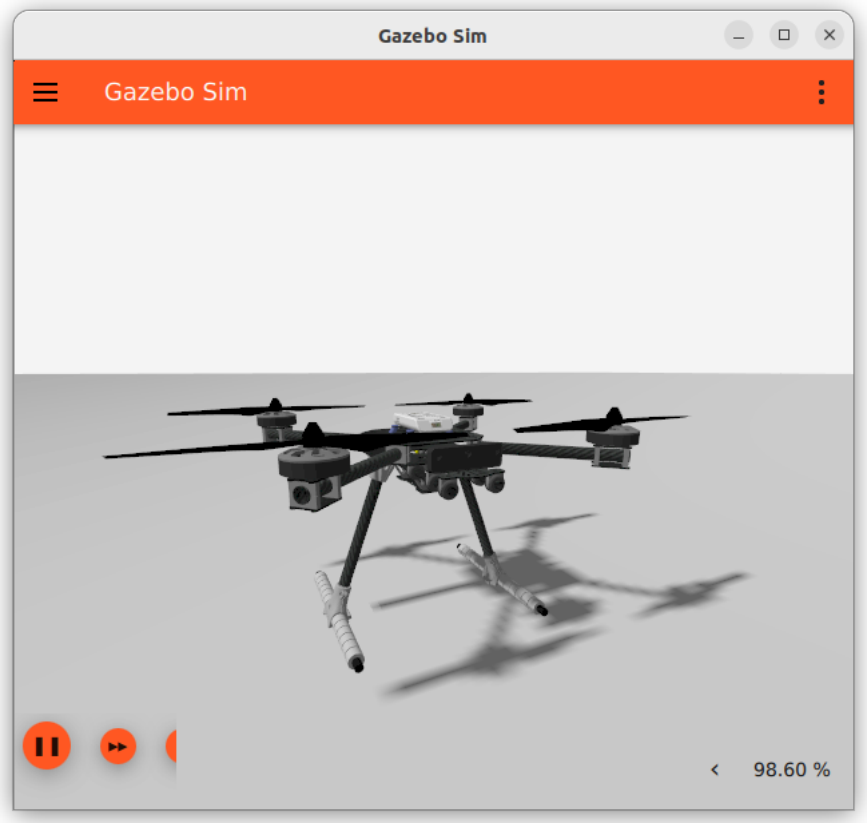
PX4 [15] používá specifický modul pro simulaci, který se připojuje k místnímu portu TCP 4560. Simulátory si pak s PX4 vyměňují informace pomocí rozhraní Simulator API MAVLink. Toto rozhraní definuje sadu zpráv MAVLink, které posílají PX4 data ze simulovaných senzorů a po přepočtení autopilot nastavuje hodnoty motorů a akčních členů, které jsou následovně použity na simulované vozidlo. Autopilot a simulátor mohou běžet buď na stejném zařízení, nebo na různých počítačích ve stejné síti. Ostatní komponenty systému mohou být také fyzicky rozloženy v rámci stejné sítě a jejich komunikace probíhá přes protokol UDP, kde specifické porty použité pro tuto komunikaci jsou uvedeny na obrázku 3.1.

3.2.2 Dron X500

Společnost Holybro vyvinula tento modulární dron, který se vyznačuje snadnou montáží bez potřeby pájení, díky čemuž je ideální pro vývojáře a vědce zaměřující se

na pokročilé aplikace a výzkum v oblasti autonomního létání [16]. Konstrukce je vyrobena z uhlíkových vláken, proto je velmi lehká a pevná. Řídící jednotka dronu podporuje nejmodernější autopilotní softwary jako jsou PX4 a Ardupilot. Dron může být hardwarově rozšířen o větší baterii, hloubkovou neboli RGB-D kameru, což umožňuje větší dolet s kvalitním kamerovým zpracováním.

V rámci PX4 SITL se pro simulace v prostředí Gazebo, popsané v sekci 3.4, využívá dron X500 z důvodů jeho široké podpory v komunitě. Tento model dronu je použit kvůli své vysoké modularitě, což umožňuje výzkumníkům a vývojářům testovat a optimalizovat různé letové algoritmy a senzorové systémy bez rizika poškození fyzického zařízení. Díky těmto vlastnostem se X500 stává preferovanou volbou pro simulace, které věrně napodobují reálné podmínky.



Obrázek 3.2: Ukázka dronu v simulačním prostředí s hloubkovou kamerou. Zdroj: [17]

3.2.2.1 Úprava modelu

3D model dronu je uložen v souboru formátu Simulation Description Format (SDF). Ten využívá XML pro definici objektů a scénářů v rámci různých robotických simulátorů, jako je například Gazebo. Tento formát umožňuje detailní parametrizaci

nejen konstrukce samotného dronu, ale i různých senzorů a aktuátorů, které model zahrnuje. Pro potřeby projektu je nezbytné tento model upravit a přizpůsobit pro specifické účely řízení. V kontextu úlohy sledování založené na vizuálním zpracování je detekce vedoucího dronu pomocí kamery fundamentálním krokem.

V praktických aplikacích je identifikace dronu v reálném prostředí komplexní výzvou, což se řadí do samotné problematiky, proto je k modelu přidána segmentační kamera. Tato komponenta je integrována do SDF souboru tak, aby byla optimalizována ve vztahu k poloze dronu, a je konfigurována tak, aby vizuálně rozlišovala pouze objekty označené specifickým pluginem, čímž se zaměřuje výhradně na vedoucí dron. Obnovovací frekvence kamery je nastavena na 24 FPS, což je považováno za dostatečné pro účely sledování bez potřeby častější aktualizace obrazu.

V novějších verzích autopilota PX4 je možné některé komponenty modelu importovat prostřednictvím URL odkazů. Tento přístup umožňuje centralizovanou aktualizaci komponenty na jediné URL adrese, což zjednodušuje správu a zajišťuje, že všichni uživatelé mají přístup k nejnovějším verzím. Nevýhodou však je, že uživatelé nemají možnost individuálně upravit parametry těchto komponent. V případě, kdy je potřeba umístit senzor s hloubkovou kamerou do stejné polohy jako segmentační kamera, je nezbytné přidat hloubkovou kameru jako externí komponentu s identickou frekvencí obnovy, i přestože model už jednu RGB-D kameru obsahuje. Cílem této modifikace je synchronizovat obraz z obou kamer, aby se polohy sledovaného dronu na obou záběrech překrývaly. Tento proces je zásadní, neboť detekce dronu na vizuálním záběru má potřebu dotazovat se na odpovídající bod na záběru z hloubkové kamery, aby bylo možné určit vzdálenost dronu.

3.3 MAVROS

Tento balíček v ROS2 slouží pro komunikaci pomocí MAVLink protokolu. Balíček umožňuje přímý přístup k telemetrii, řídicím a navigačním systémům prostřednictvím standardizovaných ROS2 zpráv a služeb, které MAVROS překládá do MAVLink zpráv a naopak [18]. Díky tomuto propojení a podpoře obousměrné komunikace můžeme vytvářet programy anebo řízení, které může využívat komunikaci mezi letounem a ROS systémem. V práci tento balíček umožňuje zprostředkovat komunikaci PX4 autopilotu s ostatními komponentami v systému.

3.4 Simulátor Gazebo Garden

Gazebo Garden [19] představuje nejnovější instanci jednoho z nejznámějších open-source simulátoru Gazebo, který je používán především pro simulaci robotů v 3D prostředí. Zaměřuje se především na realistické fyzikální modelování tak, aby si-

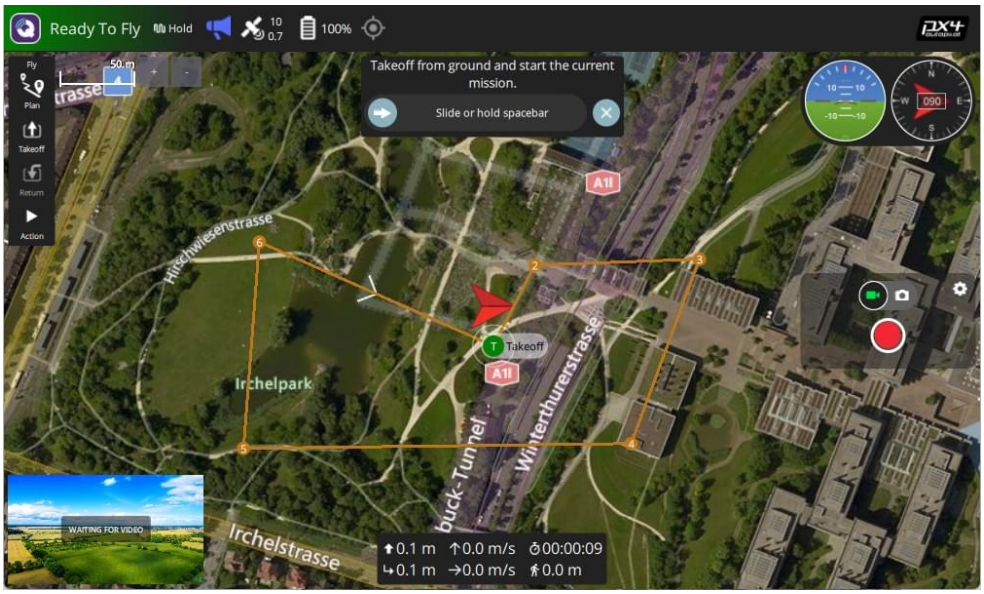
mulace odpovídala co nejvíce realitě i za specifických podmínek jako je vítr nebo modelování pohybu pod vodní hladinou.

Simulátor podporuje integraci s různými robotickými frameworky, zejména s systémem ROS2, což umožňuje bezproblémovou komunikaci a koordinaci mezi simulovanými robotickými systémy a aplikacemi. Gazebo Garden je vyvíjen za účelem modularity a široké aplikovatelnosti, takže umožňuje uživatelům přizpůsobit simulace specifickým potřebám projektů. Tímto způsobem redukuje náklady na testování, tvorbu prototypů, a hlavně zvyšuje bezpečnost. V simulátoru je každý simulovaný prvek reprezentován jako instance připojená přes TCP protokol, tedy lze s ním komunikovat i přes vzdálené připojení.

Jak již bylo zmíněno, simulátor se primárně zaměřuje na co nejvěrnější simulaci reálného prostředí. Díky svým schopnostem, jako je simulace kamerového záznamu, možnost modifikace modelů a realistické chování dronů, představuje ideální nástroj pro dosažení co nejrealističtějším podmínkám jak jen je to možné.

3.5 QgroundControl

Aplikace poskytuje uživatelsky přívětivé grafické rozhraní pro řízení a konfiguraci bezpilotních letounů či vozidel [20]. Je designovaná tak aby byla kompatibilní především s autopilotnímy systémy jako PX4 a ArduPilot, které patří k těm nejrozšířenějším. Dokáže monitorovat a vizualizovat telemetrická data, nastavovat a kalibrovat parametry řídících systémů, plánovat mise, vizualizovat kamerový přenos a to vše v reálném čase. QgroundControl je velmi efektivní a důležitý nástroj pří vývoji a testování řídících algoritmů bezpilotních vozidel.



Obrázek 3.3: Plánování letové mise v programu QgroundControl. Zdroj: [21]

Toto grafické rozhraní umožňuje nenáročné ovládání obou dronů najednou, včetně vizualizace trajektorie. Díky přehledným datům a možnosti parametrizace dronů tento nástoj přispěl k vývoji řídícího algoritmu a efektivního testování na specifických misích.

3.6 OpenCV

Open Source Computer Vision Library (OpenCV) je knihovna, která především zaměřená na real-time počítačové vidění a strojové učení [22]. Je jednou z nejrozšířenějších knihoven pro implementaci vizuálních algoritmů, která používána v širokém spektru aplikací, od jednoduchých úloh zpracování obrazu až po složité systémy rozpoznávání obličejů, navigace autonomních vozidel a analýzy medicínských obrazových dat.

Z příkladu použití je patrné, že je navržena s důrazem na efektivitu a optimalizaci, a díky hojné rozšířenosti je tedy podporována různými programovacími jazyky, jako jsou C++, Python, Java a další. Toto je velmi velká výhoda při implementaci do specifické úlohy na různých operačních systémech. Knihovna obsahuje více než 2500 optimalizovaných algoritmů, které zahrnují funkce od základního zpracování obrazu, například filtrace a morfologické operace, přes vyšší úrovně počítačového vidění, jako jsou detekce objektů, sledování pohybu, rozpoznávání obličejů a gest.

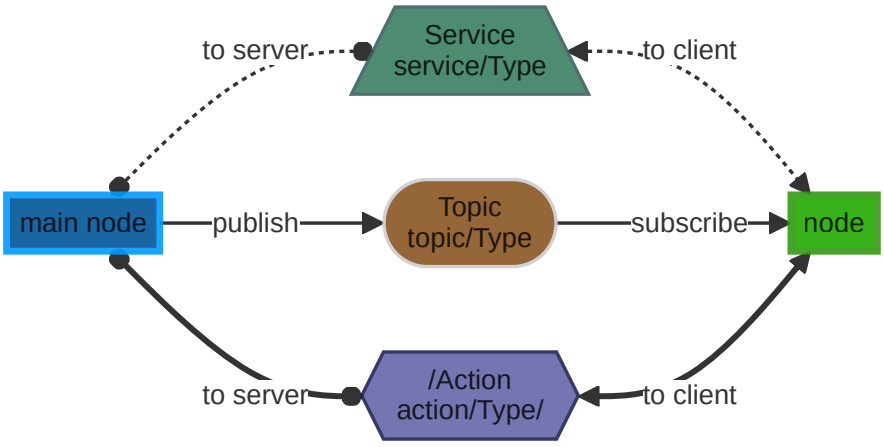
Tato knihovna se využívá ke zpracování obrazových dat streamovaných přímo z kamer. Umožňuje rozlišit segmentované záběry dronu a následně provádět výpočty rozdílů mezi body nebo analyzovat data z hloubkové kamery, čímž získává vektorovou vzdálenost mezi nimi. Detailnější vysvětlení tohoto procesu je poskytnuto v sekci 4.3.

Návrh algoritmu pro sledování

4

V této kapitole je prezentována nejen implementace algoritmu řízení, ale i vizualizace komunikačního schémat, které jsou využívány pro interakci mezi ROS a autopilotním systémem PX4. Vizuální reprezentace slouží k lepšímu pochopení toho, jak různé uzly v prostředí komunikují mezi sebou a s PX4, a to pomocí specifických komunikačních prvků, jako jsou témata, služby a akce.

Na obrázku 4.1 je reprezentována legenda popisující způsob značení jednotlivých prvků v následujících komunikačních diagramech ROS2 prostředí.



Obrázek 4.1: Legenda diagramu ROS

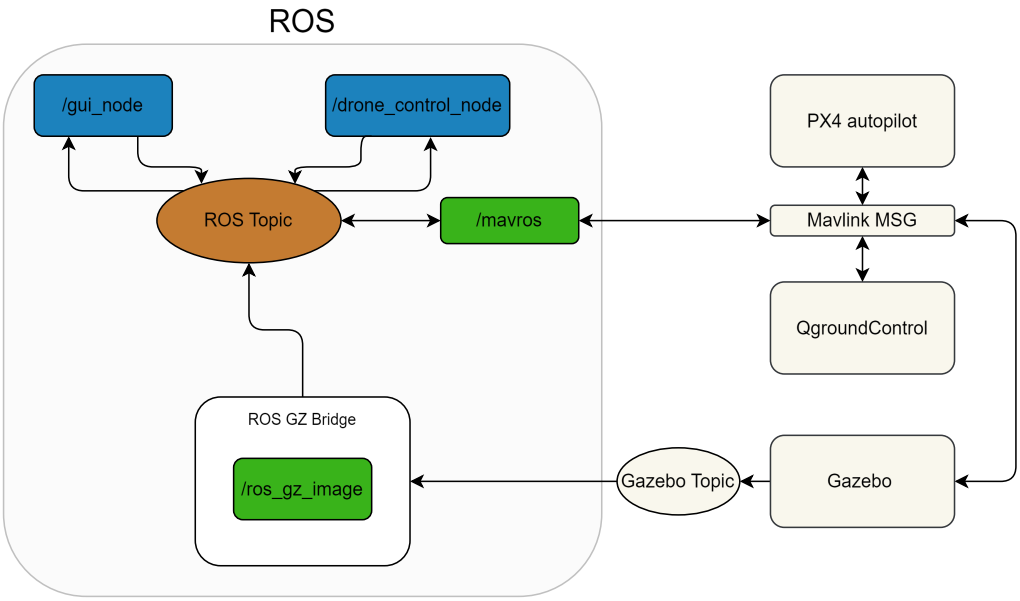
Tato legenda poskytuje vysvětlení symbolů a terminologie použité v komunikačních diagramech:

- 'Main node' reprezentuje uzel v systému, který je součástí řídícího algoritmu.
- 'Topic' je téma, kde jsou zprávy publikovány a odebírány.
- 'node' reprezentuje uzel v systému.

- 'Service' představuje specifický typ komunikace, kde klient může posílat požadavky na server a očekávat odpovědi.
- 'Action' je komunikační mechanismus pro dlouhotrvající úkoly s možností zpětné vazby a přerušení akcí.

4.1 Architektura systému

Schéma architektury systému, které je zobrazeno na obrázku 4.2, udává komunikační strukturu a rozdělení procesů v rámci celého systému. Zde jsou uvedeny jednotlivé softwarové komponenty interagují mezi sebou nejen v rámci ROS2, ale i externě mimo něj. Příkladem je PX4 autopilot, který běží jako samostatný nezávislý proces komunikující s ostatními komponentami skrz zmíněné MAVLink zprávy, ty následně zpracovává QgroundControl jenž slouží jako uživatelské rozhraní pro interakci s drony, vizualizuje jejich stav a především dokáže plánovat mise a navigovat drony na určité souřadnice.



Obrázek 4.2: Diagram architektury systému

V rámci simulovaného systému jsou spuštěny dvě instance autopilota PX4, přičemž první instance reprezentuje sledující dron vybavený kamerovým systémem a druhá představuje vedoucí dron, jenž koordinuje letecké operace. Po spuštění se obě instance připojují k simulovanému prostředí, avšak simulátor je schopen napřímo komunikovat s PX4, ROS pro vzájemnou komunikaci a interakci vyžaduje specifický

balíček funkcí, uveden v sekci 3.3, neboť není schopen nativně tyto zprávy zpracovávat. Zásadní roli zde hraje uzel */mavros*, který funguje jako komunikační most mezi ROS ekosystémem a autopilotním softwarem PX4, a to konkrétně prostřednictvím překladu MAVLink zpráv do formátu ROS Topic. Tyto zprávy následně umožňují obousměrný tok telemetrických dat mezi komponentami systému.

Mezi důležitá data patří záznamy z RGB-D kamery umístěné na sledujícím dronu. Tato kamera poskytuje nejen obraz v RGB formátu, kde RGB znamená červenou (Red), zelenou (Green) a modrou (Blue) barvu, což jsou základní barvy používané pro digitální zobrazení obrazu, ale také hloubkové mapy (parametr 'D'), což je nezbytné pro prostorovou orientaci. Dále využívá segmentační kameru, uvedenou v skeci 4.3, která je specializovaná na identifikaci a sledování jednotlivých dronů v záběru. Tato kombinace obou kamerových záznamů v reálném čase je následně využívána ve zpětné vazbě pro řídící algoritmus. V prostředí ROS je simulátor Gazebo integrován s vlastními GZ tématy (Gazebo Topic), prostřednictvím kterých simulátor komunikuje, je důležité poznamenat, že témata v Gazebo a standardní ROS Topic nejsou vzájemně propojeny a využívají specifickou implementaci. Pro tento účel je využit balíček ROS GZ Bridge, který umožňuje jednosměrnou komunikaci tématům z Gazebo do ekosystému ROS. Balíček poskytuje rozsáhlé možnosti uzlů téměř pro veškeré typy GZ témat, ale v našem případě je použit uzel */ros_gz_image*, jenž zajišťuje konverzi kamerových záznamů z dronu do ROS Topic kanálů.

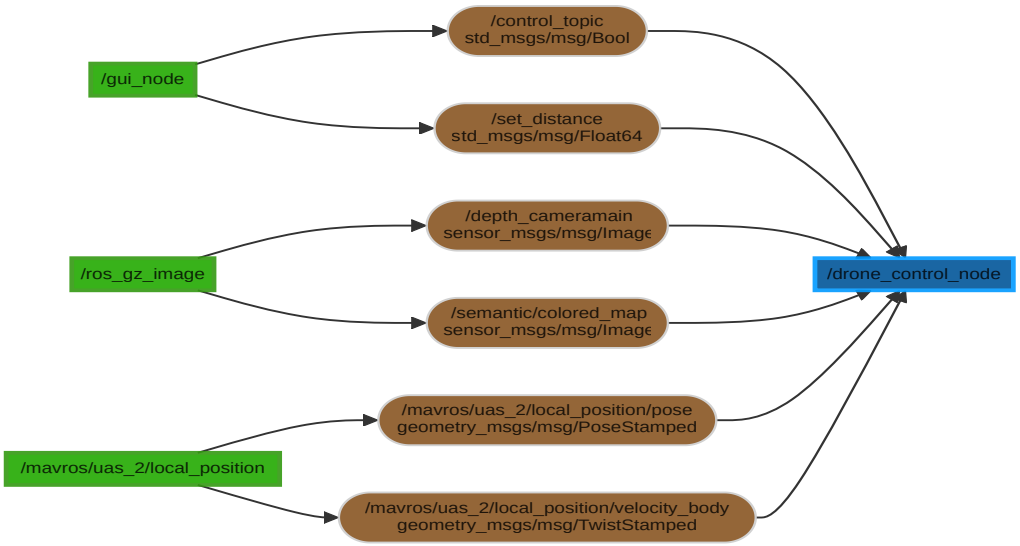
V diagramu systému jsou zřetelně identifikovány dva primární kontrolní uzly: */gui_node* a */drone_control_node*. Uzel */gui_node* hostuje Qt aplikaci uživatelského rozhraní (prezentována v sekci 4.7), která umožňuje interaktivně ovládat funkcionalitu pro aktivaci, vzlet a přepínání do Offboard módu. Toto grafické rozhraní dále nabízí vizualizaci letového prostoru v 3D a poskytuje indikátory stavu připojení obou dronů. Dále je zde možnost nastavit specifickou vzdálenost pro sledování a zapínat nebo vypínat řídicí systém, který je spuštěn uzlem */drone_control_node*. Tento uzel je zodpovědný za zpracování obrazu z kamery a vysílá instrukce pro rychlost v jednotlivých osách a úhlovou rychlost rotace podél osy z, aby bylo zajištěno optimální sledování vedoucího dronu, který koordinuje letové mise.

4.2 Komunikace řídícího algoritmu v ROS2

Autopilotní systém PX4, simulátor Gazebo a uživatelské rozhraní jsou zásadní komponenty poskytující informace pro řídicí algoritmus. Tento algoritmus shromažďuje data z různých uzlů, které publikují specifická témata s definovanými datovými typy, vizualizované na obrázku 4.3. V diagramu je uveden název každého tématu spolu s příslušným datovým typem, který je upřesněn pod názvem. Řídicí algoritmus primárně odebírá dva klíčové typy dat: pozici sledujícího dronu, publikované uzlem */mavros*, z něhož algoritmus extrahuje souřadnice x, y, z, a data z uzlu */ros_gz_image*,

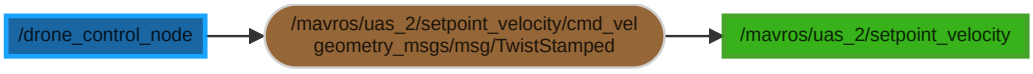
který poskytuje segmentovaný video přenos a data z hloubkové kamery pro vizuální detekci dronů v prostoru. Způsob zpracování a využití dat je uveden je uveden v sekci 4.3

V diagramu lze dále pozorovat, že řídící algoritmus přijímá data z uzlu */gui_node*, což představuje dříve zmíněné grafické uživatelské rozhraní (vysvětleno v sekci 4.7). Tento uzel je využíván pro spuštění nebo zastavení řídícího algoritmu, prostřednictvím publikovaných booleovských hodnot true nebo false na téma */control_topic*. Další téma */set_distance* je určeno pro definování cílové vzdálenosti, na kterou má být vzdálenost sledujícího dronu od vedoucího dronu udržována a regulována.



Obrázek 4.3: Diagram odesílání požadavků do autopilota

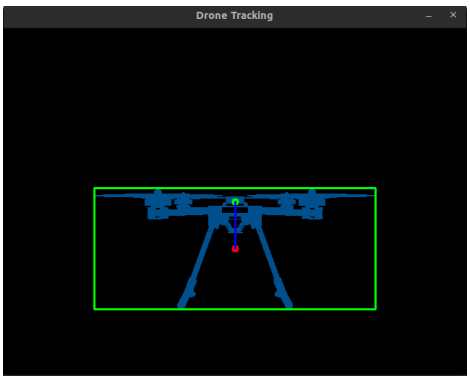
Na Obrázku 4.4 je znázorněna část schématu, ilustrující proces, v rámci něhož řídící algoritmus určuje požadované rychlosti a orientace dronu vůči jednotlivým osám. Uzel */drone_control_node* posílá rychlosti prostřednictvím publikování dat na uzel */mavros*. Tato komunikace představuje předávání řídících příkazů od navigačního stacku ROS2 k modulu PX4 Autopilot, čímž se koordinuje samotný dron.



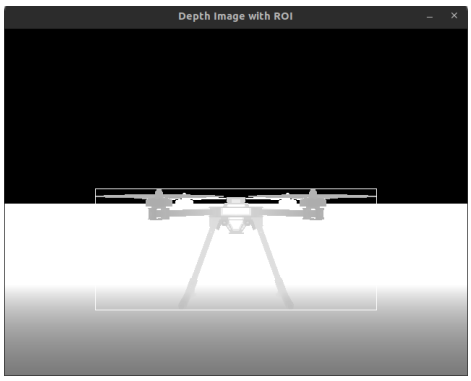
Obrázek 4.4: Diagram odběru dat řídícím algoritmem

4.3 Detekce dronu

Detekce dronu představuje základní stavební kámen algoritmu pro řízení a je nezbytná pro dosažení efektivní regulace. V našem přístupu je tato fáze realizována pomocí segmentační kamery, která je schopna identifikovat a vizuálně oddělit objekty v zorném poli kamery prostřednictvím aplikace specifických barevných filtrů. Tento postup umožňuje bezprostřední vizuální rozpoznání dronu bez nutnosti vzájemné komunikace mezi letouny. Pro praktické využití v reálných podmínkách lze segmentační kameru nahradit systémem UVDAR, který byl zmíněn v sekci 2.1, a nakonfigurovat ho za účelem reálné aplikace.



Obrázek 4.5: Vizualizace pohledu drona skrz segmentační kameru



Obrázek 4.6: Vizualizace obrazu hloubkové kamery

Knihovna OpenCV umožňuje efektivní zpracování a vizualizaci obrazových dat. Po detekci dronu v obrazu je kolem něj vytvořen obdélník, definovaný jako ROI (Region of Interest), ve kterém je následně vypočítán a zvýrazněn střed dronu pomocí červeného bodu. Zelený bod označuje střed kamery, jak je znázorněno na obrázku 4.5. Rozdíl mezi polohou těchto dvou bodů je využit k regulaci výšky a natočení, princip je více rozveden v sekci 4.5.1. ROI je také promítnut do obrazu z hloubkové kamery, z něhož je poté získána minimální vzdálenost v rámci dané oblasti.

4.4 Regulace

V této sekci je prezentován princip, jímž řídící systém reguluje požadované vzdálenosti v horizontální a vertikální rovině, a popisuje metodu výpočtu řídících signálů. Regulace je založena především na analýze zpětné vazby získané z obrazového záznamu (vysvětleného v sekci 4.3). Klíčovým krokem je převod rozdílu v pixelech na jednotky rychlosti a rozdělení vektorové vzdálenosti mezi drony na individuální rychlosti pro osy X a Y.

4.5 PID regulátor

Řídící systém potřebuje provádět korekce rychlostí, které jsou přepočítány například z pixelů obrazů. Proto Algoritmus vychází z fundamentálních principů klasické teorie řízení. Konkrétně využívá PID (Proporcionální-Integrační-Derivační) regulátor 90% reálných aplikací a z toho 97% tvoří regulátor typu PI (Proporcionální-Integrační), neboli bez derivační složky [23]. Tento regulátor je založen na základě zpětné vazby, kde odezva systému je neustále srovnávána s požadovanou hodnotou (referenční signál) a na základě tohoto porovnání je generován regulovaný výstup. Výstup regulátoru, kterým je akční zásah, je poté aplikován na řízený systém s cílem minimalizovat odchylku od referenční hodnoty a udržet systém stabilní a v požadovaném pracovním bodu.

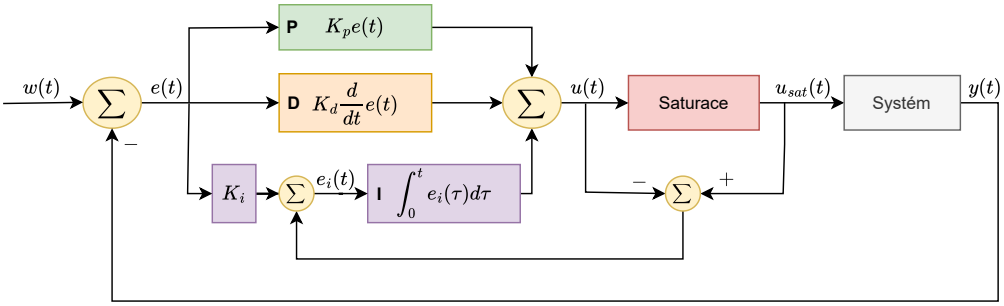
Matematický model PID regulátoru je reprezentován vztahem

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t), \tag{4.1}$$

kde K_p, K_i, a K_d jsou konstanty proporcionální, integrační a derivační složky regulátoru, které se nastavují podle potřeby úlohy. Dále $e(t)$ je regulační odchylka, tedy rozdíl mezi referenční a aktuální hodnotou měřené veličiny reprezentovaná jako:

$$e(t) = w(t) - y(t), \tag{4.2}$$

kde $w(t)$ představuje vstupní referenční signál a $y(t)$ aktuální hodnotu měřené veličiny. Na obrázku 4.7 je bloková vizualizace PID regulátoru. Proporcionální složka P reaguje přímo na aktuální odchylku a chybu regulace, integrální složka I akumuluje odchylky v čase a reaguje i na minulé hodnoty. Akumulací odchylky může nastat tzv. unášení této složky. Pokud je chyba moc velká, regulátor nemusí pracovat dle očekávání. Proto je zde zavedeno řešení unášení integrální složky, která se používá při saturovaném vstupu do systému. Když je vstup saturovaný, složka I dále neroste a naopak klesá na úroveň saturace. Derivační složka D předpovídá budoucí chyby, čímž zvyšuje stabilitu a rychlost odezvy systému, ale zároveň zavádí nechtěnou reakci šumu a skokovou změnu reference.



Obrázek 4.7: Diagram PID regulátoru

Efektivita, robustnost a rychlost regulovaného systému jsou ovlivněny správným nastavením konstant K_p, K_i, a K_d PID regulátoru. Existují různé přístupy k jejich nastavení, včetně metody Ziegler–Nichols, Loopshaping, nebo experimentálního ladění. Pro první dva zmíněné přístupy je nutná existence fyzikálního modelu daného systému, který umožňuje přesnější nastavení konstant. Tyto metody otevírají možnosti nakonfigurování regulátoru určitému systému přímo na míru.

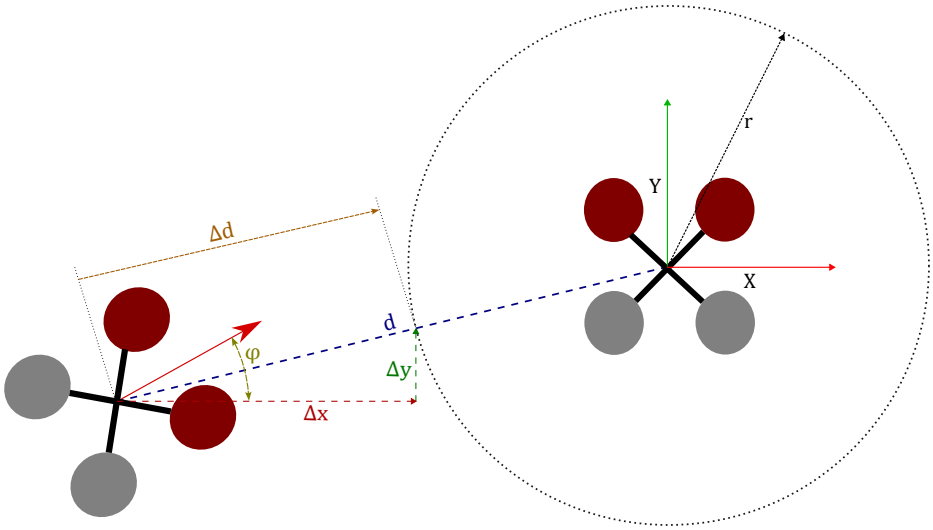
4.5.1 Regulace horizontální vzdálenosti

Tato sekce se zaměřujeme na vývoj řídicího systému, založeného na zpětné vazbě z kamery, který by byl nejenom robustní vůči různým provozním podmínkám, ale také schopný efektivně reagovat na krizové scénáře. Tyto scénáře zahrnují dynamický let, kde se musí vypořádat s rychle měnícími se letovými podmínkami, ale i situace, kdy dojde k vizuální ztrátě a je potřeba ho rychle a přesně lokalizovat. Schopnost efektivního detekování a následného nalezení dronu po jeho vizuálním ztracení je klíčovou: a je přiblížena v sekci 4.6, kde jsou představeny metody pro rychlou lokalizaci sledovaného dronu.

V rámci regulovaného systému je implementována sada čtyř PID regulátorů, které dynamicky kontrolují rychlosti dronu. Dva z těchto regulátorů jsou specificky navrženy pro horizontální regulaci a umožňují řídit vzdálenost v horizontálních osách X a Y, což jsou hlavní osy pohybu. Tyto regulátory zajišťují, že dron může efektivně navigovat a manévrovat ve svém prostředí, reagovat na změny trajektorie a udržovat stabilní letovou dráhu. Třetí z regulátorů je zaměřen na vertikální regulaci, která má za cíl eliminovat jakýkoliv rozdíl ve výšce mezi drony, a tím zajistit, pohyb ve stejné úrovni. Poslední regulátor se věnuje natáčení kolem osy Z, což umožňuje udržení cílového dronu v zorném poli kamery a adaptovat svou orientaci tak, aby optimálně sledoval pohybující se objekt, nebo aby zůstal ve správném úhlu vůči statickému cíli.

Nastavení konstant v rámci řízení bylo realizováno experimentálně, protože identifikace modelu pro efektivní reprezentaci zpětnovazebního systému dronu je značně komplikovaná. Regulátory byly tedy nastavovány při specifických scénářích, což dává prostor a možnosti pro lepší konfiguraci konstant. Vzhledem k tomu, že dron často mění dynamiku letu, chová se systém v různých situacích odlišně, zejména v horizontálním a vertikálním režimu letu, je vhodné za účelem zvýšení kvality řízení použít pro každou situaci speciálně navržené regulátory.

Obrázek 4.8 ilustruje fyzikální a geometrické parametry využívané v řízené soustavě. Proměnná r reprezentuje žádoucí vzdálenost, kterou má sledující dron udržovat od dronu vedoucího, a je znázorněna jako poloměr kružnice.



Obrázek 4.8: Zobrazení parametrů v rámci regulace dronů v rovině XY

Důvodem, proč se uvažuje kruhová trajektorie, je parametr d, který definuje minimální vektorovou vzdálenost mezi drony zaznamenanou hloubkovou kamerou. Rozdíl Δd pak představuje chybu vzdálenosti, tedy odchylku, kterou je třeba korigovat, aby sledující dron dosáhl požadované vzdálenosti r. Ideálně by tato chyba měla být nulová. Výpočet chyby vzdálenosti je vyjádřen následující rovnicí:

$$\Delta d = d - r. \tag{4.3}$$

Vzhledem k tomu, že Δd představuje zmíněnou vektorovou vzdálenost mezi drony, je potřebné tuto vzdálenost rozložit na složky odpovídající jednotlivým osám X a Y, aby bylo možné přesně regulovat pohyb s využitím úhlu φ mezi směrem dronu a osy X. Tento rozklad využívá trigonometrické funkce sinus a cosinus k transformaci vektorové vzdálenosti na chyby vzdálenosti ve směrech os X a Y, což umožňuje přizpůsobit řídicí signály pro každou osu zvlášť. Výpočet chyb pro tyto osy je vyjádřen

jako:

$$\Delta x = \Delta d \cdot \cos(\varphi), \quad \Delta y = \Delta d \cdot \sin(\varphi), \tag{4.4}$$

kde úhel φ je vypočítán z kvaternionu $q = [w_q, x_q, y_q, z_q]^\mathsf{T}$, jež popisuje orientaci dronu, jako:

$$\varphi = \arctan 2 \left(2 \cdot (w_q \cdot z_q + x_q \cdot y_q), 1 - 2 \cdot (y_q^2 + z_q^2)\right). \tag{4.5}$$

Je důležité zmínit, že tento přístup předpokládá, že dron je ve stabilizované poloze. V případě, že je nakloněný, nemusí úhel φ odpovídat přímo skutečnému směru letu (heading), jelikož průmět do osy Z může být zkreslený, zejména když se změny v yaw mohou projevovat opačnou změnou headingu v závislosti na orientaci dronu vůči lokálnímu rámu.

Výpočet φ pomocí arctan 2 z kvaternionových komponent zajišťuje přesné určení směrového úhlu. Funkce arctan 2, která uvažuje dva argumenty a poskytuje výhodu v tom, že dokáže rozlišit kvadrant, ve kterém se výsledný úhel nachází a umožňuje tím pokrytí celého rozsahu 360° (nebo 2π radiánů).

Dalším zásadním aspektem je také efektivní udržování pozice vedoucího dronu ve vizuálním poli, a to na základě informací získaných z vizuální zpětné vazby. V tomto kontextu $\Delta x_{img} = x_{imgCenter} - x_{LimgCenter}$, kde Δx_{img} reprezentuje rozdíl mezi pozicí středu zorného pole kamery sledujícího dronu (na obrázku 4.5 označený zeleným bodem) a polohu středu vedoucího dronu (na obrázku 4.5 červený bod). Tento rozdíl určuje, zda by sledující dron měl vykonat korekci směru doleva či doprava, aby udržel vedoucí dron optimálně ve svém vizuálním poli.

V rámci regulace horizontální polohy dronu je nezbytné udržet přesnou a efektivní kontrolu nad pohybem v osách X a Y. Tato kontrola je řízena následujícím souborem podmínek, které určují kdy a jak bude dron reagovat na změny v prostředí.

- **Kontrola podmínek pro horizontální regulaci:**
  - **Podmínka stavu pohybu a výškové odchylky**: Regulace v osách X a Y je aktivována, pokud dron buď nevykazuje výraznější pohyb (rychlost nižší než 1 $m \cdot s^{-1}$) a současně má malou výškovou chybu (do 100 pixelů) aby se nejdříve dron vyrovnal s výškou nebo pokud se pohybuje rychlostí vyšší než 2 $m \cdot s^{-1}$. Tato podmínka zajistí, že bude reagovat na potřebu pohybu nebo korekce polohy pouze v situacích, kdy je to vhodné – buď při stabilním letu na nízké rychlosti nebo při zjevném pohybu.
  - **Podmínka natočení**: Pokud dron překročí definovaný úhel natočení γ (viz obrázku 4.9), čímž signalizuje náklon a rychlý dopředný pohyb,

zamezí se nepotřebné korekci výšky. To umožňuje dronu adaptivně reagovat na změny v jeho orientaci. V tomto případě je hraniční úhel pro aktivaci podmínky $\gamma >= 3°$.

- **Vzdálenostní podmínka**: Aktivace regulace je také závislá na tom, zda je minimální vzdálenost od cíle větší než požadovaná vzdálenost a také, jestli je dron v pohybu.

- **Reakce na splnění podmínek:**
  - Pokud je splněna alespoň jedna z výše uvedených podmínek a současně je zjištěna dostatečná vzdálenost od cíle (větší než $0,2\ m$) nebo je dron v pohybu, aktivují se PID regulátory pro osy X a Y. Tyto regulátory pak dynamicky upravují pohyb na základě aktuálních vzdálenostních chyb a orientace, což dronu umožňuje efektivně dosáhnout požadované polohy.
  - V situaci, kdy je dron velmi blízko svého cíle (chyba vzdálenosti menší než $0,2\ m$) a zároveň je téměř nehybný, dojde k resetování PID regulátorů pro osy X, Y a řídicí signály se nastaví na nulovou rychlost. Tento postup minimalizuje nadměrné korekce a oscilace v těsné blízkosti cíle.

Důsledky není-li splněna žádná podmínka:

- V případě, že žádná z podmínek pro aktivaci horizontální regulace není splněna, PID regulátory pro osy X a Y se automaticky resetují a řídicí signály pro tyto osy se nastaví na 0. Tento stav znamená, že dron zůstane v aktuální pozici nebo pokračuje v letu bez dalších korekcí v horizontálním směru. Tento přístup předchází zbytečným korekcím a udržuje stabilitu letu, pokud nejsou identifikovány žádné specifické potřeby pro změnu polohy.

Tento přístup k regulaci umožňuje dronu udržet optimální polohu a orientaci v prostoru s vysokou přesností a efektivitou, zároveň minimalizuje riziko nadměrných korekcí nebo oscilací. V rovnici řízení 4.6 je znázorněno, jak probíhá výpočet řídících signálů.

$$u(t) = K_p e(t) + K_i \int_0^t e_i(t)dt + K_d \frac{d}{dt} e(t) \tag{4.6}$$

Definujeme $e_i(t)$ podle vztahu

$$e_i(t) = e(t) \cdot K_i + (u_{t\_sat} - u(t)), \tag{4.7}$$

kde u_{t_sat} je saturace výstupu regulátoru definovaná jako

$$u_{t\_sat} = \begin{cases} u(t), & \text{pro } u_{\min} \leq u(t) \leq u_{\max} \\ u_{\max}, & \text{pro } u(t) > u_{\max} \\ u_{\min}, & \text{pro } u(t) < u_{\min} \end{cases}, \tag{4.8}$$

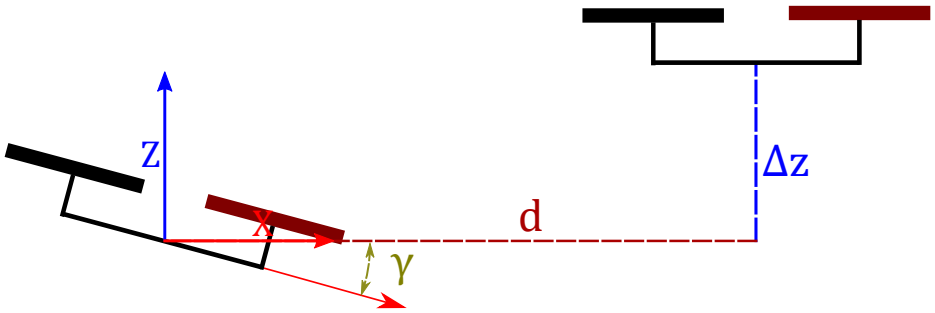
kde pro regulátory v horizontální rovině XY jsou rychlosti omezeny na $5,5\ m \cdot s^{-1}$ v obou směrech a ve vertikální platí omezení na $3,5\ m \cdot s^{-1}$ taktéž v obou směrech. Regulátory jsou implementovány v diskrétním algoritmu, který má fixní periodu 10 *ms*, takže poskytují akční zásahy jednou za tuto periodu. Vyhodnocení jednotlivých podmínek je také znázorněno v algoritmu 4.5.1.

Algorithm 1 Algoritmus pro navigaci dronu

```
if ¬finding then
    controlFind ← False
    // Resetovat regulátory pro hledání
    if pidFind_x.prev_error ∨ pidFind_y.prev_error ∨ pidFind_z.prev_error
then
        pidFind_x.reset()
        pidFind_y.reset()
        pidFind_z.reset()
    end if
    if phase ≠ −2 then
        phase ← −2
    end if
    // Zkontrolujte podmínky pro pohyb a orientaci
    if (¬isDroneMoving(1) ∧ |ΔZ| ≤ 100) ∨ γ ≤ 3 ∨ isDroneMoving(2) ∨ (d >
r ∧ isDroneMoving(0.3)) then
        if (|Δd| ≥ 0.2) ∨ isDroneMoving(1) then
            u_x ← pid_x.compute(Δd · cos(φ), dt)
            u_y ← pid_y.compute(Δd · sin(φ), dt)
        else
            pid_x.reset()
            pid_y.reset()
            u_x ← 0
            u_y ← 0
        end if
    else
        pid_x.reset()
        pid_y.reset()
        u_x ← 0
        u_y ← 0
    end if
    if |ΔZ| ≥ 20 ∨ ¬(γ ≤ 7) then
        u_z ← pid_z.compute(ΔZ, dt)
    else
        pid_z.reset()
        u_z ← 0
    end if
end if
```

4.5.2 Regulace vertikální vzdálenosti

Ilustrace 4.9 demonstruje proměnné použité pro výpočet regulace ve vertikální rovině. V rámci vertikální regulace dronu je použit PID regulátor a Δz představuje rozdíl mezi pozicemi zobrazenými zeleným a červeným bodem na obrázku 4.5 v y-rovině snímku kamery. Tato hodnota je využívána jako vstupní veličina pro regulátor vertikální polohy, která je matematicky popsána rovnicí $\Delta z = y_{imgCenter} - y_{LimgCenter}$.



Obrázek 4.9: Zobrazení parametrů v rámci regulace dronů v rovině XZ

Jelikož tato odchylka může vzniknout při náklonu sledujícího dronu, například když dron prudce zrychlí, může to vést k falešným signálům pro regulaci výšky. Je tedy nezbytné tyto scénáře filtrovat pomocí následujících podmínek. Vertikální regulace je postavena na PID regulátoru, který byl popsán v rovnici 4.6.

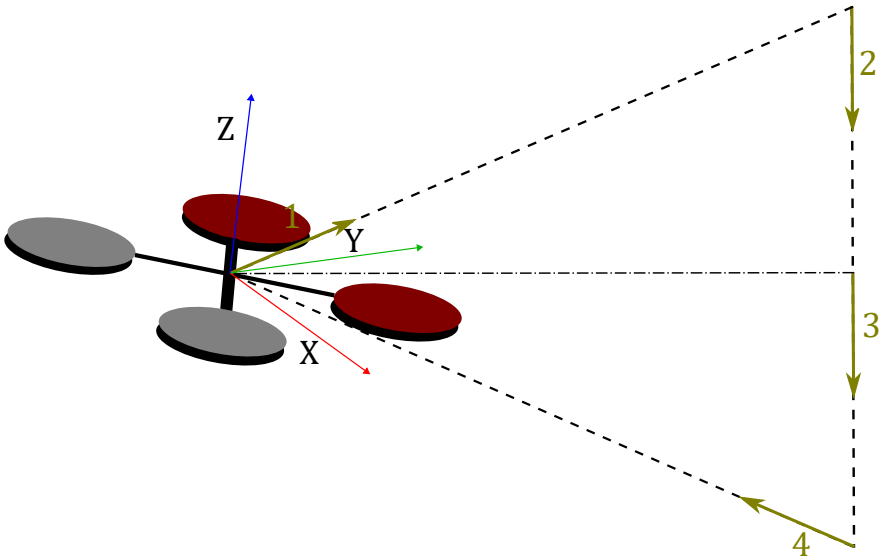
- **Podmínky pro aktivaci vertikální regulace:**
  - **Výšková odchylka**: Vertikální regulace je aktivována, pokud je absolutní hodnota výškové chyby $\Delta z \geq 20px$ obrazu kamery. Toto nastavení zaručuje, že regulace se spustí pouze v případě značných odchylek, což je důležité pro minimalizaci drobných oscilací.
  - **Natočení dronu**: Regulace je také aktivována, pokud úhel γ nepřekročí zvolený práh 7°. Tato podmínka zamezí regulaci výšky pokud dron zrychluje a snaží se minimalizovat vzdálenost.

4.6 Nalezení ztraceného dronu

Lokalizace představuje fundamentální prvek systému řízení. Jedním z nejdůležitějších faktorů v tomto procesu je použití vyhledávacího schématu, které je navrženo tak, aby dron mohl metodicky prozkoumávat prostředí a efektivně lokalizovat vedoucí dron. Vyhledávání probíhá v několika fázích, ve kterých v průběhu algoritmus přechází do módu vyhledávání a aktivuje se hledání podle předdefinovaného vzoru. Dronem sledovaný vzor je navržen jako série pohybů tvořících trojúhelníkový tvar,

vzor je zvolen jelikož lze efektivně pokrýt rozsáhlou oblast během hledání, což zvyšuje pravděpodobnost detekce cíle. Dron během hledání nepřetržitě rotuje kolem své osy ve směru ztraceného dronu. Procedura po ztrátě z vizuálního pole je následující:

1. **Pokračování v letu**: Dron začíná s poloviční rychlostí pohybu ve směrech X a Y, což zajišťuje aby dron pokračoval ve stabilním letu. Tato fáze trvá přibližně 1,5 sekundy aby bylo možné okamžité vysledování v případech krátkodobé ztráty.
2. **Rotace a Pozorování**: V této fázi dron zůstává na místě a provádí pouze rotaci kolem své osy Z, což mu umožňuje provádět 360° sken oblasti rychlostí 1 rad/s ve směru ztraceného dronu. Tento krok trvá přibližně 6 sekund a po tuto dobu dron hledá ve stejné rovině. V této fázi také proběhne aktivace módu vyhledávání, který potvrzuje, že dron je ztracen a nelze ho nalézt na své výškové úrovni. Tato funkce je více rozebrána v sekci 4.6.1.
3. **Trojúhelníkové schéma**: Pokud nebyla úspěšná lokalizace v předešlé fázi dron začne provádět sérii pohybů vytvářejících trojúhelníkový vzor. Tento vzor zahrnuje diagonální pohyb se změnou výšky tak, aby schéma bylo tvořeno rovnoramenným trojúhelníkem a ošetřením vůči havarování zobrazené na obrázku 4.10, kde každý segment trvá 12 s. Každá strana, respektive fáze, trojúhelníku jsou provedeny během fixně stanoveného časového intervalu (přibližně 12 sekund), po kterém dron mění svou pozici a směr podle předem definované sekvence.



Obrázek 4.10: Vizualizace trojúhelníkového schéma hledání

Díky tomuto řešení, lze dron efektivně dohledat a navrátit se na požadovanou trajektorii.

4.6.1 Aktivace módu hledání

Po vizuální ztrátě vedoucího dronu a dosažení 2. fáze hledání, se řízení dostává do módu hledání, které spočívá v odlišné korekci vzdáleností než u je vysvětleno v sekci 4.5.1 po zpětném vizuálním kontaktu. Předtím než je navozen standardní režim řízení při znovunalezení, je dron omezen rychlostí ve všech osách tak, aby došlo ke srovnání vzdáleností pomocí určitých podmínek.

Po vizuálním lokalizování vedoucího dronu se sledující dron nejprve natočí maximální rychlostí $1,6\ rad/s$, aby měl vedoucí dron uprostřed svého vizuálního pole, což mu umožní udržet zaměření po dobu 3 s pro stabilní sledování. Následně upraví svou výšku maximální rychlostí $3\ m \cdot s^{-1}$, aby se dostal na stejnou výškovou úroveň jako vedoucí dron a nebyl ztracen kontakt. Jakmile jsou tyto dvě podmínky splněny, dron začne regulovat horizontální vzdálenost s maximální rychlostí $3,5\ m \cdot s^{-1}$ v každé ose, čímž se minimalizují prudké pohyby a zajišťuje se plynulé sledování vedoucího dronu. Pokud se vzdálenostní chyba sníží na $0,5\ m$ nebo méně, dron přechází do standardního režimu řízení. Řízení použité v módu hledání je prezentováno v algoritmu 4.6.1.

Algorithm 2 Řídicí Algoritmus pro nalezení dronu

```
if finding then
    // Jestliže byl dron ztracen příliš dlouho, nastaví se optimální hledání a regulace
    if pid_x.prev_error ∨ pid_y.prev_error ∨ pid_z.prev_error then
        pid_x.reset()
        pid_y.reset()
        pid_z.reset()
    end if
    now ← this->now()
    elapsedTime ← (now − lastDepthUpdateTimeLateral_).seconds()
    if ΔX_img ≥ 50 ∧ ¬controlFind then
        lastDepthUpdateTimeLateral_ ← now
    end if
    if elapsedTime ≥ 3 then
        controlFind ← True
        u_z ← pidFind_z.compute(ΔZ, dt)
        if |ΔZ| ≤ 50 ∨ isDroneMoving(0.3) then
            u_x ← pidFind_x.compute(Δd · cos(φ), dt)
            u_y ← pidFind_y.compute(Δd · sin(φ), dt)
        end if
    else
        u_z ← 0
    end if
    if Δd = 0.5 then
        finding ← false
    end if
end if
```

4.7 Uživatelské rozhraní

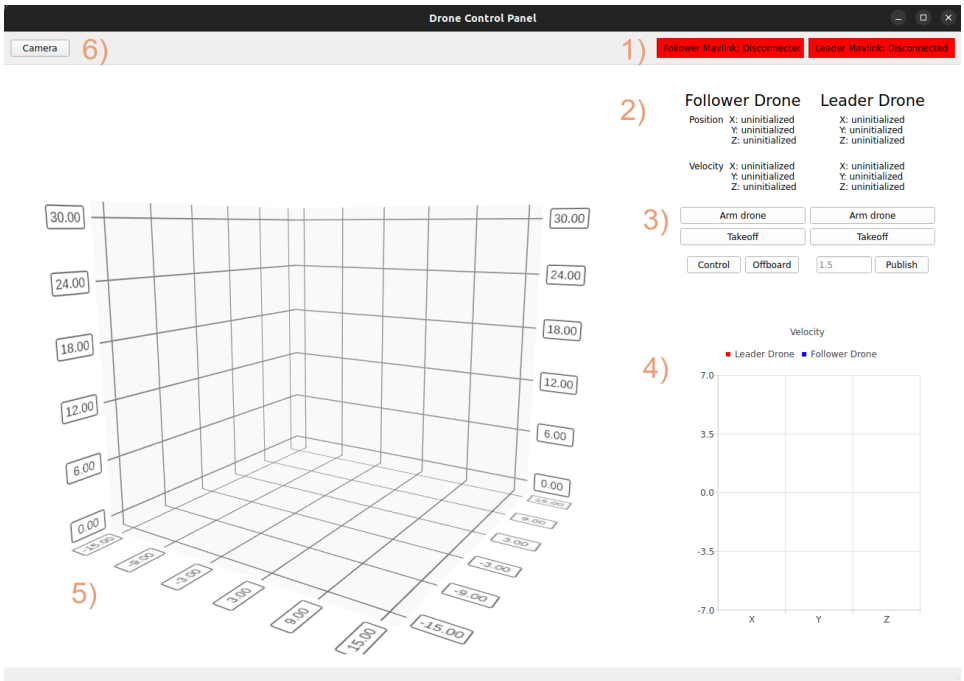
Rozhraní je naprogramováno pomocí Qt [24], což je rozsáhlá vývojová platforma pro programovací jazyk C++. Qt umožňuje tvorbu multiplatformních aplikací s uživatelským rozhraním a zahrnuje knihovny pro 2D a 3D grafiku, síťování a databázové operace. Tato platforma je oblíbená díky své vysoké úrovni abstrakce, která usnadňuje vývoj komplexních aplikací, a mechanismu signálů a slotů pro událostmi řízené programování. Díky své flexibilitě a širokému spektru funkcí najde Qt uplatnění v řadě průmyslových odvětví, od softwaru pro desktopové aplikace až po zařízení IoT a automobilový průmysl.

Aplikace se skládá ze tří základních částí. První a nejdůležitější částí je vizualizace letového provozu prostřednictvím grafu Q3DScatter, kde jsou souřadnice obou dronů zobrazeny pomocí barevně rozlišených bodů. Jednotlivé drony jsou

navíc rozlišitelné díky kruhovému prstenci kolem sledovaného dronu, který reprezentuje požadovanou vzdálenost pro regulaci. Tato vizualizace umožňuje neustálé sledování letového provozu a poskytuje uživateli přehledný obraz o jejich aktuálních polohách a vzájemné vzdálenosti.

Druhou část tvoří informativní tabulka, která poskytuje přesné informace o pozicích a rychlostech jednotlivých dronů. Kromě toho tabulka obsahuje dvě ovládací tlačítka pro každý dron, sloužící k aktivaci motorů (označované jako "Arm"). Dalším tlačítkem je možnost vydat příkaz pro zahájení letu. Navíc tabulka obsahuje různé specifické ovládací prvky pro každou kvadrokoptéru. Pro sledující dron jsou přidána dvě tlačítka pro aktivaci a deaktivaci řízení, což je zásadní pro zásah v případě neobvyklého chování nebo pro zahájení běžného provozu. Dalším tlačítkem je přepnutí do offboard módu, což umožňuje použít řízení pomocí externího algoritmu. Dále je přidáno textové pole s tlačítkem, které umožňuje nastavování vzdálenosti pro regulaci.

Třetí část aplikace obsahuje sloupcový graf, který vizualizuje rychlosti dronů v jednotlivých osách. Tento graf umožňuje uživatelům sledovat a hodnotit kvalitu řídicího algoritmu a zjišťovat nepravidelné jevy, které mohou indikovat chyby v řízení. V horní části aplikace se nachází další funkce, jako jsou indikátory připojení obou kvadrokoptér a tlačítko pro zobrazení kamerového záznamu sledujícího dronu. Ve spodní části aplikace jsou umístěny zprávy informující o úspěchu nebo neúspěchu prováděných operací, například o vzletu dronu.



Obrázek 4.11: Uživatelské rozhraní

1. **Stav připojení Mavlink:** Tato oblast zobrazuje aktuální stav připojení pro oba drony (follower a leader). Barva textu a pozadí se změní podle stavu připojení: červená pro odpojení a zelená pro připojení.

2. **Panel s informacemi o dronu:** Zde jsou zobrazeny detailní informace o polohách a rychlostech obou dronů (follower a leader). Tyto informace se aktualizují v reálném čase, jak aplikace přijímá data z ROS.

3. **Ovládací panel dronu:** Panel obsahuje tlačítka pro základní ovládání dronů. Detailní popis tlačítek:

   Arm drone: Aktivace dronů pro přípravu k letu.

   Takeoff: Automatický vzlet dronu do přednastavené výšky.

   Control: Aktivace nebo deaktivace řídicího systému pro follower dron.

   Offboard: Nastavení offboard režimu pro plně programovatelné ovládání dronu.

   Textové pole: Zadání vzdálenosti pro regulaci mezi leader a follower dronem.

   Publish: Publikování nastavené vzdálenosti do řídicího systému dronu.

4. **Graf rychlosti:** Vizualizace rychlosti dronů v osách X, Y, Z.

5. **3D Scatter Plot:** Trojrozměrná vizualizace polohy dronů v prostoru.

6. **Tlačítko kamery:** Otevření okna pro zobrazení živého obrazu z kamery dronu.

Experimenty 5

Tato kapitola se věnuje prezentaci experimentů zaměřených na ověření funkčnosti a efektivity řídícího algoritmu dronů v simulačním prostředí. Cílem experimentů bylo podrobně testovat, jak řídící algoritmus reaguje na různé předem definované scénáře a s jakou efektivitou byly řešeny. Experimenty byly navrženy tak, aby otestovaly veškeré funkce řízení prezentované v kapitole 4, jako je držení požadované vzdálenosti, výšky nebo nalezení ztraceného dronu.

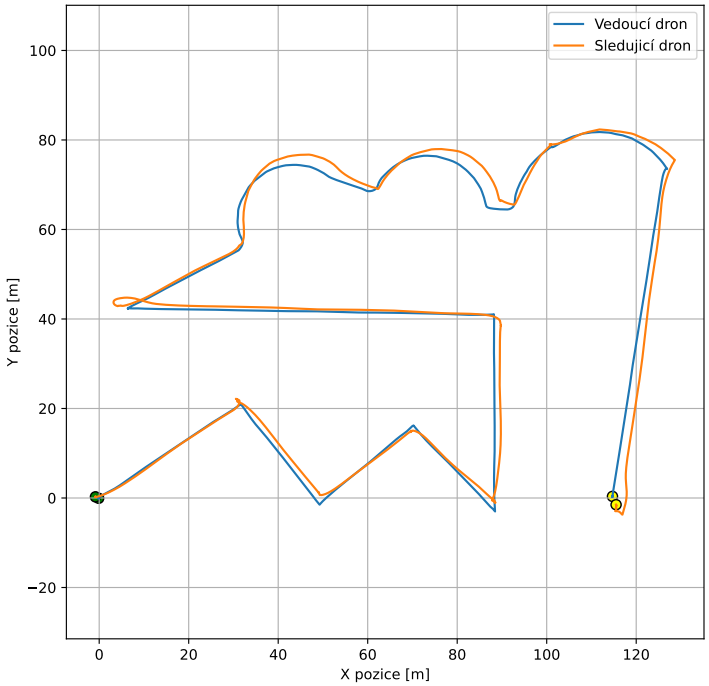
Konkrétně byly zvoleny dva experimenty, kde v prvním je testována dynamika letu, kde se vedoucí dron snaží co nejrychleji přemisťovat mezi danými body s mírnou změnou výšky. Naopak u experimentu druhého, vedoucí dron sleduje dvě kruhové trajektorie s výškovým rozdílem tak, aby došlo k vizuální ztrátě dronu.

5.1 Dynamický let

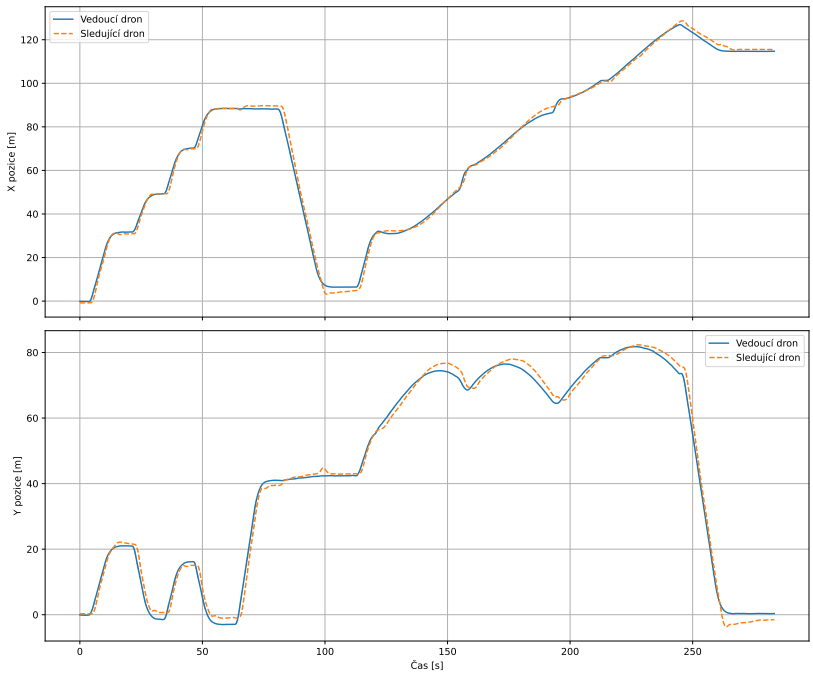
Tento testovací let byl zvolen tak, aby otestoval chování řízeného dronu při možných reálných aplikacích. Zaměřuje se na několik částí, kterými jsou bodové plánování mise, tedy vedoucí dron má předem stanovené body do kterých se snaží dostat za co nejkratší čas a krátké sledování trajektorie s malým výškovým rozdílem.

Celý průběh plánované trajektorie je zobrazen na grafu roviny XY na obrázku 5.1. Drony startují ze zeleně označených bodů v počátku. Vedoucí dron následně sleduje měnící se diagonální trajektorii se čtyřmi zastávkami, kde na konci této trajektorie můžeme pozorovat, jak sledující dron zvládá těsný průlet vedoucího drona kolem něj, což si vyžádalo obrat o 180° za neustálé korekce vzdálenosti. Následuje další série zastávek podél obdélníkové trajektorie, kde na jejím konci vedoucí dron výrazně zpomaluje a sledující dron se vyhýbá kolizi, obchází ho a vrací se do požadované vzdálenosti od něj. Z této obdélníkové trajektorie vedoucí dron přechází k sledování kruhových trajektorií s proměnlivou výškou mezi nimi, kde si řídicí systém s touto částí poradil bez problémů, což platí i pro poslední úsek trajektorie. Celkově lze pozorovat, že sledující dron přesně kopíruje trajektorii vedoucího dronu.

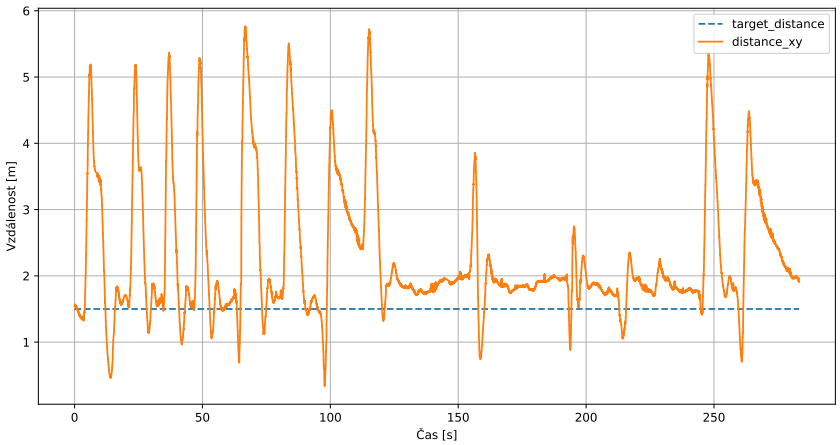
Graf prezentován na obrázku 5.2 ukazuje detailní průběh jednotlivých os v čase, kde je viditelné přesné sledování trajektorie a hladký průběh regulace. V čase 100 *s* lze vidět zmíněné vyhnutí kolizi na konci obdélníkové trajektorie.



Obrázek 5.1: Vizualizace trajektorie obou dronů v rovině XY

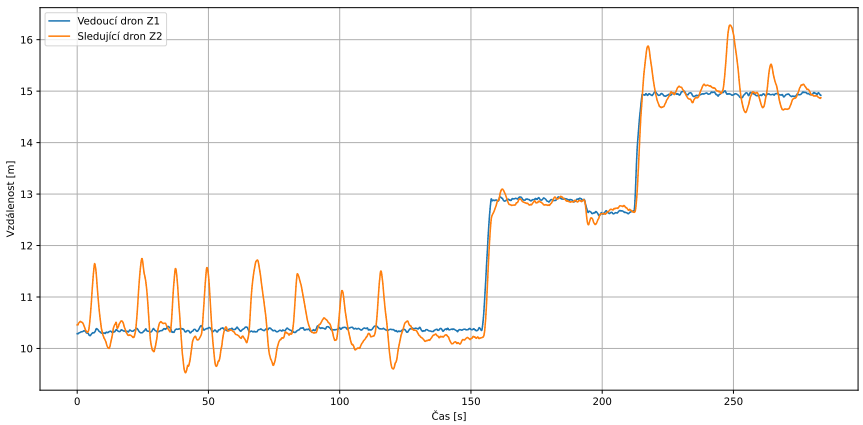


Obrázek 5.2: Vizualizace trajektorie v čase osy X a Y dronů



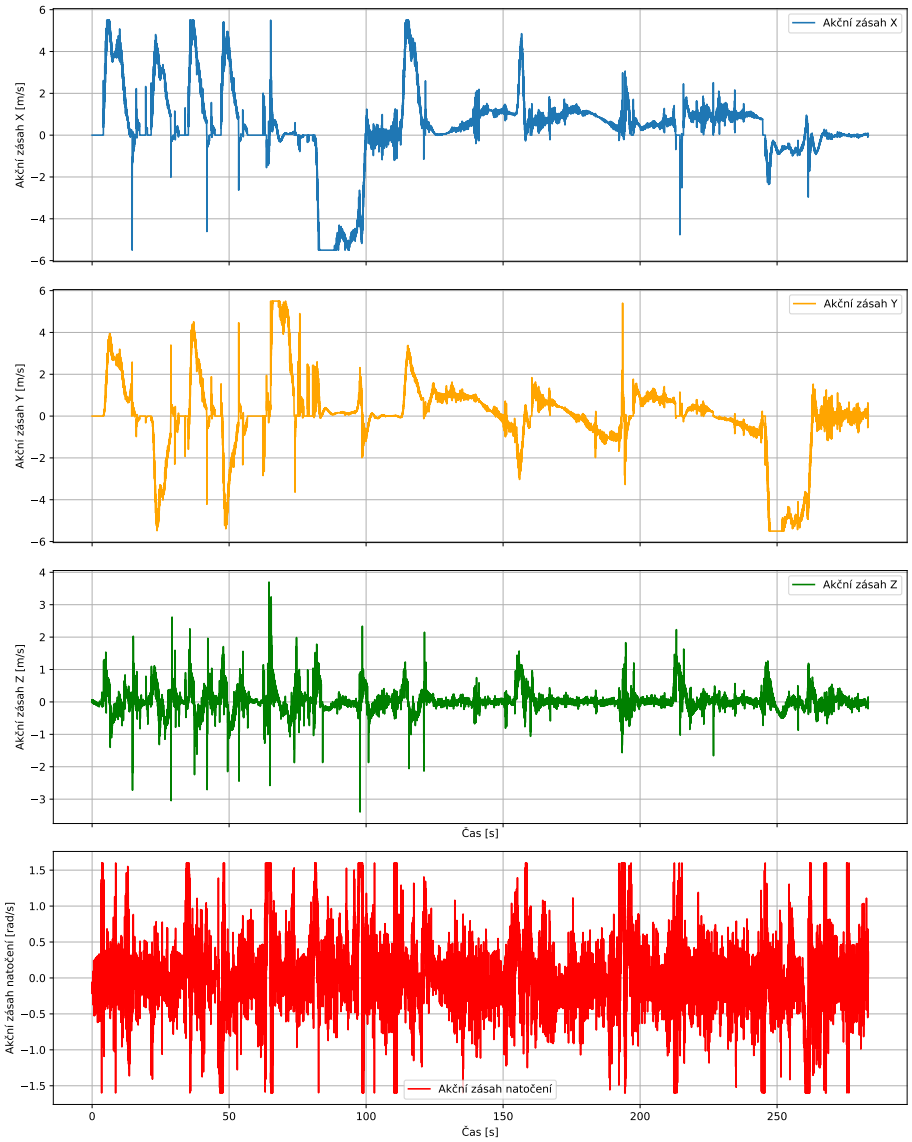
Obrázek 5.3: Vizualizace vektorové vzdálenosti dronů

Obrázek 5.3 vizualizuje vektorovou vzdálenost mezi drony v čase. Jsou zde viditelné jednotlivé úseky v podobě velkých výchylek, kde se vedoucí dron po krátkém zastavení přesouvá k dalšímu bodu. Tyto výchylky jsou způsobeny rychlostí pohybu vedoucího dronu mezi body. Přestože jsou tyto výchylky poměrně vysoké, jsou si mezi sebou amplitudově i průběhové podobné, takže dron reaguje na tyto situace předvídatelně s rychlou korekcí vzdálenosti. Co se týče kruhové trajektorie, nechává si sledující dron menší prostor od požadované vzdálenosti, tak aby mohl rychle zareagovat v případné změně trajektorie.



Obrázek 5.4: Vizualizace obou výšek dronů

Letové výšky obou dronů v čase jsou zobrazeny na obrázku 5.4, kde lze pozorovat stejný počet téměř identických výchylek jako v předchozím grafu, což je způsobeno změnami polohy vedoucího dronu mezi jednotlivými body. Takovéto chování je žádoucí, protože při rychlé změně polohy vedoucího dronu si sledující dron zvýší svou výšku, aby získal lepší přehled o směru letu vedoucího dronu, než se k němu přiblíží na požadovanou vzdálenost. V čase 150 *s* vedoucí dron začíná upravovat svou požadovanou výšku a následovně pokračuje v letu podél kruhové trajektorie, kde v obou těchto případech došlo k rychlé reakci a okamžité korekci výšky, což umožnilo perfektní sledování. Nakonec lze pozorovat několik menších odchylek ve výšce, které byly způsobeny změnou trajektorie vedoucího dronu.



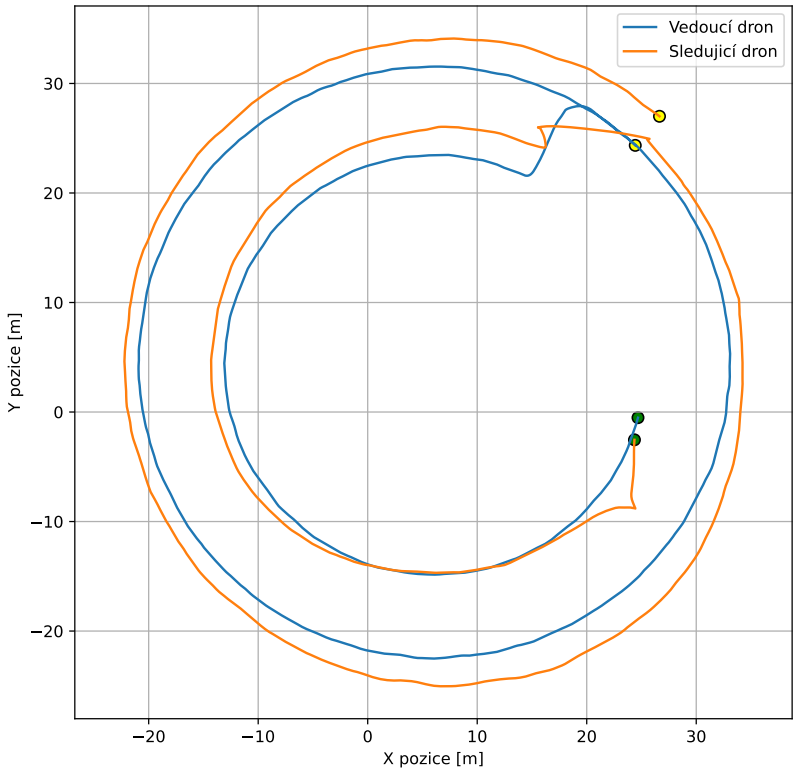
Obrázek 5.5: Akční zásahy regulátorů

Přehled akčních zásahů, lze pozorovat v grafu 5.5, kde jsou vidět požadované korekce rychlostí v jednotlivých osách X, Y, Z a natočení ve vertikální rovině, předávané řízeným systémem do autopilotu. Jelikož konstanty PID regulátoru byly voleny experimentálně, lze pozorovat velké akční zásahy při požadovaném natočení, kde se regulátor často dostává do saturace a reaguje chaoticky oproti ostatním regulátorům. Toto lze vyřešit vhodnějším zvolením konstant nebo použít jednu ze zmíněných metod pro nastavení regulace v sekci 4.5.1.

5.2 Let po kružnici

Experiment se zabývá schopností dronu sledovat pevně danou trajektorii a funkčností systému pro opětovné nalezení ztraceného dronu. Specificky je testována reakce dronu na trajektorii tvořenou dvěma kružnicemi, které jsou od sebe odděleny významným výškovým rozdílem. Tento výškový rozdíl je klíčový, neboť vede k situaci, kdy sledující dron ztratí vizuální kontakt s vedoucím dronem.

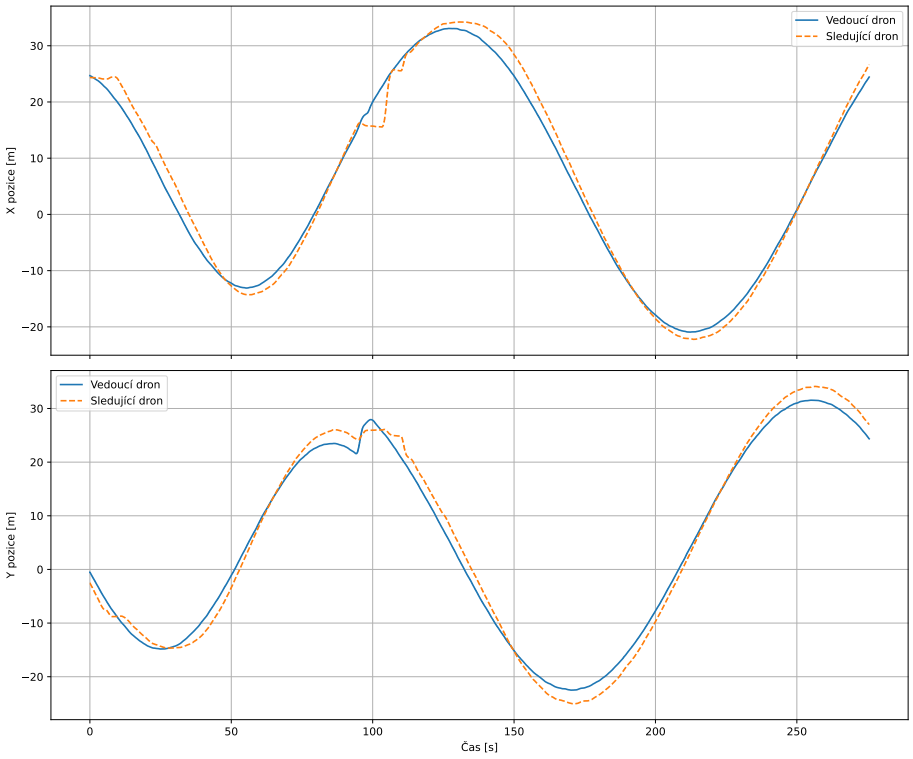
Graf na obrázku 5.6, znázorňuje pohyb dronů v rovině X a Y.



Obrázek 5.6: Vizualizace trajektorie obou dronů

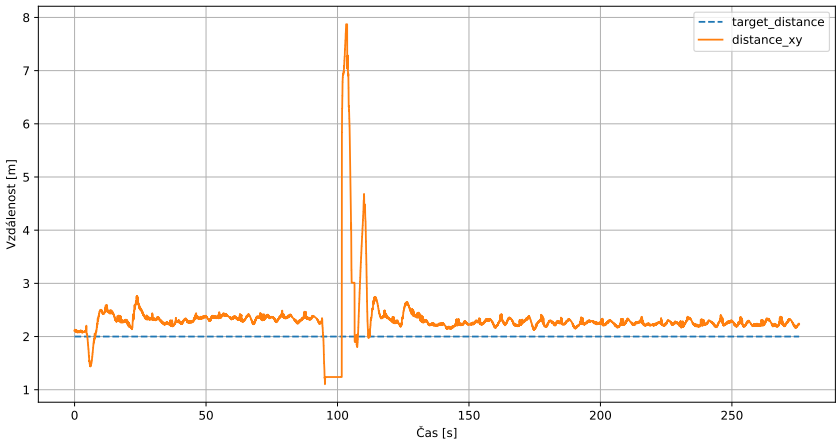
Počátek letu je vyznačen zelenými body, u nichž lze pozorovat, že dron ihned po startu experimentu odbočil z plánované trajektorie. Toto odchýlení bylo záměrné, protože došlo k rychlé reakci na potenciální kolizi, přičemž sledující dron si během celého manévru udržoval vizuální kontakt s vedoucím dronem. Experiment následně pokračoval po kruhové trajektorii s konstantním sledováním cíle, ale během přechodu mezi jednotlivými kružnicemi došlo u sledujícího dronu k nepravidelnému chování kvůli dočasné ztrátě vizuálního kontaktu, což vyústilo v aktivaci režimu hledání. Po úspěšném znovunalezení cíle dron znovu navázal na původní trajektorii a sledování proběhlo bez dalších problémů až do konečného ukončení experimentu označeného žlutým bodem.

Reprezentace hodnot získaných v čase na osách X a Y je zobrazena na obrázku 5.7. Na tomto grafu můžeme pozorovat plynulý průběh sledování, který je přerušen pouze dvěma nepravidelnostmi, jejichž vliv je vysvětlen v předchozím odstavci.



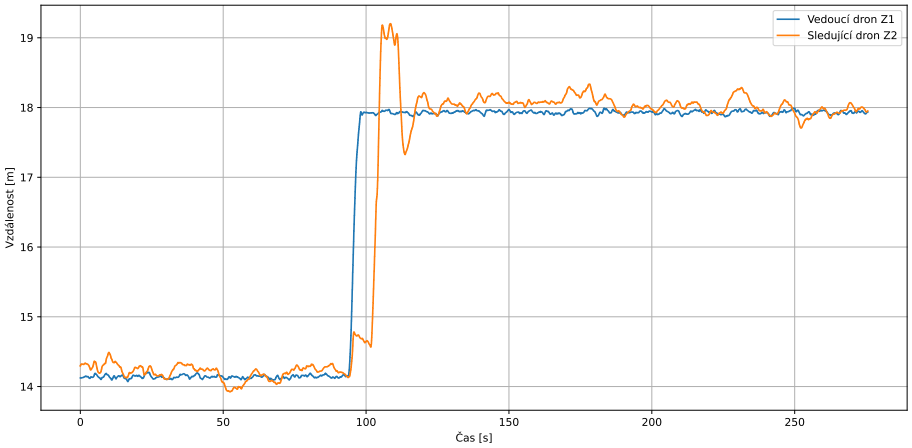
Obrázek 5.7: Vizualizace trajektorie v čase osy X a Y dronů

Vektorová vzdálenost mezi dorny na obrázku 5.8 je téměř neměnná pokud se jedná o sledování sledování kruhové trajektorie. Přechod mezi nimi je však velmi znatelný, zejména v počátku nebo v momentě vizuální ztráty vedoucího dronu ve 200. sekundě. Navzdory výraznému vychýlení vzdálenosti a aktivace funkce hledaní, dron byl schopen vyhledat svůj cíl a včasně se k němu navigovat.



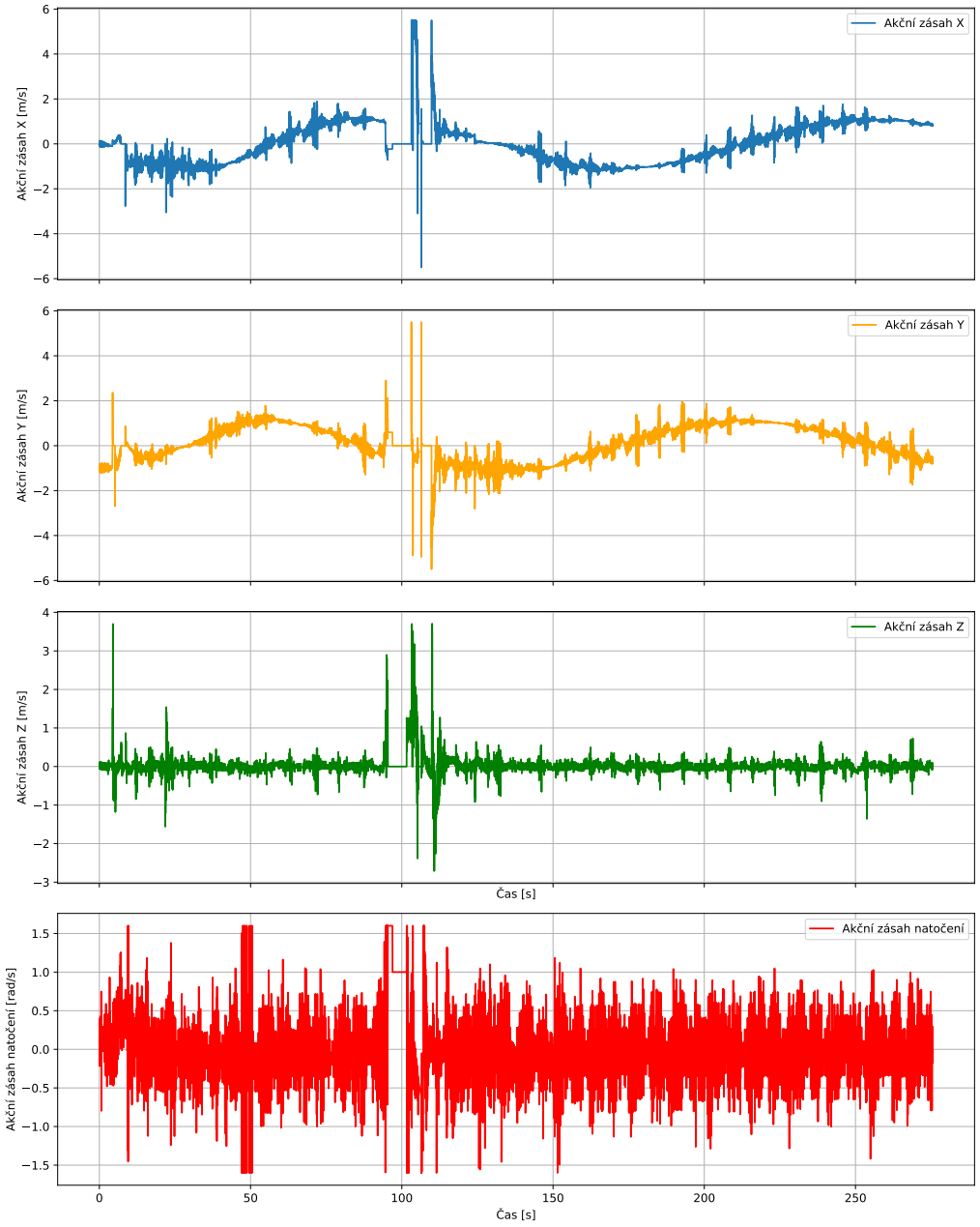
Obrázek 5.8: Vizualizace vektorové vzdálenosti dronů

Na obrázku 5.9 je prezentováno porovnání letových výšek, kde jediná nesrovnalost v regulaci výšky, vzniká při ztrátě vedoucího dronu. Nedošlo však k velkému překmitu kde se dron zastavil a po úspěšném znovunalezení vedoucího dronu a srovnání natočení, opět začal regulovat výšku letu na požadovanou hodnotu.



Obrázek 5.9: Vizualizace obou výšek dronů

Na posledním grafu 5.10 jsou vizualizovány akční zásahy regulátorů. V osách X a Y můžeme pozorovat velmi jasně kruhový průběh trajektorie bez výraznějších kmitů, tedy kromě fáze, kdy došlo k vizuální ztrátě a vyhnutí kolize. Obdobný případ vzniká v ose Z, kde je potřeba regulace výšky jen v případě hledání vedoucího dronu. Poslední veličinou je natočení kolem osy Z, zde platí totožné jako pro předešlí experiment 5.1.



Obrázek 5.10: Akční zásahy regulátorů

Závěr 6

V této bakalářské práci byl prozkoumán a implementován komplexní algoritmus pro sledování dronů v podobě leader-follower s využitím technologií ROS2 a PX4 Autopilot, zahrnující simulační prostředí Gazebo Garden. Hlavním cílem bylo vyvinout a experimentálně otestovat řídící algoritmus sledujícího dronu tak, aby dokázal efektivně následovat vedoucí dron i v krizových podmínkách. Nejen že se dron snaží držet v definované vzdálenosti od vedoucího dronu, ale dokáže ho i nalézt pomocí funkce vyhledávání pokud dojde k vizuální ztrátě.

Výsledky ukazují, že navržený systém umožňuje přesné sledování a korekci vzdálenosti při dynamickém letu, a to jak při sledování bodové trajektorie, tak i kruhové trajektorie, jak je uvedeno v sekcích 5.1 a 5.2. Při sledování dochází téměř k dokonalému opisu trajektorie s malým rozdílem vzdáleností tak aby sledující dron měl prostor na potřebnou reakci. Ukázalo se tedy, že systém je robustní i ve složitějších scénářích, kdy může docházet k náhlým změnám v prostředí nebo v chování vedoucího dronu. Systém efektivně řeší potenciální kolize, což zvyšuje bezpečnost a autonomii dronu. Navíc, díky použití open-source technologií, je systém dobře přizpůsobitelný a může být dále rozšířen o další funkcionality nebo integraci s jinými systémy. Zdrojový kód použitý v této práci je dostupný na adrese https://github.com/TheJamesq/PX4_leader-follower_byCameraDetection.

Pro budoucí práci je možné systém dále optimalizovat, přičemž je vhodné se zaměřit na detekci UAVs z barevného obrazu z hlediska reálné aplikace. Dále je možné využití umělé inteligence pro automatizaci rozhodovacích procesů a pro zlepšení adaptivních schopností systému v reálném čase. Také by bylo vhodné identifikovat různé matematické modely dronu a přizpůsobit tak PID regulátory pro optimální řízení.

Bibliografie

1. MALLICK, Tuton Chandra; BHUYAN, Mohammad Ariful Islam; MUNNA, Mohammed Saifuddin. Design implementation of an UAV (Drone) with flight data record. In: *2016 International Conference on Innovations in Science, Engineering and Technology (ICISET)*. 2016, s. 1–6. Dostupné z DOI: 10.1109/ICISET.2016.7856519.
2. GHAMRY, Khaled A.; ZHANG, Youmin. Formation control of multiple quadrotors based on leader-follower method. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015, s. 1037–1042. Dostupné z DOI: 10.1109/ICUAS.2015.7152394.
3. BACCO, M.; BARSOCCHI, P.; FERRO, E.; GOTTA, A.; RUGGERI, M. Drones for Smart Agriculture: A Technical Report. *IEEE Access*. 2019, roč. 7, s. 27519–27526. Dostupné z DOI: 10.1109/ACCESS.2019.2900035.
4. WAHARTE, Sonia; TRIGONI, Niki. Supporting Search and Rescue Operations with UAVs. In: *2010 International Conference on Emerging Security Technologies*. 2010, s. 142–147. Dostupné z DOI: 10.1109/EST.2010.31.
5. AUSTIN, Reg. *Unmanned Aircraft Systems: UAVs Design, Development and Deployment*. Wiley, 2010. ISBN 978-0470058190.
6. WALTER, Viktor; STAUB, Nicolas; FRANCHI, Antonio; SASKA, Martin. UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs. *IEEE Robotics and Automation Letters*. 2019, roč. 4, č. 3, s. 2637–2644. Dostupné z DOI: 10.1109/LRA.2019.2901683.
7. ZHOU, Xin et al. Swarm of micro flying robots in the wild. *Science Robotics*. 2022, roč. 7, č. 66, eabm5954. Dostupné z DOI: 10.1126/scirobotics.abm5954.
8. ROBOTICS, Open. *ROS 2 Documentation* [https://docs.ros.org/en/humble/index.html]. [B.r.]. Accessed: 2024-03-05.
9. MACENSKI, Steven; FOOTE, Tully; GERKEY, Brian; LALANCETTE, Chris; WOODALL, William. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*. 2022, roč. 7, č. 66, eabm6074. Dostupné z DOI: 10.1126/scirobotics.abm6074.

10. THOMAS, Dirk. *ROS 2 Middleware Interface* [https://design.ros2.org/articles/ros_middleware_interface.html]. [B.r.]. Accessed: 2024-03-06.

11. MEIER, Lorenz; HONEGGER, Dominik; POLLEFEYS, Marc. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, s. 6235–6240. Dostupné z DOI: 10.1109/ICRA.2015.7140074.

12. KOUBÂA, Anis et al. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access*. 2019, roč. 7, s. 87658–87680. Dostupné z DOI: 10.1109/ACCESS.2019.2924410.

13. KUBÍČEK, Karel; ČECH, Martin; ŠKACH, Jan. Continuous enhancement in model-based software development and recent trends. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, s. 71–78. Dostupné z DOI: 10.1109/ETFA.2019.8869237.

14. TEAM, PX4 Development. *PX4 Autopilot SITL Simulation Environment* [https://docs.px4.io/main/en/simulation/]. 2024. Accessed: 2024-03-12.

15. TEAM, PX4 Development. *PX4 Autopilot Software Documentation* [https://docs.px4.io/main/en/]. 2024. Accessed: 2024-03-12.

16. HOLYBRO. *PX4 Development Kit X500 v2* [https://docs.holybro.com/drone-development-kit/px4-development-kit-x500v2]. [B.r.]. Accessed: 2024-03-12.

17. TEAM, PX4 Development. *PX4 AutopilotSimulation X500 Quadrotor with Depth Camera* [https://docs.px4.io/main/en/sim_gazebo_gz/vehicles.html]. 2024. Accessed: 2024-03-13.

18. MAVLINK. *MAVLink extendable communication node for ROS* [https://github.com/mavlink/mavros]. 2015.

19. KOENIG, N.; HOWARD, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. 2004, sv. 3, 2149–2154 vol.3. Dostupné z DOI: 10.1109/IROS.2004.1389727.

20. TEAM, QGroundControl Development. *QGroundControl User Guide* [https://docs.qgroundcontrol.com/master/en/qgc-user-guide/fly_view/fly_view.html]. [B.r.]. Accessed: 2024-03-18.

21. TEAM, QGroundControl Development. *Screenshot from QGroundControl User Guide* [https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html]. [B.r.]. Accessed: 2024-03-18.

22. *The OpenCV Reference Manual*. 2.4.9.0. vyd. Itseez, 2024.

23. ŽÁN, Vilém; KUBÍČEK, Karel; ČECH, Martin. Design of robust PI controller by combining robustness regions with time-domain criteria. In: *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2022, s. 1–8. Dostupné z DOI: 10.1109/ETFA52439.2022.9921544.

24. THE QT COMPANY. *Qt Introdutcion* [online]. 2024. [cit. 2024-04-04]. Dostupné z: https://wiki.qt.io/About_Qt.

Seznam obrázků