VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Michal Panský

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

LOKALIZACE A NAVIGACE BEZPILOTNÍHO PROSTŘEDKU TELLO
LOCALIZATION AND NAVIGATION OF TELLO UNMANNED AIRCRAFT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Michal Panský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Gábrlík**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Michal Panský ***ID:*** 203313
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Lokalizace a navigace bezpilotního prostředku Tello

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je seznámit se s miniaturním bezpilotním prostředkem Ryze Tello a otestovat jeho možnosti lokalizace a navigace ve vnitřním prostředí.

1. Seznamte se s bezpilotním prostředkem Ryze Tello a jeho oficiálním i alternativními SDK.
2. Zaintegrujte Tello do ROS (Robot Operating System) tak, aby jej bylo možné z tohoto prostředí ovládat, vyčítat data ze senzorů a vizualizovat provoz.
3. Zprovozněte dostupnou knihovnu pro detekci AR tagů pomocí palubní kamery a na jejich základě vytvořte algoritmus pro lokalizaci Tella ve vnitřním prostředí. Zaměřte se na robustnost lokalizace (fúze dat z více tagů, různých senzorů apod.)
4. Navrhněte algoritmus pro navigaci ve vnitřním prostředí zahrnující poziční stabilizaci a let na definované souřadnice.
5. Realizované řešení lokalizace a navigace otestujte a vyhodnoťte.

DOPORUČENÁ LITERATURA:

PYO, YoonSeok, HanCheol CHO, RyuWoon JUNG a TaeHoon LIM. ROS Robot Programming. Republic of Korea: ROBOTIS Co., 2017. ISBN 979-11-962307-1-5.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Petr Gábrlík

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

UPOZORNĚNÍ:
Autor bakalářské práce nesmí při vytváření bakalářské 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

Tato bakalařská práce se zabývá návrhem vhodného řešení ovládání bezpilotního letadla Ryze Tello při použití pro autonomní mise. Jejím obsahem je seznámení se a otestování všech dostupných SDK a veškerých funkcí, které nabízejí. Seznamení s lokalizací a navigací robotů v uzavřených prostorech. Dále je popsan ROS a jeho implementace pro daný problém využitím dostupných SDK s návodem pro instalaci a spuštění praktické části. V závěru práce jsou zobrazeny data získaná během letu bezpilotního letadla a struktura komunikačního systému, popis problémů, které se vyskytly v průběhu testování a diskuze ohledně možného rozšíření tohoto řešení.

KLÍČOVÁ SLOVA

Dron Ryze Tello, autonomní mise, bezpilotní letadlo, ROS, TelloPy

ABSTRACT

This bachelor thesis deals with the design of a suitable solution to control the unmanned aircraft Ryze Tello for use in autonomous missions. Its content is to familiarize and test all available SDKs and all the features they offer. Introduction to the location and navigation of robots in indoor spaces. Then there is described ROS and its implementation for the problem using the available SDK with instructions for installation and launch of the practical part. At the end of the work are displayed data obtained during the unmanned aircraft flight and the structure of the communication system, a description of the problems encountered during testing and a discussion of possible extension to this solution.

KEYWORDS

Drone Ryze Tello, autonomnous missions, unmanned aerial vehicle, ROS, TelloPy

PANSKÝ, Michal. *Autonomní bezpilotní prostředek Tello*. Brno, 2020, 57 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky. Vedoucí práce: Ing. Petr Gábrlík

Vysázeno pomocí balíčku `thesis` verze 3.05; `http://latex.feec.vutbr.cz`

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Autonomní bezpilotní prostředek Tello“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petrovi Gábrlíkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Bezpilotní letadla UA (Bezpilotní letadlo – Unmanned aircraft), též označovaná jako drony jsou létající dopravní prostředky, které nemají na palubě posádku. UA je součástí bezpilotního systému UAS (Systém bezpilotního letadla – Unmanned aircraft system), který se skládá ze samotného letadla, pozemního ovládání a komunikačního systému mezi nimi. Toto označení se používá výhradně pro vojenské účely, ale tento princip je víceméně využíván bezpilotními letadly pro většinu účelů. Použití bezpilotních letadel bylo zpočátku výhradně pro vojenské účely (např. nebezpečné průzkumné mise na nepřátelském území). V dnešní době se drony klasifikují od průzkumných, přes bojové, logistické, výzkumné až po civilní a komerční.

Cílem této práce je seznámit se s bezpilotním letadlem Ryze Tello a různými způsoby jeho řízení. Jsou zde popsány jednotlivé způsoby ovládání a jejich použití v praxi. Dále je rozebrána navigace robotů a bezpilotních letadel v uzavřených prostorech a druhy lokalizace těchto robotů.

V praktické části práce je řešeno implementování vytvořené ROS aplikace, která obsahuje několik podprogramů, které jsou označovány jako nody. Je zde detailně popsána instalace aplikace a její následovné spuštění. Dále je vysvětlena funkčnost a úkol jednotlivých nodů.

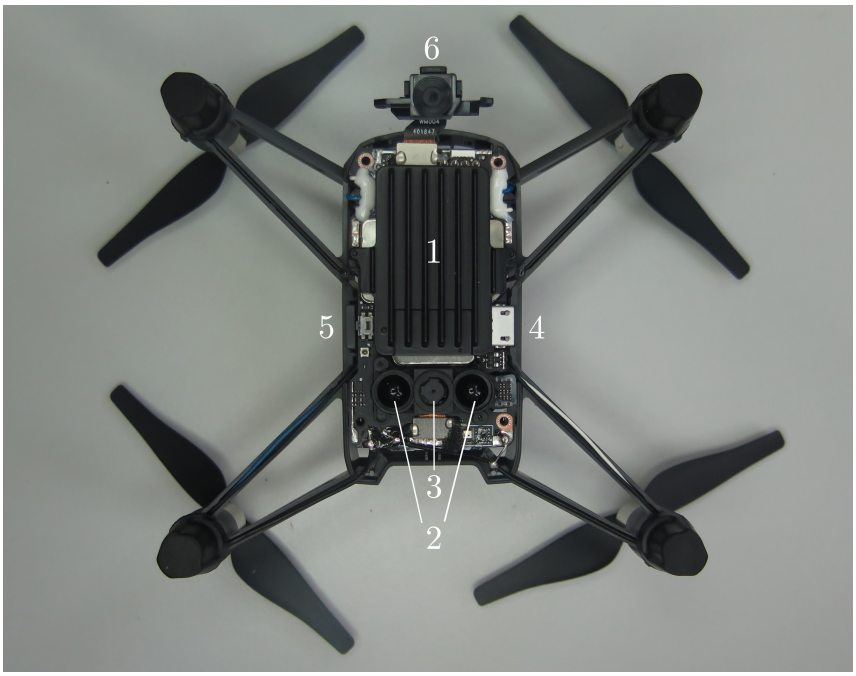
Na závěr je zhodnoceno vybrané řešení a popsány problémy, které nastaly při zhotovování této práce. V neposlední řadě vysvětluji porovnání s jinými druhy řešení a poté případné rozšíření aplikace o jiná komplexní využití.

1 Bezpilotní letadlo Ryze Tello

Pro tuto práci je využito bezpilotního letadla Ryze Tello EDU (model TLW004), které patří mezi cenově dostupnější a jehož největší výhodou je možnost ovládání mimo mobilní aplikace pomocí SDK (Sada vývojových nástrojů – Software development kit) dodaným přímo od Ryze nebo jeho upravenou verzí pro Python TelloPy, která je využitá i ve většině driverů pro ROS (Robot operating system).

1.1 Konstrukce bezpilotního letadla

Základem bezpilotního letadla je plastové tělo, ve kterém se zespodu (obr. 1.1) nachází základní deska, na které je umístěn procesor Intel s pasivním chlazením (1), díky němuž se po chvíli na zemi přehřeje a vypne. V zadní části se poté nachází přijímač a vysílač laserového výškoměru (2) a mezi nimi kamerka (3) pro stabilizaci a údaje o poloze a rychlosti Tella pomocí optického toku. Z tohoto důvodu má Tello problém zůstat na místě při slabém osvětlení (obr. 1.3) nebo pokud podlaha nemá texturu. Na levé straně se nachází USB micro (4) pro nabíjení baterie a na pravé straně Power tlačítko (5), jehož stiskem se Tello zapíná a vypíná. V přední části je indikační dioda a HD kamera (6), která dokáže natáčet video v rozlišení 720p při 30 snímcích za sekundu a fotit při rozlišení 5 MegaPixel. V zadní části se nachází WiFi modul, jehož anténa je vedena zadními rameny. Na desce je také umístěn barometr.



Obr. 1.1: Spodní část bezpilotního letadla.

V horní části těla se nachází odnímatelná baterie s kapacitou 1100 mAh, což odpovídá přibližně 13 minutám letu. Bezpilotní letadlo pohání 4 coreless motory, jejichž napájení je vedeno samotnými rameny, na kterých jsou umístěny a celkově váží přibližně 80 gramů i s baterií.

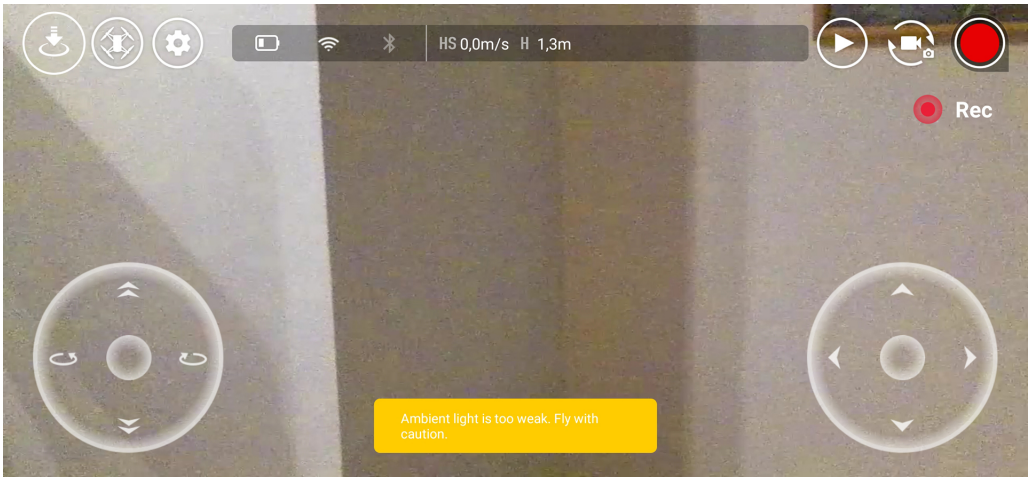
1.2 Komunikace s bezpilotním letadlem

1.2.1 Mobilní aplikace

Při výchozím nastavení Tella se po zapnutí vytvoří WiFi hotspot (v mém případě TELLO-FCA831) bez zabezpečení. Po připojení k této WiFi se indikační dioda rozbliká žlutě. Nyní můžeme Tello ovládat přes mobilní aplikaci.

TELLO

Tato aplikace slouží k přímému ovládání bezpilotního letadla pomocí joysticků (obr 1.3). Dále lze fotit fotografie a natáčet videa, která se ukládají do úložiště mobilu. Aplikace dále nabízí několik letových módů jako například natočení 360° videa a podobně.



Obr. 1.2: Mobilní aplikace TELLO.

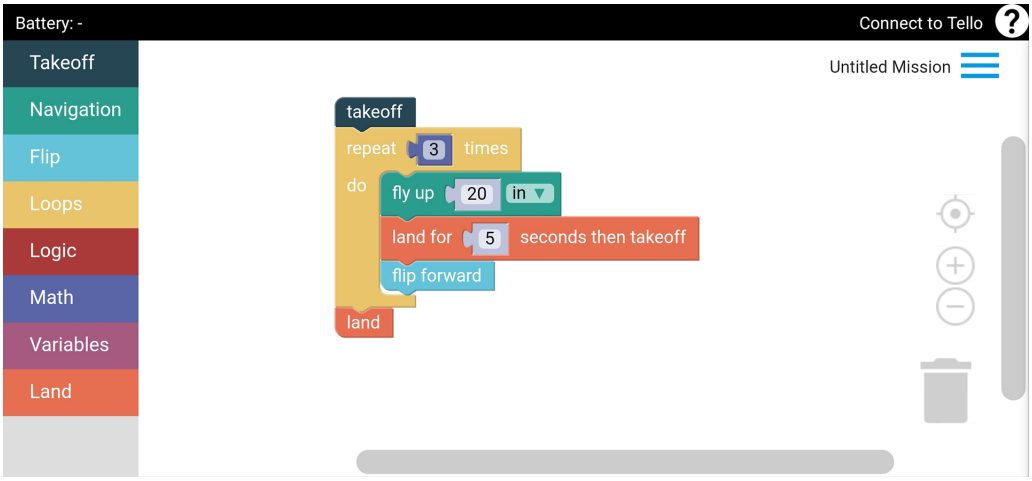
Aplikace podporuje VR headset, připojení bluetooth ovladače a lze změnit WiFi bezpilotního letadla. Dále je zde možnost zkalibrovat senzory, nastavení maximální výšky, nastavení bitratu videa a další.

TELLO EDU

Aplikace také nabízí přímé avšak zjednodušené ovládání bezpilotního letadla, ale hlavním cílem je programování letu pomocí jednoduchých bloků a Mission Padů, které jsou součástí balení a slouží jako orientační značení, které letadlo snímá spodní kamerou.

DroneBlocks

Vychází ze stejného principu jako předchozí aplikace tj. blokové programovaní letu bezpilotního letadla, ale bez animované grafiky a herního stylu jako TELLO EDU, které je spíše určeno pro děti.



Obr. 1.3: Mobilní aplikace DroneBlocks.

2 Softwarové nástroje

2.1 Tello SDK

Pro pokročilé ovládání UA (Bezpilotní letadlo – Unmanned aircraft) můžeme využít již připraveného SDK od výrobce, jenž se skládá ze zasílání několika základních příkazů pro pohyb UA a nastavení WiFi. Poté lze přijímat video stream a informace o stavu UA.

Po připojení k WiFi UA lze začít posílat příkazy buď pomocí Packet Senderu nebo jednoduchého Python skriptu od výrobce dostupného online. Příkazy převedené do hexadecimálního tvaru se posílají na IP adresu UA tj. 192.168.10.1 UDP PORT: 8889. Po odeslání a vykonání příkazu UA odpoví "ok"a v případě nevykonání příkazu odpoví "error".

Pro přijímání informací o stavu UA Tello State je potřeba vytvořit UDP server a číst zprávy z IP adresy 0.0.0.0 UDP PORT:8890. UA poskytuje informace o aktuální rychlosti ve třech směrech, orientaci UA, teplotě procesoru, času letu, výšce, baterii, tlaku a pár dalších.

Stejný proces platí i pro videostream, kdy IP adresa je stejná, ale liší se port, tj. 0.0.0.0 UDP PORT 11111. Tento stream můžeme jednoduše přehrát pomocí balíčku ffplay. Pro povolení režimu SDK je nutné jako první zaslat příkaz **command**. Tento příkaz je nutné zaslat minimálně jednou za 15 vteřin jinak UA přistane. SDK má dohromady 34 příkazů, ale z toho je 6 pro použití s Mission Pady, které jsem se rozhodl nepoužít. Proto hlavní příkazy, které jsem využíval a testoval jsou:

- **command** - přepnutí do SDK režimu a zamezení timeoutu při letu,
- **takeoff** - vzlétnutí zhruba do výšky 0,7 m,
- **land** - přistání, motory se zastaví jakmile je vertikální akcelerace téměř nulová,
- **streamon** - zapnutí videostreamu do UDP serveru,
- **streamoff** - vypnutí videostreamu,
- **emergency** - vypne motory v jakékoliv situaci (i během letu),
- [**směr**] [**x**] - uletí **x** cm v daném směru: **up, down, left, right, forward** a **back**,
- [**orientace**] [**x**] - otočí se o **x** stupňů v zadané orientaci: **cw** (po směru hod. ručiček) a nebo **ccw** (protisměru hod. ručiček),
- **stop** - zastaví se ve vzduchu,
- **go** [**x**] [**y**] [**z**] [**rychlost**] - přeletí na zadané souřadnice **x,y,z** [cm] zadanou rychlostí vzhledem k pozici UA,
- **speed** [**x**] - nastavení rychlosti letu **x** = 10-100 cm/s,
- **rc** [**x**] [**y**] [**z**] [**r**] - letí zadaným směrem, kde **x, y, z** je vektor letu a **r** rotace UA, dokud není směr změněn nebo vynulován (tj. pokračuje v tomto směru i

po přistání a vzlétnutí),

- **flip** [**směr**] - přemet UA daným směrem: **r, f, b, l**. Pouze zajímavý trik bez výrazného využití, nelze provádět pod 50% baterie,
- **battery?** - odpoví aktuální procentem baterie,
- **speed?** - odpoví aktuálně nastavenou rychlostí UA,
- **time?** - odpoví aktuálním časem letu,
- **ap** [**ssid**] [**pass**] - slouží pro připojení UA k již existující WiFi s názvem **ssid** a heslem **pass**. Výrazně se tak zlepšuje dosah, ale s aktuálním firmwarem není možné přijímat videostream při používání externí WiFi, což je v mém případě nepoužitelné.

Příkazy battery?, speed? a time? jsou zbytečné pokud využíváme Tello State, která nám tyto informace poskytuje aktuálně bez nutnosti zasílat příkaz.

2.2 TelloPy

TelloPy je Python balíček dostupný na Githubu, který byl vytvořen reverzním inženýrstvím mobilní aplikace TELLO, kdy jsou vysílány a čteny nezpracované packety UA. Tato knihovna je široce využívána ve většině ROS driverů, protože nabízí více funkcí než samotné SDK od Ryze. Přidává možnost *throw and go*, což je vzlétnutí UA vyhozením do vzduchu, přistání na dlani, změna výškového limitu, nastavení video módu (16:9, 4:3), clony kamery, bitratu video enkóderu, zapnutí fast módu (pro venkovní použití nezbytné) a další. TelloPy obsahuje i balíček pygame, takže je možné UA ovládat pomocí gamepadu nebo klávesnice. Balíček lze nainstalovat pomocí příkazu **pip install tellopy**.

2.3 ROS

ROS (Robot operating system) je linuxový open-source framework pro modulární použití u velké škály robotických aplikací. Využívá objektového programování, což je metoda programování, která dává přednost datům oproti procedurám s daty a především komunikaci v modulárním systému.

Hlavní část takového systému se nazývá balíček (package). Balíček obsahuje spustitelné programy ROSu, které se nazývají nody, dále ROS knihovny, konfigurační soubory a další software, který by měl být pospolu. Cílem tohoto uspořádání je poskytnout jednoduchost použití a potencionální využití u jiné aplikace. Toto umožňuje právě modulární použití několika balíčků najednou, čímž docílíme žádoucího výsledku pro naši aplikaci.

Tyto balíčky si předávají data v peer-to-peer síti. Základními prvky této sítě jsou právě nody jednotlivých balíčků, které odebírají topicy k vykonání úkonů (např. pohyb robota) a jiné publikují data ze senzorů pro jejich použití.

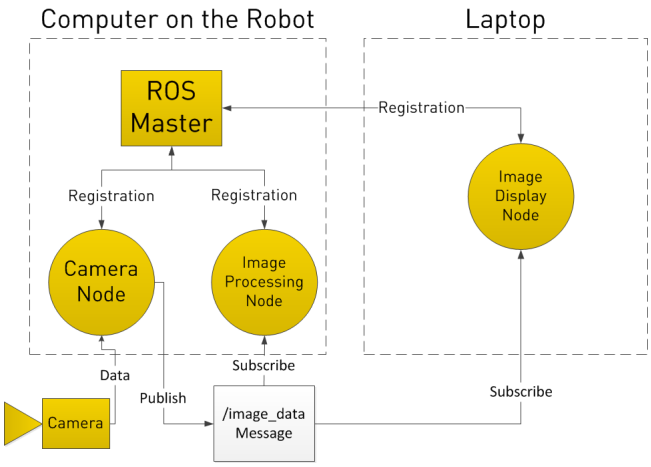
ROS master je doménový server, který sdílí topicy jednotlivým nodům v systému. Pokud chce node publikovat zprávu do topicu, oznámí to masteru. Stejné se děje pokud chce node odebírat topic. ROS master se spouští terminálovým příkazem **roscore** a musí vždy běžet. Při výchozím nastavení běží ROS master na adrese *http://localhost:11311*. Další výhodou ROSu je, že můžeme zapnout ROS master na jednom centrálním počítači a nody následně mohou komunikovat přes lokální síť z jiného počítače. Vytvoříme tak distribuovaný ROS.

Nody si data předávají pomocí zpráv. Zpráva obsahuje data uspořádána ve standardizovaném formátu pro jednoduché použití ostatními nody. Formátů takové zprávy je několik definovaných a lze vytvořit vlastní. V této práci je použito předdefinovaných zpráv jako například geometry_msgs/Twist nebo std_msgs/Empty, ale také vlastní tello_driver/TelloStatus (obr. 2.1). [8]

```
float32 height_m
float32 speed_northing_mps
float32 speed_easting_mps
float32 speed_horizontal_mps
float32 speed_vertical_mps
float32 flight_time_sec
bool imu_state
bool pressure_state
bool down_visual_state
bool power_state
bool battery_state
bool gravity_state
bool wind_state
uint8 imu_calibration_state
uint8 battery_percentage
float32 drone_fly_time_left_sec
float32 drone_battery_left_sec
bool is_flying
bool is_on_ground
bool is_em_open
bool is_drone_hover
bool is_outage_recording
bool is_battery_low
bool is_battery_lower
bool is_factory_mode
uint8 fly_mode
float32 throw_takeoff_timer_sec
uint8 camera_state
uint8 electrical_machinery_state
bool front_in
bool front_out
bool front_lsc
float32 temperature_height_m
float32 cmd_roll_ratio
float32 cmd_pitch_ratio
float32 cmd_yaw_ratio
float32 cmd_vspeed_ratio
bool cmd_fast_mode
```

Obr. 2.1: Struktura zprávy tello_driver/TelloStatus.

Zprávy, které nody odebírají a publikují, se předávají přes topicy. Topic je doména, ve které se publikují pouze jí určené zprávy s daným formátem. O to se starají právě publisher nody, které získají data ze snímače, přetvoří na formát dané zprávy a publikují do topicu. Jinde v modulárním systému se nachází subscriber node, která tento topic odebírá a čte každou zprávu do tohoto topicu publikovanou. Zjednodušené schéma toho systém je vidět na obrázku 2.2.



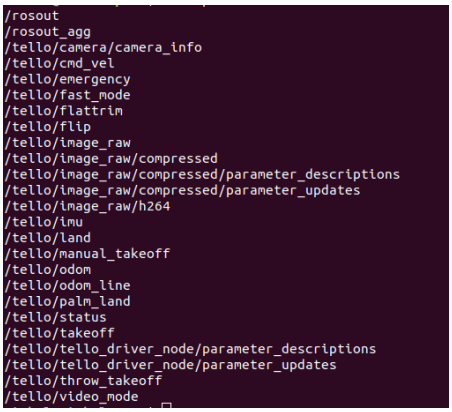
Obr. 2.2: Příklad komunikace pomocí topiců a zpráv. [17]

Z obrázku je patrné, že odebírat, ale i publikovat do jednoho topicu může několik nodů. To umožňuje plynulé dodávání dat všem nodům, nehledě na to, z jakého balíčku pochází. Jeden node může také zároveň publikovat do více topiců najednou a zároveň jich více odebírat.

O aktivních topicích v našem systému se můžeme dozvědět pomocí terminálového příkazu **rostopic**. Pomocí **rostopic list** získáme seznam všech publikovaných topiců jako v případě mého systému na obrázku 2.3.

Podobně použitím příkazu **rostopic echo [topic]** můžeme nechat do terminálu vypisovat zprávy publikované v daném topicu. Popřípadě pomocí **rostopic pub [topic] [data]** publikovat data ručně do daného topicu. Tyto dva příkazy jsou určeny spíše pro debug programu.

Nody můžeme spouštět pomocí takzvaných launch souborů. Jedná se o jednoduchou metodu jak spustit několik nodů nebo balíčků najednou. Příkaz vypadá následovně: **roslauch [jmeno_balicku] [soubor.launch]**. Launch soubory jsou psány v markdown jazyce XML a popisují jaké nody se spustí s jakými parametry.



Obr. 2.3: Seznam topiců v balíčku tello_driver.

2.3.1 Rviz

Rviz je grafické rozhraní pro ROS, které umožňuje zobrazit spoustu informací. Data lze zobrazit jak v 3D, tak i 2D. Nejčastěji se využívá pro zobrazení toho, co robot vidí, nebo jako v mém případě poloha robota vzhledem k okolí. Umí zobrazovat většinu předdefinovaných zpráv a chová se jako subscriber. Odebírá určitý topic a tyto data pak zobrazuje v grafickém podání. Dá se využít jako pro člověka jednodušší debuging nástroj právě díky možnosti sledování nějakého objektu pohybujícího se v prostoru na místo rychle se měnících čísel.[18]

3 Navigace robotů v uzavřených prostorech

Navigace robotů v uzavřených prostorech má mnoho druhů využití. Ať už od běžného domácího použití, jako například robotické vysavače přes automatické průmyslové manipulátory až po speciální záchranné bezpilotní letadla, vozítka nebo ponorky pro využití v životu nebezpečných situacích a místech.

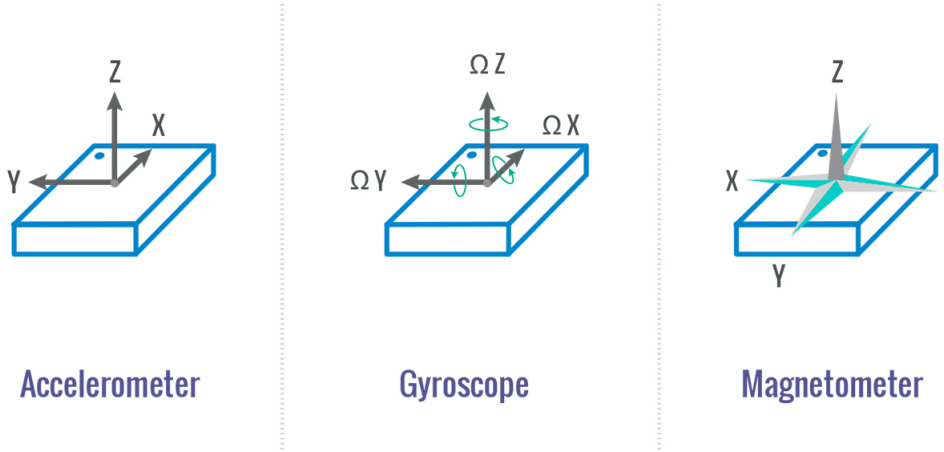
Hlavním problémem navigace těchto robotů je jejich lokalizace, vzhledem k tomu, že ve více patrových budovách, garážích nebo podzemních a podvodních lokacích je využití GPS signálu nepřesné a proto nevhodné, je potřeba využít jinou metodu nebo jejich kombinace. Žádná metoda však není universální a je potřeba navrhnout ideální řešení pro daný problém.

Z toho důvodu byly využity metody jako je IMU (Inertial measurement unit), které využívá akcelerometrů, gyroskopů a magnetometrů k měření rychlostí a orientace tělesa. Nejčastěji se využívá u všech druhů letadel, satelitů a dalších.

3.1 Druhy navigace

Inerciální měření

Jedná se o nejrozšířenější druh lokalizace v budovách, která je využita ve smartphonech, robotech a jiných. Uvnitř budov se využívá pouze v 2D pro zvýšení přesnosti.



Obr. 3.1: Inertial measurement unit. [9]

Vzhledem k tomu, že nejspíše nebudeme znát rozložení budovy je třeba tuto metodu zkombinovat s jinou. Nabízí se kombinace strojového učení, kdy robot mapuje okolí a zná lokace, kde už byl. Toto řešení není úplně vhodné pro bezpilotní letadlo, protože při nárazu může dojít k havárii a nebo dokonce poškození bezpilotního letadla. [11]

Lokalizace založená na vizuálních značkách

Metoda vhodná pro zařízení, která disponují kamerou. Většina bezpilotních letadel má minimálně jednu, kterou lze pro tento účel využít. Kamera snímá značky rozmístěné po budově a pokud známe jejich velikost není problém vypočítat polohu UA vůči dané značce.

Pro známe lokace lze využít podobnou metodu, která rozpoznává objekty a místa uložené v databázi a porovnáváním se vstupem z kamery pak určuje svoji polohu.

Bezdrátové technologie

Jakákoliv bezdrátová technologie může být využita pro lokalizaci. Nejčastěji se využívá Wi-Fi vzhledem k její dostupnosti a pokrytí. Technologie je založena na rozmístění několika přístupových bodů a pomocí síly signálu jednotlivých bodů je vypočítána poloha robota.

Další používaná metoda je Bluetooth. Ta však měří pouze přiblížení robota k vysílači. Novější verze Bluetooth 5.1 dokáže díky měření úhlu k vysílači měřit i přesnou polohu na centimetry.[11]

3.2 AR detekce značek

Vzhledem k UA, které je využito pro moji práci se jako nejlepší řešení nabízí využití detekce značek pomocí zabudované kamery, jejíž obraz můžu bez problému zpracovat a tím určit pozici vůči okolí.

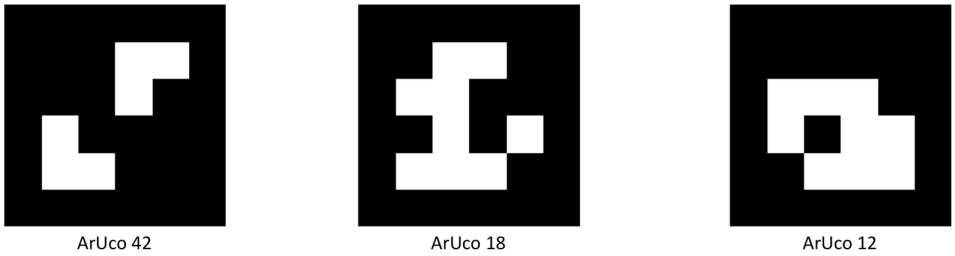
Lokalizace využitím detekce vizuálních značek je jedna z nejdůležitějších částí u mnoha aplikací využívajících počítačové vidění, jako například právě navigace robotů nebo rozšířená realita. Jedná se o proces, který se snaží hledat 2D referenční značky v reálném světě pomocí kamery.

3.2.1 Značky

ArUco značky, které byly využity pro lokalizaci v mém případě, jsou čtvercové matice rozměru 5x5 s vnějším černým ohraničením, které slouží pro rychlé bezchybné rozpoznání algoritmem. Každá značka má unikátní ID, které je dáno rozložením bitů. Značka s rozměrem 5x5 má 25 bitů.

Každá značka má unikátní rozložení bitů a značky jdoucí po sobě mají maximální změnu pro vyloučení chyb detekce.

Značky je potřeba vygenerovat a vytisknout. Proto existuje několik řešení. Samotná knihovna ArUco obsahuje funkce pro vygenerování značek, ale existují i online nástroje pro jejich generaci. Pro moji práci jsem používal online generátor



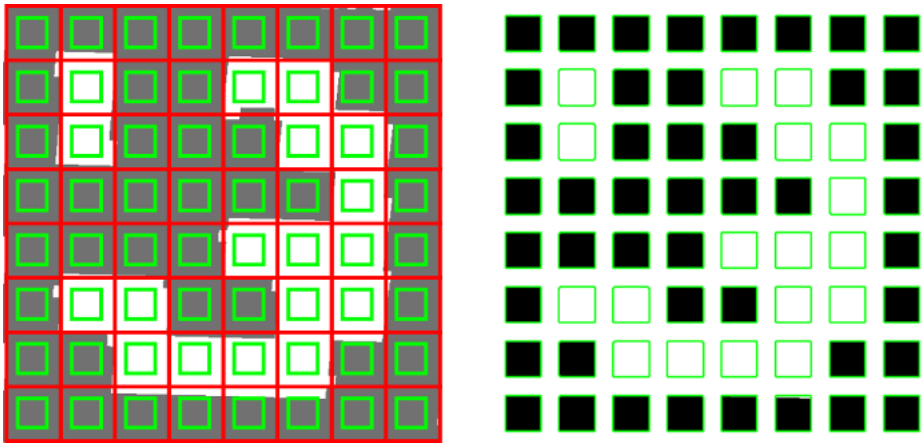
Obr. 3.2: Příklad ArUco značek. [10]

https://chev.me/arucogen/. Lze zde vybrat knihovna (original, 4x4, 5x5 ...), velikost značky a samozřejmě ID. [10]

Detekce

Pokud má kamera v zorném poli alespoň jednu značku, detektor poskytuje seznam detekovaných značek. O každé detekované značce poté zná polohu všech čtyř rohů a ID, které značka představuje.

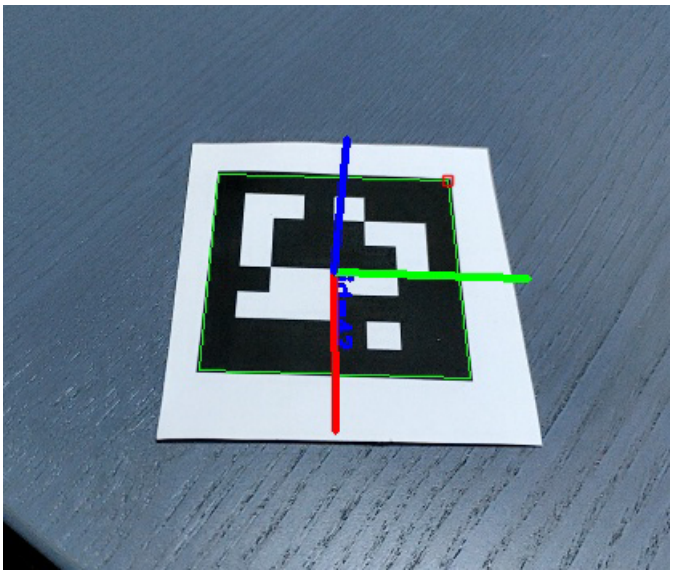
Detekce je tvořena dvěma hlavními kroky. Detektor nejdříve označí všechny kandidáty na značky a poté prověří vnitřek značky a vyfiltruje tak referenční značky používané pro lokalizace od podobný tvarů, které značky pouze připomínají. Poté jsou detekované značky transformovány tak, aby odpovídaly jejich referenční podobě. To znamená, že jsou upraveny na velikost 1:1 a je odstraněna perspektiva. Dále je značka rozdělena na jednotlivé bity, které jsou analyzovány jestli mají černou nebo bílou výplň. Příklad analýzy značky je vidět na obrázku 3.3. [10]



Obr. 3.3: Příklad analýzy bitů značky. [10]

Odhad polohy

Pokud je známa velikost značky, lze bez problému určit vzdálenost kamery. Značka má tvar čtverce, tím pádem detektor pomocí vzdálenosti a polohy jednotlivých rohů značky určí natočení vůči kameře. Spolu se vzdáleností lze tedy určit přesnou polohu značky vůči kameře. Do značky jsou poté přidány osy a ID a je reprezentována jako transformace 3.4. [10]



Obr. 3.4: Detekovaná značka s osami (upraveno). [12]

4 ROS aplikace

Jak již bylo řečeno, možnost ovládat UA pomocí příkazů, přijímání videostreamu a stavu UA nabízí použití ROSu.

4.1 ROS Tello driver

Pro Tello se na Githubu nachází již několik driverů, z nichž většina využívá právě knihovnu TelloPy. Jeden z prvních repositářů dostupných na Githubu[15] je postaven právě na TelloPy a je víceméně připraven k použití. Po pár dnech používání nastaly problémy s verzemi Pythonu. Driver je psán pro Python2, ale stále se snažil pustit v Pythonu3. Z toho důvodu jsem zkusil kód upravit podle rad autora, ale poté některé funkce nefungovaly tak jak by měly a vznikaly další problémy. Druhý driver, také dostupný na Githubu[16], který jsem zkusil, je inspirován předchozím driverem. Instalaci tohoto driveru provázelo také pár problémů. Z neznámého důvodu jsem nebyl schopný doinstalovat povinný ROS balíček použitím apt install ros-melodic-camera-info-manager-py. Proto jsem tento balíček zahrnul do svého repositáře, aby při instalaci nedocházelo k problémům.

4.1.1 Topicy

Hlavní node tello_driver_node publikuje šest topiců a odebírá jich deset. Publikované topicy a typy zpráv jsou:

- **/tello/camera/camera_info** (sensor_msgs/CameraInfo) - informace o přenášeném videu (např. rozlišení)
- **/tello/image_raw** (sensor_msgs/Image) - nezpracovaný video signál z kamery
- **/tello/image/raw/h264** (h264_image_transport/H264Packet) - zpracovaný video signál kodekem h264
- **/tello/odom** (nav_msgs/Odometry) - údaje o přibližné pozici a orientaci UA ve volném prostoru
- **/tello/imu** (sensor_msgs/Imu) - údaje o zrychlení ve všech osách
- **/tello/status** (tello_driver/TelloStatus) - údaje o stavu UA jako například stav baterie, čas letu, teploty, tlak, pokud UA letí a mnoho dalších

Topicy pro ovládání a nastavení UA, které odebírá jsou:

- **/tello/cmd_vel** (geometry_msgs/Twist) - směr, rychlost a rotace UA
- **/tello/emergency** (std_msgs/Empty) - Nouzové vypnutí motorů
- **/tello/fast_mode** (std_msgs/Empty) - Zapnutí fast módu. UA je přibližně 2x rychlejší

- **/tello/flattrim** (std_msgs/Empty) - Kalibrace senzorů
- **/tello/flip** (std_msgs/Uint8) - Přemet UA
- **/tello/land** (std_msgs/Empty) - Přistání UA
- **/tello/palm_land** (std_msgs/Empty) - Přistání UA na dlani
- **/tello/takeoff** (std_msgs/Empty) - Vzlétnutí UA
- **/tello/manual_takeoff** (std_msgs/Empty) - Vzlétnutí UA pomocí například nastavením nějakého směru letu.
- **/tello/throw_takeoff** (std_msgs/Empty) - Vzlétnutí UA hodem do vzduchu.

4.2 TelloBUT aplikace

TelloBUT aplikace, která je náplní této práce, je postavena na popsaném ROS driveru a jejím účelem je implementovat lokalizaci a navigaci cenově velmi dostupného bezpilotního letadla k již vytvořenému jádru základních funkcí a následně i možnost rozšířených o další komplexní řešení.

4.2.1 Instalace aplikace

Repositář se nachází na školním GitLabu pod názvem tellobut_ws Project ID: 121. Pro funkčnost je nutné mít nainstalovaný ROS Melodic. V případě nefunkčnosti nějakého kroku zkopírujte příkaz nejprve do například textového editoru pro odmazání enterů.

Nejprve nastavíme, aby počítač přijímal software z *packages.ros.org*.

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

Poté přidáme klíč pro zabezpečenou komunikaci s repozitářem.

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-
   key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Updatneme balíčkový index a provedeme plnou instalaci ROSu.

```
sudo apt update
sudo apt install ros-melodic-desktop-full
```

Nyní je třeba updatnout rosdep, který obstarává doinstalování potřebných balíčků při kompilaci.

```
sudo rosdep init
rosdep update
```

Nakonec přidáme načtení proměnných ROSu při spuštění terminálu a druhým řádkem je načteme do aktuálně otevřeného terminálu.

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Jedná se o plnou instalaci včetně rqt, rvizu a dalších. Po nainstalování ROSu by měl být nainstalovaný i Python, do kterého je nutné nainstalovat balíček TelloPy.

```
pip install tellopy
```

Pip by měl být nainstalovaný s Pythonem. Pokud se tak však nestalo je nutné ho doinstalovat.

Pro správné sestavení je potřeba doinstalovat kodek pro video.

```
sudo apt-get install ros-melodic-codec-image-transport
```

Po instalaci ROSu lze přistoupit k získání samotné aplikace TelloBUT. Tu je třeba naklonovat z GitLabu pomocí následujícího příkazu.

```
git clone --recurse-submodules git@student.robotika.ceitec.vutbr.cz:
    student-projects/tellobut_ws.git
```

Přesuneme se do workspacu.

```
cd tellobut_ws
```

A sestavíme balíček.

```
catkin_make
```

Catkin by měl být nainstalovaný společně s ROSem, pokud byla provedena plná instalace (tj. ros-melodic-desktop-full).

Dále přidáme cestu k workspacu, aby si Linux našel zkompilovaný balíček.

```
source devel/setup.bash
echo "source ~/tellobut_ws/devel/setup.bash" >> ~/.bashrc
```

Nyní by se po každém spuštění terminálu měl načíst.

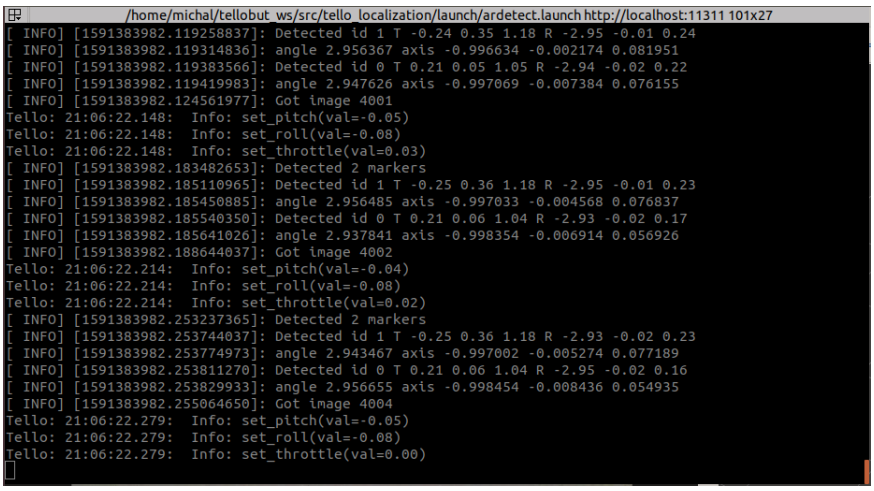
4.2.2 Spuštění aplikace

Nejprve je nutné spustit UA a připojit se k jeho WiFi.

ArUco detect node

Poté je třeba spustit ROScore, což je zajištěno pomocí launch souboru, který spustí driver a zároveň detekci tagů. Soubor také spustí RVIZ s přednastavenou konfigurací.

```
roslaunch tello_localization ardetect.launch
```



Obr. 4.1: Příklad okna terminálu nodu ArUco detect při letu UA.

Obrázek 4.1 je screenshot během letu UA. Ze zpráv nástroje loginfo lze vidět, že jsou detekovány dvě značky a následně jejich polohy. Dále je info o změně rychlostí UA.

Transformace značek místnosti

Dále je potřeba publikovat statické transformace místnosti, kde se bude UA pohybovat. V balíčku jsou obsaženy tři nastavení těchto transformací, a to místnost T12/SE 1.112 pod názvem tags_se1112.launch, můj pokoj na koleji a doma.

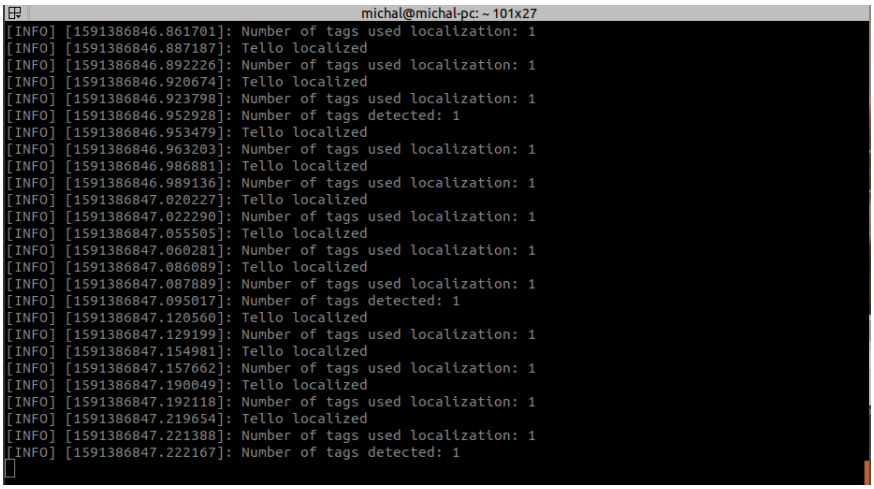
```
roslaunch tello_localization tags_<room_name>.launch
```

Po spuštění by se mělo v RVIZu zobrazit rozmístění tagů, a pokud je UA připojen, tak i obraz z kamery, kde lze vidět detekce tagů, pokud je UA na nějaký namířeno.

Localization node

Data, která jsou získána pomocí ArUco detectu jsou zpracovány algoritmem localization. Ten pustíme v dalším terminálovém okně.

```
rosrun tello_localization localization.py
```



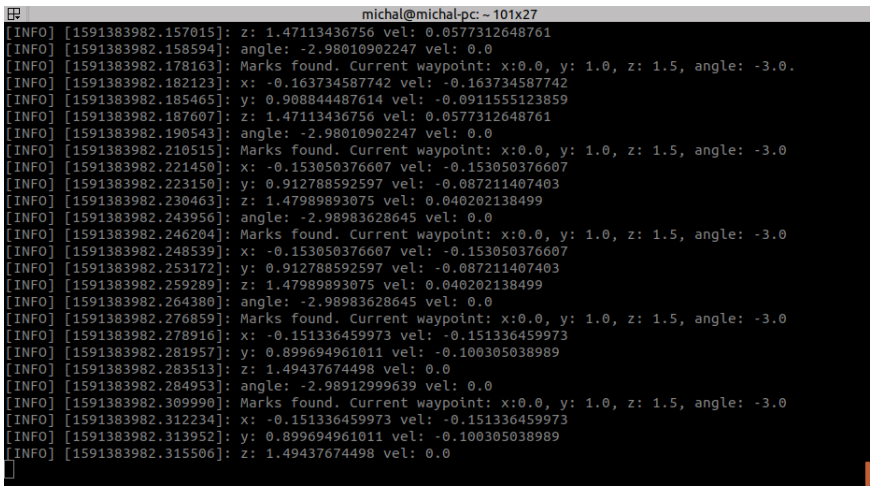
Obr. 4.2: Příklad okna terminálu nodu localization při letu UA.

Node hlásí, že se UA lokalizovalo a kolik značek bylo použito pro výpočet polohy.

Navigation node

Nyní je vše připraveno pro samotnou navigaci. Spuštěním následujího nodu UA vzlétne a přesune se do výchozího bodu, tj. x = 0.0 m, y = 1.0 m, z = 1.5 m a úhel 0° vůči definovanému počátku.

```
rosrun tello_navigation navigation.py
```



Obr. 4.3: Příklad okna terminálu nodu navigation při letu UA.

Z obrázku 4.3 je vidět, že UA detekuje značky a aktuální nastavený waypoint. Dále vypisuje aktuální polohu a úhel natočení vůči počátku a za každým prvkem akční zásah vypočítaný regulátorem.

Command node

Pro zadání waypointu, na který se má UA přesunout, je nutné spustit node command.

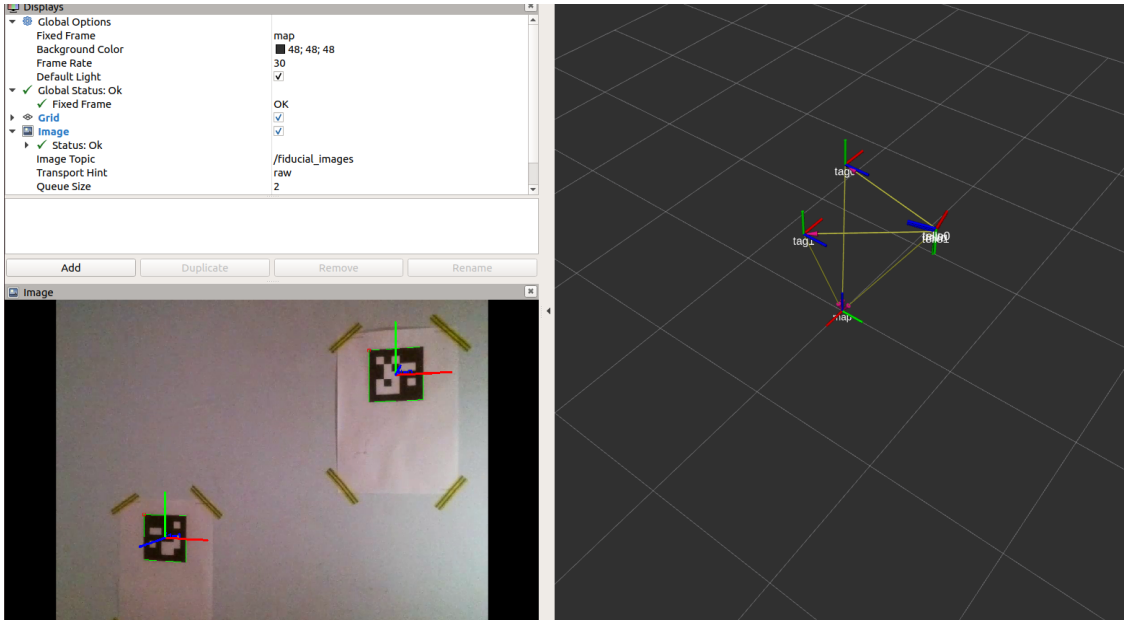
```
rosrun tello_command command.py
```

Obr. 4.4: Příklad zádání nového waypointu.

Všechny nody jsou bez problému opakovatelně spustitelné. Nedoporučuji ukončování nodů, pokud je UA v letu. UA bude po ukončení nodů pokračovat v letu, dokud se nevybije baterie nebo dokud nenarazí. Při ztrátě Wi-Fi spojení je možné znovu připojení a pomocí nodu command poté bezpečně přistát zadáním nečíselného vstupu jako například písmeno "j".

4.2.3 Zobrazení v RVIZu

Na obrázku 4.5 je vidět jak vypadá Rviz po zapnutí launch souborem. V levém sloupci jsou jednotlivé přidané prvky - Grid, TF a Image. Pod ním je přenos z kamery zpracovaný Aruco detectem. Uprostřed obrazovky se nachází 3D prostor s přidaným Gridem, statickými transformacemi představující značky a transformace představující UA.



Obr. 4.5: Okno RVIZu během letu.

U každé značky jsou na kameře přidány osy, které jsou natočeny podle polohy značky. Dále je vidět ID značky. Na obrázku 4.5 jsou vidět 3 statické transformace. Jsou to dvě značky a počátek, dále jsou vidět 3 transformace a to tello0, tello1 a tello. Tello0 a tello1 jsou transformace vypočítané pozice UA k jednotlivým značkám a transformace tello je výsledná vypočítaná transformace.

4.3 Nody

TelloBUT aplikace obsahuje 5 nodů. Node tello_driver, popsaný dříve, a aruco_detect jsou použity bez úpravy a jsou spuštěny spolu s ROScore. Nody localization, navigation a command jsou vytvořeny jako cíl této práce. Node localization odebírá zprávy o poloze značek a přepočítává je na na pozici UA vůči počátku. Node navigation odebírá právě pozici UA vůči počátku a obstarává regulaci letu na určitou polohu. A node command pouze publikuje požadovaný waypoint, na který má UA letět.

4.3.1 ArUco detect node

Node se spouští hned po spuštění ROScore launch souborem. Odebírá 2 topicy. První topic je */camera/camera_info*, ve kterém jsou publikovány zprávy od UA (*tello_driver*) typu *sensor_msgs/CameraInfo*. Tello_driver publikuje tento topic se svým prefixem, proto je v launch souboru topix */tello/camera/camera_info* přemapován na korektní název bez prefixu */camera/camera_info*. Tyto údaje jsou potřeba pro výpočet matice kamery, která je použita pro odhad polohy kamery vůči značce.

Druhý topic, který odebírá je */camera*. Tello_driver publikuje dva topicy */tello/image_raw* a */tello/image*. Topic */tello/image_raw* obsahuje nezpracované snímky přímo z kamery a topic */tello/image* by měl pomocí podpůrného balíčku překompresovat video kodekem h264. Z důvodu nekompatibilních verzí kodeku image transportu a tello_driver je proto v launch souboru přepublikován a kodekem kompresován obraz z topicu */tello/image_raw* do topicu */camera*. Tím je zajištěna správná funkčnost obrazu pro ArUco detect node.

Node publikuje dva typy zpráv. První je typu *fiducial_msgs/Fiducials* do topicu */fiducial_vertices* a zpráva obsahuje informace o vrcholech detekovaných značek. Druhá je typu *fiducial_msgs/FiducialTransforms* do topicu */fiducial_transforms* a představuje transformaci odpovídající poloze značky vůči kameře.

4.3.2 Localization node

Node odebírá zprávy z topicu */fiducial_transforms*. Při každé nové zprávě v tomto topicu se zavolá funkce localize s jediným parametrem, a to je právě přijatá zpráva. Všechny transformace jsou nejprve přidány do pole a poté je vždy poslední transformace v poli inverzně transformována, aby odpovídala poloze kamery vůči značce.

Takto upravené transformace aktuálně detekovaných značek jsou opět vloženy do pole. Poté je provedena fúze jednotlivých transformací. Prvky jednotlivých transformací v poli jsou sečteny a následně poděleny jejich počtem. Je tak vypočten průměr jednotlivych prvků a výsledná transformace je dále publikována do topicu *tf*.

4.3.3 Navigation node

K samotnému řízení UA slouží právě navigation node. Transformace publikované localization nodem jsou zde čteny z topic *tf* a následně algoritmem zpracovány.

Po spuštění nodu dojde k inicializaci jak samotného nodu, tak konstant jako třeba výchozí waypoint a další. Výchozí nastavený waypoint je x = 0.0 m, y = 1.0 m, z = 1.5 m a úhel = 0°. Po inicializaci se čeká až se node připojí k topicům pro ovládání UA. To jsou topicy */tello/cmd_vel*, */tello/takeoff* a */tello/land*. Pokud se k těmto topicům node nepřípojí do 20 sekund dojde k timeoutu a node se ukončí.

Po úspěšném připojení k topicům je UA připraveno k letu. Publikuje se tedy Empty zpráva do topicu */tello/takeoff*. Po zaslání této zprávy čeká 4 sekundy. Poté je publikována další Empty zpráva do topicu */tello/fastmode*. Nyní je povolen fastmode a UA reaguje rychleji na změnu akčních veličin regulátoru. Nyní je UA ve vzduchu. S fastmodem je také publikován nulový vektor do topicu */tello/cmd_vel*. Pokud UA havaruje nebo nouzově přistane, zůstane nastaven poslední poslaný vektor rychlosti a UA by se po vzletu řídil tímto vektorem.

Nyní se ověřuje přítomnost značek v zorném poli. Pokud po vzletu UA nebo kdykoliv během letu nemá v zorném poli žádnou značku, je publikován nulový vektor rychlosti a UA se zastaví na místě po uplynutí čekací doby 1 sekundy, která je implementována kvůli srovnání obrazu kamery z důvodu jeho občasného trhání. Pokud se po této době stále neobjeví žádná značka UA se začne otáčet kolem své osy a hledat nějakou značku. UA se bude otáčet po dobu 20 sekund, což odpovídá téměř 2 otáčkám a pokud se stále neobjeví nějaká značka je opět publikován nulový vektor a následně Empty zpráva do topicu */tello/land* a UA přistane a node se vypne.

Popis regulace

Pokud je po vzletu nebo během letu 1 a více značek v zorném poli UA a nastavený waypoint je realná nenulová pozice volá se funkce navigation. V opačném případě je publikován nulový vektor a následně Empty zpráva do topicu */tello/land*. UA přistane a node se ukončí.

```
navigation(tfBuffer, waypoint.linear.x, waypoint.linear.y, waypoint.
    linear.z, waypoint.angular.z)
```

Funkce navigation požaduje při volání 5 argumentů. **tfBuffer** - data o aktuální pozici UA, která jsou dodávána z nodu localization ve tvaru transformace, **set_x** - požadovaná pozice v ose x, **set_y** - požadovaná pozice v ose y, **set_z** - požadovaná pozice v ose z a **set_angle** - požadovaný úhel natočení UA.

Funkce navigation obstarává hledání značek po vzletu, ale i při ztrátě během letu. Hledání je vyřešeno publikováním vektoru s nulovými linearními složkami a kladnou nebo zápornou konstantou pro rychlost otáčení. Konstanta je určována podle aktuálního úhlu natočení a požadovaného úhlu.

Pokud je viditelná alespoň jedna značka v zorném poli kamery je splněna podmínka pro vykonání druhé části funkce. Dodávaná transformace argumentem tfBuffer představuje polohu UA vůči značce, kterou právě snímá. Proto je vypočítaná poloha UA vůči počátku pomocí funkce *lookup_transform* z knihovny tf2_ros. V případě více značek se poloha počítá podle vypočítané zprůměrované transformace.

Lineární část přepočítané transformace z tfBuffer je předávána regulátoru. Pro každou osu je jeden regulátor typu P, kdy je od požadované pozice odečtena aktuální, vynásobena konstantou regulátoru a poté přepočítána podle aktuálního natočení UA.

```
if abs(set_x - zf.transform.translation.x) > 0.02:
        final_x = Kp * (set_x - zf.transform.translation.x)
else:
        final_x = 0.0
```

Pokud je aktuální hodnota různá od požadované o více než 2 centimetry dojde k výpočtu akčního zásahu. V opačném případě je akční zásah roven 0.

Úhlová část vypočítané transformace je ve tvaru kvaterniónu. Ten je převeden do tvaru RPY (Roll, Pitch, Yaw) pomocí funkce *euler_from_quaternion* knihovny tf. Regulace úhlu natočení UA je poté počítána z hodnoty yaw. Rychlost otáčení je z důvodu trhání obrazu konstantní a není nijak regulována. UA se otáčí dokud není v nějaké mezi od požadované pozice.

```
if abs(set_angle - yaw) > 0.03:
        if set_angle > yaw:
                final_rx = -0.5
        else:
                final_rx = 0.5
else:
        final_rx = 0.0
```

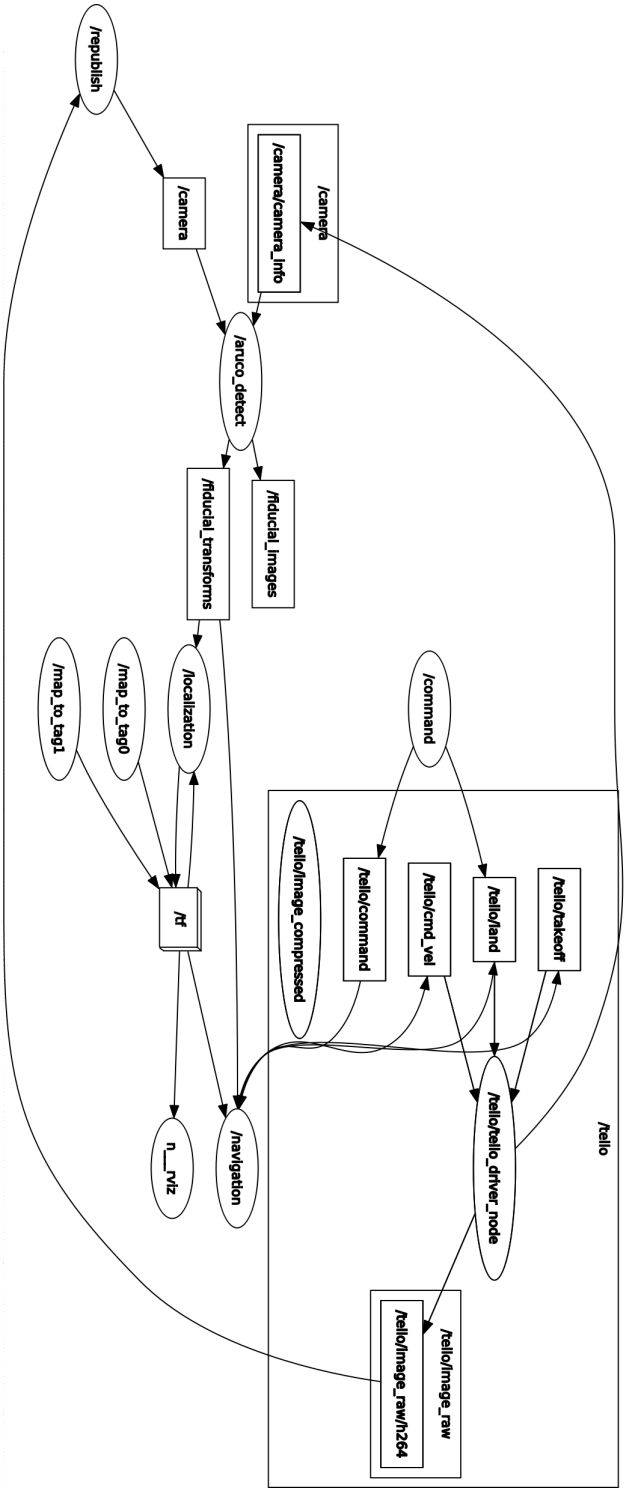
Pokud je rozdíl aktuálního úhlu natočení proti požadovanému větší než 3° je nastaven akční zásah pro otáčení. UA se začne otáčet po směru nebo proti směru hodinových ručiček podle aktuální otočení a požadovaného úhlu.

4.3.4 Command node

Pro změnu waypointu během letu slouží command node. Node stále čeká na vstup uživatele z terminálu. Vstup je zadáván v sekvenci x, y, z a úhel. Hodnoty x, y a z jsou v jednotkách metrů a úhel je zadáván v rozmezí -3.14 až 3.14, kdy při zadaném úhlu 0.0 u výchozí waypointu bude UA notočeno směrem k počátku. Po zadání realných a alespoň jednoho nenulového čísla se tyto hodnoty zabalí do zprávy typu Twist, která je následně publikována do topicu /tello/command.

V případě, že všechny hodnoty jsou rovny nule nebo některá z nich není číslo, node odešle nulový vektor a následně příkaz k přistání. Jedná se o pojistku pokud by UA zůstal v letu, aniž by běžel navigation node.

Na obrázku 4.6 je znázornění komunikace mezi nody pomocí topiců. Lze snadno říct, že většinu úkonů obstarává node /tello a ostatní nody komunikují s maximálně 4 topicy.

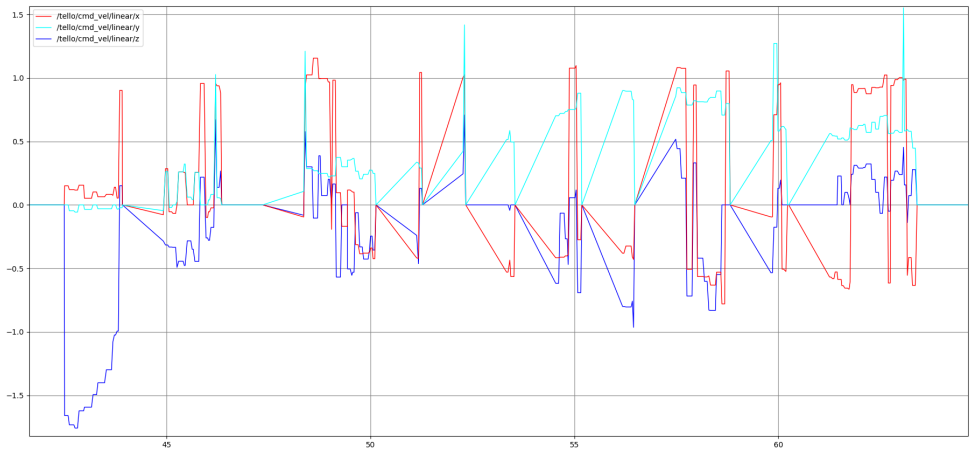


Obr. 4.6: ROSgraph komunikace mezi nody.

5 Testování a zhodnocení

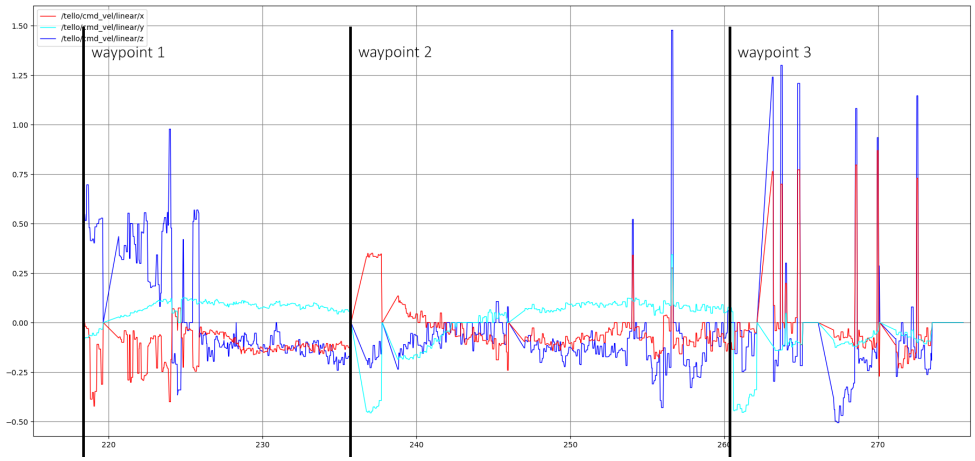
5.1 Testování

Testování aplikace jsem prováděl vždy s maximálně 3 značkami. Na obrázku 5.1 je vidět chování akčních veličin při vletu UA, prvnotní regulace a následné ztracení značek, opětovná regulace, timeout a přistání. Vodorovná osa je v jednotkách sekund a svislá udává nastavenou rychlost v metrech za sekundu. Červená křivka znázorňuje pohyb UA vpravo, světle modrá pohyb vpřed a tmavě modrá regulaci výšky.



Obr. 5.1: Chování akčních veličin při prvním letu.

U dalšího letu, kde jsem zaznamenal chování akčních veličin, jsem dvakrát změnil waypoint. Průběhy jsou vidět na obrázku 5.2. Waypoint 1 je výchozí waypoint, kde je x= 0.0 m, y = 1.5 m, z = 1.5 m a úhel 0.0°. Ze začátku grafu je vidět, jak se UA dostávalo do pozice. Druhý waypoint jsem zadal x = -0.5 m, y = 2.0 m, z = 1.5 m a úhel = 0.0°. Z chování akčních veličin je vidět, že se UA po zadání waypointu začalo pohybovat dozadu a vpravo a po zhruba 2 až 3 sekundách se již nacházelo v daném waypointu. Poslední waypoint jsem zadal x = -0.5 m, y = 2.5 m, z = 1.5 m a úhel = 0.0°. Po zadání waypointu je vidět nárust v záporném směru veličiny y, UA tedy začalo s pohybem dozadu. Opět po přibližně 2 sekundách je UA v pozici. Ke konci letu jsou vidět velké skoky veličiny z tj. výška a x tj. směr vpravo. Tyto skoky jsou způsobeny vzdáleností od značek, kdy UA často ztratilo data o své pozici. Dalším faktorem, který situaci nepřispěl je fakt, že tento waypoint se nachází nad hranou postele.



Obr. 5.2: Chování akčních veličin při druhém letu.

5.2 Problémy

Během ladění a testování jsem narazil na několik problémů. Některé z nich šly vyřešit úpravou kódu, ale některé přetrvaly.

Trhání obrazu

Zdaleka největší problémy způsobovalo trhání obrazu. Po vzletu se UA stabilizovalo na výchozím waypointu a než přišel nový waypoint došlo k trhnutí obrazu, detektor nezaznamenal žádnou značku a začal hledání. Dále to způsobovalo problém právě u hledání značek, kdy UA hledalo značku a právě v době, kdy by nějaká byla v jeho zorném poli, došlo k trhnutí obrazu, tím pádem detektor nic nezaznamenal, hledal dál a při malém pokrytí značkami nejčastěji pošle příkaz k přistání kvůli timeoutu.

Na obrázku 5.3 je vidět příklad trhnutí obrazu. V zorném poli jsou dvě značky, ale díky trhnutí nezaznamená detektor ani jednu.

Pokrytí značkami

Během testování jsem používal maximálně 3 značky, které byly rozmístěny po 90 stupních, abych pokryl co největší úhel, ale i tak existovala místa mezi značkami, kde se nenacházela žádná značka a z regulace na polohu přešel node na hledání značek. To v nějakých situacích vyvolávalo zbytečné zdržení.

Tento problém lze odstranit hustým pokrytím značkami, tak aby byly detekovány vždy alespoň dvě značky, kdy dochází k fůzi a lokalizace je tedy přesnější než při detekci pouze jedné značky.

Obr. 5.3: Ukázka trhnutí obrazu.

Přehřívání procesoru

Během testování kódu, konstant regulátoru nebo lokalizace jsem několikrát narazil na problém, že se mi UA přehřálo. Tím pádem se odpojilo a aplikace timeoutnula. Po přehřátí je poté potřeba chvíli počkat dokud se neschladí jinak není možné pokračovat.

Problém se částečně vyřešil použitím větráku, na který jsem UA odkládal, ale to vyžaduje přistávat vždy na stejném místě a při testování to nebylo většinou možné.

Vlastnosti kamery

Dalším problémem, avšak ne tak významným, byla integrovaná kamera UA. Pro správnou funkci detekce značek bylo vyžadováno optimálního osvětlení bez jasných zdrojů světla nebo odrazů. Například při natočení UA na okno za slunečného dne trvalo až 2 sekundy, než se balance bílé opět vrátila do použitelných poměrů a obraz byl čitelný.

Kamera měla limity i v minimální velikosti značek. To přímo souvisí s maximální vzdáleností od značky, kdy byl detektor schopen značku zachytit a rozpoznat. Maximální vzdálenost, kdy detekce ještě fungovala s velikostí značky 10 cm se pohybuje okolo 3 metrů. Poté již není detekce zaručena a UA s pravděpodobností přistane z důvodu timeoutu.

Let nad nerovným terénem

Během létaní v pokoji jsem se nemohl vyhnout nelétat nad různými objekty jako je postel, stůl a další. Tyto objekty mají každý jinou výšku proti zemi a při regulaci vyšky UA se toto projevovalo jako největší překážka.

Docházelo k "nadskakování"UA, když byl nastaveny waypoint například nad hranou stolu. Při mírném pohybu do stran se tedy UA nacházelo chvíli nad stolem a chvíli ne. To vyvolalo pokles nebo naopak vznesení výše, do kterého poté začal zasahovat regulátor, protože rozdíl požadované a značkami měřené polohy se neshodoval. To vedlo k oscilacím a jiným nežádoucím jevům.

5.3 Zhodnocení jiných řešení

Řešení lokalizace UA pomocí značek patří mezi ty používanější a zároveň spolehlivé. Avšak může dojít k situaci, kdy je třeba navigovat a lokalizovat UA v budově, kde není možnost připravit značky nebo jiné referenční objekty.

Pro takové účely existuje několik druhů lokalizace. Jako první se nabízí využití lidaru, ale vzhledem k váze těchto komponentů se nejspíše u menších UA neprosadí. Pro větší modely by váha nemusela hrát takovou roli, ale pro použití v budovách je velikost UA také rozhodující. Další problém může nastat při konstantním otáčení lidaru, kdy může působit na samotné UA a může dojít k nežádoucím jevům.

Dalším způsobem je využití ultrazvuku, kdy je UA vybaveno několika ultrazvukovými měniči a je měřena vzdálenost v několika směrech. Jedná o vcelku přesnou a zároveň levnou variantu. [13]

Osobně nejzajímavější způsob mi přišel využití stereo kamer, kdy jsou dvě kamery namířeny jedním směrem a pomocí známé vzdálenosti mezi kamerami a mírně rozdílného obrazu, který poskytují lze určit vzdálenost od objektu. Tato metoda však patří mezi ty méně přesné a zároveň složité na realizaci.[14]

5.4 Návrh dalšího možného rozšíření aplikace

Aplikace má v aktuální podobě dost prostu na zlepšení, ale zároveň se dá díky využití ROSu bez problému toto řešení rozšířit a vytvořit mnohem komplexnější systém lokalizace a navigace bezpilotního letadla Tello.

Pro pokračování v této aplikaci by se dalo určitě zapracovat na vylepšení lokalizačního algoritmu pro přesnější lokalizaci. To za předpokladu, že by bylo poskytnuto dostatečné pokrytí značek a algoritmus mohl provádět lokalizace vždy minimálně ze 3 značek, které by byly různě rozmístěny tj. na zdech, podlaze a natočené různými směry.

Navigační algoritmus má také spoustu nedostatků. S rozsáhlejším testováním v ideálních podmínkách by bylo třeba doladit regulační konstanty. Dále by bylo možné nějakým značkám přiřadit význam. Například značka s ID 49 znamená, že UA musí provést sadu úkonů. Obletění překážky, přistání nebo dokování, kdy by se UA zarovnalo a přistálo do dokovací stanice a čekalo na další příkazy. Vhodnou úpravou by bylo přidání omezení letu UA za rozmístěné značky, tím pádem by nedocházelo k možnosti zadat waypoint, který se nachází za značkou a tedy nejspíše za zdí. Počítá se však s případem, že jsou značky umístěny pouze na zdech a podlaze. Při použití značek, které by byly umístěny jako praporky v místnosti by bylo nutné udělat výjimky.

Další možnost vylepšení aplikace je zadávání waypointů a zobrazení záběrů z kamery. Nyní lze waypointy zadávat pouze pomocí terminálu nodem command a sledovat záběry z kamery v RVIZu. Tvorba uživatelského rozhraní, kde bude možnost zadat waypoint, přistát, vzlétnou, sledovat přenost z kamery s nebo bez výstupu detektoru značek, by byla velký krok k vylepšení aplikace.

Po provedení zmíněných vylepšení lze již pouze přidávat další funkce. Různé druhy letových módů, jako například kooperace s člověkem, integrace umělé inteligence a strojového učení, kdy by UA ukládalo pozice značek a mohlo se poté rozhodovat pro nejkratší a nejbezpečnější cestu mezi waypointy a jiné.

Největším zhodnocením aplikace by však nastalo, pokud by byla možnost připojit UA k již stavající Wi-Fi síti bez připojování jako Access Point. Ryze Tello takovou možnost podporuje, ale funkčnost je omezená. Během testování tohoto režimu šlo pouze zasílat příkazy, ale nebylo možné přijímat videostream, který je nejdůležitější součástí lokalizace.

Závěr

Tato bakalářská práce se zabývala problematikou ovládání bezpilotního letadla Ryze Tello z hlediska použití pro autonomní mise.

Po prozkoumání všech možností ovládání tohoto dronu, pomocí jak dodaného SDK, tak komunitou vytvořeného, jsem se rozhodl, že využití frameworku ROS s komunitním SDK bude nejideálnější vzhledem k dostupným materiálům a zároveň možnosti modulárně přidávat další funkce. Z dostupných ROS driverů pro bezpilotní letadlo Ryze Tello jsem vybral tello_driver od uživatele appie-17, který je založen na nejoblíbenějším tello_driveru na Githubu. Důvodem pro výběr tohoto driveru byl fakt, že driver je z mého pohledu zjednodušen a přívětivější k použití a během instalace a používání jsem narazil na výrazně menší množství problémů.

Během této práce se mi povedlo zprovoznit tello_driver tak, aby fungoval bez jakýchkoliv potíží, tj. funkční přenos kamery, bezchybné opětovné zapnutí a pár dalších, se kterým jsem měl problém u ostatních driverů.

Do aplikace jsem zakomponoval detekci značek pomocí knihovny ArUco detect, která zaručila bezproblémovou detekci rozmístěných značek při nejen ideálních podmínkách. Díky funkční detekci značek bylo možné vytvořit algoritmus pro lokalizaci UA využití fúze dat z jednotlivých značek a poskytovat pozici UA vůči stanovenému počátku.

S aktuální pozicí UA jsem poté vytvořil další node, který se staral o navigaci bezpilotního letadla v prostoru, kdy se s velkou procentuální úspěšností dokázal dostat na určený waypoint v přijatelném čase.

V poslední kapitole jsem popsal problémy, se kterými jsem se setkal během ladění a testování aplikace. K nejvýznamnějším problémům patřilo trhaní obrazu, což mírně souvisí s vlastnostmi kamery a chování UA při letu nad terénem s velkými okamžitými změnami výšky. Poté jsem navrhl možnosti vylepšení aktuální aplikace, které by určitě zvýšily hodnotu aplikace jako takové. A na závěr rozšíření nad rámec aplikace, kde se však jedná o komplexnější implementace a zhotovení.

Na přiloženém dvd je poté video letu se záznamem obrazovky. Záznam z ubuntu se celkem seká vzhledem k běžící kompresi videa u detektoru značek. Podstatné tam je však vidět. Na pravé straně jsem pomocí nodu command zadával waypointy.

Literatura

[1] Tello Specs. *RyzeRobotics* [online]. [cit. 2020-01-04]. Dostupné z: `https://www.ryzerobotics.com/tello/specs`

[2] Unmanned aerial vehicle. *Wikipedia* [online]. [cit. 2020-01-04]. Dostupné z: `https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle`

[3] About ROS. *ROS* [online]. [cit. 2020-01-04]. Dostupné z: `https://www.ros.org/about-ros/`

[4] ROS Introduction. *ROS* [online]. [cit. 2020-01-04]. Dostupné z: `https://wiki.ros.org/ROS/Introduction`

[5] SDK 2.0 User Guide. *RyzeRobotics* [online]. [cit. 2020-01-04]. Dostupné z: `https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf`

[6] ROS 101: Intro to the Robot Operating System. *Robohub* [online]. [cit. 2020-01-04]. Dostupné z: `https://robohub.org/ros-101-intro-to-the-robot-operating-system/`

[7] PYO, YoonSeok, HanCheol CHO, RyuWoon JUNG a TaeHoon LIM. ROS Robot Programming. Republic of Korea: ROBOTIS Co., 2017. ISBN 979-11-962307-1-5.

[8] LUM, Joshua S. UTILIZING ROBOT OPERATING SYSTEM (ROS) IN ROBOT VISION AND CONTROL. Monterey, California, 2015. Disertace. Naval Postgraduate School. Vedoucí práce Xiaoping Yun.

[9] Optimize your IMU with Dynamic Calibration [online]. [cit. 2020-06-06]. Dostupné z: https://www.ceva-dsp.com/ourblog/optimize-your-imu-with-dynamic-calibration/

[10] Detection of ArUco Markers [online]. [cit. 2020-06-06]. Dostupné z: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html

[11] Indoor positioning system [online]. [cit. 2020-06-06]. Dostupné z: https://en.wikipedia.org/wiki/Indoor_positioning_system

[12] ARCore Pose and Aruco estimatePoseSingleMarkers [online]. [cit. 2020-06-06]. Dostupné z: https://hemant9807.blogspot.com/2019/01/arcore-pose-and-aruco.html

[13] Review of UAV positioning in indoor environments and new proposal based on US measurements [online]. [cit. 2020-06-06]. Dostupné z: http://ceur-ws.org/Vol-2498/short35.pdf

[14] An Indoor Location-Based Positioning System Using Stereo Vision with the Drone Camera [online]. [cit. 2020-06-06]. Dostupné z: https://www.hindawi.com/journals/misy/2018/5160543/

[15] Tello_driver [online]. [cit. 2020-06-06]. Dostupné z: https://github.com/anqixu/tello_driver

[16] Tello_driver [online]. [cit. 2020-06-06]. Dostupné z: https://github.com/appie-17/tello_driver

[17] Intro to ROS [online]. [cit. 2020-06-06]. Dostupné z: http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html

[18] RVIZ UserGuide [online]. [cit. 2020-06-06]. Dostupné z: http://wiki.ros.org/rviz/UserGuide

Seznam symbolů, veličin a zkratek

UA	Bezpilotní letadlo – Unmanned aircraft
UAS	Systém bezpilotního letadla – Unmanned aircraft system
SDK	Sada vývojových nástrojů – Software development kit
ROS	Robot operating system
RPY	Roll, Pitch, Yaw
IMU	Inertial measurement unit

Seznam příloh

A Obsah přiloženého CD/DVD

Na přiloženém CD/DVD je celý adresář GIT aplikace tellobut_ws s přidanými knihovnami. Dále se zde nachází video s ukázkou letu UA se záznamem obrazovky RVIZu.

```
/ ..... kořenový adresář přiloženého CD/DVD
├── tellobus_ws.zip ..... GIT adresář aplikace
│   ├── .idea
│   │   └── ...
│   ├── src
│   │   ├── camera_info_manager_py ..... externí knihovna pro získání camera info
│   │   │   └── ...
│   │   ├── codec_image_transport ..... externí knihovna pro kompresi videa
│   │   │   └── ...
│   │   ├── fiducials. ..... externí knihovna pro detekci tagů
│   │   │   └── ...
│   │   ├── tello_command
│   │   │   ├── src
│   │   │   │   └── command.py
│   │   │   ├── CMakeLists.txt
│   │   │   └── package.xml
│   │   ├── tello_driver .5 ...
│   │   ├── tello_localization
│   │   │   ├── launch
│   │   │   │   ├── ardetect.launch
│   │   │   │   ├── localization.launch
│   │   │   │   ├── tags_0441.launch
│   │   │   │   ├── tags_kzl.launch
│   │   │   │   └── tags_se1112.launch
│   │   │   ├── rviz
│   │   │   │   └── rviz_config.rviz
│   │   │   ├── src
│   │   │   │   └── localization.py
│   │   │   ├── CMakeLists.txt
│   │   │   └── package.xml
│   │   ├── tello_navigation
│   │   │   ├── src
│   │   │   │   ├── navigation.py
│   │   │   │   ├── CMakeLists.txt
│   │   │   │   └── package.xml
│   │   └── CMakeLists.txt
│   ├── .catkin_workspace
│   ├── .gitignore
│   ├── .gitmodules
│   └── README.md
└── ukazka_letu_ua.mp4 ..... video s ukázkou letu a záznamem obrazovky
```