VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZPEČNÝ PRŮZKUM DRONEM S VYUŽITÍM CHYTRÉHO POHYBU PO TRAJEKTORIÍCH

SAFE DRONE EXPLORATION WITH CLEVER TRAJECTORY MOVEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ADAM FERENCZ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. BERAN VÍTĚZSLAV, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Ferencz Adam, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové vidění
Název: **Bezpečný průzkum dronem s využitím chytrého pohybu po trajektoriích**
Safe Drone Reconnaissance Based on Smart Trajectory Following
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte možnosti API produktů od společnosti DJI. Seznamte se s problematikou řízení dronů.
2. Navrhněte sadu nástrojů, které umožní pořizovat polohové záznamy z letu dronem a tyto trajektorie využije pro opakovaný bezpečný let po vybrané trajektorii. Manuální řízení bude přemapováno tak, aby se dron automaticky držel dané trajektorie a pilot mohl sledovat jiné cíle.
3. Implementujte navržené řešení s využitím vhodných knihoven.
4. Demonstrujte řešení na vhodných situacích a vyhodnoťte vlastnosti výsledného řešení na experimentech v reálném prostředí.
5. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.
- Dále dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Piloti dronu často musí hlídat při plnění průzkumné mise více věcí zároveň. Pilot musí především sledovat dron, kromě toho ale také něco pozoruje, sleduje obraz, natočení dronu, rychlost a překážky. Cílem této práce je vytvořit asistenční algoritmus, který pilotovi ulehčí hlídání bezpečné polohy dronu. Práce prezentuje novou metodu revidující povely pilota tak, aby se dron držel „bezpečně blízko“ předem definované dráze letu. Pilot se tak může více věnovat plnění cílů mise, než přímému a bezpečnému řízení dronu.

Výstupem práce je nová metoda a její experimentální aplikace, která byla otestována uživatelskými testy. Pro testování byl do systému integrován simulátor AirSim, v němž uživatel splní nejdříve misi bez zapnuté korekce a poté s ní. V uživatelských testech byl cíl úspěšně uskutečnit průzkumnou misi. Piloti při letu bez korekce strávili 55 % času mimo definovanou bezpečnou oblast, při letu s asistentem takto strávili pouze 5 %. Testovací uživatelé uvedli, že jim řešení nejvíce pomohlo v udržení výšky a udržení na středu cesty.

Abstract

Drone pilots often have to keep an eye on multiple things at once while performing a reconnaissance mission. First and foremost, the pilot has to keep an eye on the drone, but in addition to that, he or she is also observing something, keeping an eye on the image, the drone's orientation, speed, and obstacles. The goal of this work is to create an assistance algorithm that makes it easier for the pilot to watch the safe position of the drone. The work presents a novel method revising the pilot's commands to keep the drone „safely close“ to a predefined flight path. This allows the pilot to focus more on accomplishing the mission objectives rather than directly and safely controlling the drone.

The output of this work is a novel method and its experimental application that has been user-tested. For testing, an AirSim simulator was integrated into the system, in which the user completes a mission first without correction enabled and then with it. In the user tests, the goal was to successfully complete the reconnaissance mission. Pilots spent 55 % of the time outside the defined safe area when flying without correction, and only 5 % when flying with the assistant. Test users reported that the solution helped them most in maintaining altitude and staying on the centre of the path.

Klíčová slova

bezpečný let dronem, letový asistent, polo-autonomní let, bezpečný průzkum dronem, korekce nebezpečných povelů pilota, AirSim a python

Keywords

safe drone flight, flight assistant, semi-autonomous flight, safe drone reconnaissance, correction of dangerous pilot commands, AirSim and python

Citace

FERENCZ, Adam. *Bezpečný průzkum dronem s využitím chytrého pohybu po trajektoriích.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Beran Vítězslav, Ph.D.

Bezpečný průzkum dronem s využitím chytrého pohybu po trajektoriích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. a že jsem uvedl všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Adam Ferencz
18. května 2022

Poděkování

Velmi děkuji panu Ing. Vítězslavu Beranovi, Ph.D. za jeho rady, které mi pomohly při tvorbě této práce. Poděkování patří také testovacím pilotům a Sáře Egersdorfové za pečlivou gramatickou korekturu.

Obsah

Kapitola 1

Úvod

V poslední době se zvedá zájem o využití dronů v různých komerčních sektorech. Příkladem je filmařství, bezpečnostní dohled, zemědělství nebo prodej realit. Jedním z případů užití dronu je průzkum větší oblasti. Příkladem může být kontrola technického stavu lanovky na horách. Uživatel potřebuje pravidelně kontrolovat stav lanovky a při letu se musí soustředit na pořizované video či fotografie.

Motivací této práce je navrhnout a implementovat způsob, jak pilota zbavit části zátěže pomocí asistovaného letu po bezpečné dráze. Pilot musí například vždy vědět, jak je dron natočený, protože pokud je natočený obráceně, má pilot v podstatě prohozené ovládání. Kromě letu samotného musí zkoumat přenášený obraz a například podávat hlášení o tom, co vidí. Prostředí, v němž se pohybuje může být v oblasti s lidmi, od kterých si pilot musí držet bezpečnou vzdálenost.

Existují aplikační řešení nabízející funkce s různými letovými módy pro usnadnění letu v různých situacích. Dále existují aplikace využívající rozšířenou realitu pro odlehčení zátěže pilota. Ty kladou důraz zejména na orientaci v prostoru, čímž nabízí pomoc při vícedronových misích.

Při použití navrhovaného řešení pilot neztratí pocit volného letu. Celou dobu řídí dron normálně, ale pokud se začne více soustředit na pozorování obrazu a vychýlí se z předem definované trasy, asistent ho usměrní zpět do bezpečné zóny.

Cílem práce je návrh metody pro korekci řídících povelů pilota tak, aby se dron držel na předem definované dráze letu. Součástí řešení je:

- reprezentace bezpečné dráhy a celého virtuálního prostoru,
- návrh GUI, jak pro tvorbu a editaci dráhy, tak i pro samotný bezpečný let a ovládání dronu,
- návrh vhodné architektury aplikace, aby bylo možné ji napojit na různé simulátory i reálný dron,
- vytvoření aplikace implementující korekci řídících povelů a provedení uživatelského testování.

Řešením je aplikace, která modeluje prostředí s bezpečnou dráhou a komunikuje s dronem (simulovaným, nebo reálným). V aplikaci je modul filtrující příkazy od uživatele. Ty jsou nejdříve zkontrolovány, případně poupraveny a až poté odeslány dronu.

Díky vzniklé aplikaci je možné volně ovládat dron na průzkumné misi a nemusí se věnovat největší část pozornosti hlídání bezpečné polohy dronu. V rámci řešení bylo navrhnuto

GUI, díky němuž je možné editovat dráhu letu, a který zároveň zobrazuje pilotovi návodné vizuální prvky při letu.

V teoretické kapitole 2 jsou rozebrané existující uživatelské aplikace na řízení dronu a dostupné simulátory letu dronu. Jsou zde představeny principy, díky kterým lze dron řídit. Na základě podrobné analýzy uživatelských potřeb je v kapitole 3 popsán princip celého řešení. V kapitole 4 se pak nachází konkrétní implementace experimentální aplikace, použité na uživatelské testování představené metody.

Kapitola 2

Pilotování a řízení dronů

V problematice řízení dronu (obecněji UAV - unmanned aerial vehicle, bezpilotní letoun) existuje již velké množství kvalitních řešení. Každé řešení se soustřední na různé typy využití v praxi. Práce řeší případ užití, kdy pilot přímo ovládá dron a zároveň se drží předdefinované trajektorie. Jedná se vlastně o mezistupeň mezi aplikacemi, které jsou uzpůsobené na manuální ovládání dronu a aplikacemi sloužících pro vykonávání automatických[1], či dokonce autonomních letů[2]. Nejdříve se v kapitole 2.1 budu věnovat mentální zátěži pilota, následně v kapitole 2.2 rozeberu používaná existující řešení, jenž souvisí s ovládáním dronu, nebo jeho pohybu po definované dráze. V kapitole 2.3 představím potřebnou terminologii okolo dronů a jejich řízení. Pro řešení bude potřeba správně reprezentovat bezpečné území, proto v kapitole 2.4 představím přehled relevantních možností reprezentace 3D objektů. Pro řízení dronu se používají osvědčené algoritmy jako je Kalmanův filtr a PID kontroler, jež budou vysvětleny v kapitole 2.5. V poslední sekci 2.6 provedu rešerši dostupných simulátorů za účelem výběru nejvhodnějšího z nich pro ověření funkčnosti řešení.

2.1 Mentální zátěž pilota

Mentální zátěž pilota je velmi komplexní pojem. Dala by se definovat jako míra zvýšení mozkové aktivity z důvodu komplikované situace při řízení dronu. Pro správné určení zdrojů mentální zátěže jsem provedl rešerši článků věnujících se tomuto tématu. Především se jedná o články, které prováděly testování pilotů, nebo zkoumaly jaké vlivy a strategie mentální zátěž sníží.

Torre et al. [5] se věnuje ve své práci testování jedenácti pilotů, jimž při různých úkolech v simulátoru měří vnímání pracovní zátěže. Testování prováděl pomocí testu AWT (Axon Workload Test), kde účastníci po splnění testovacích úloh vyplňují dotazník na základě kterého bude test dále vyhodnocován. Jedná se o adaptaci testu NASA-TLX [8] použitého například při testování snížení mentální zátěže při vizuální asistenci dronu jeřábu [11]. Test AWT obsahuje tyto kategorie:

- fyzická náročnost,
- časová náročnost,

[1] Automatický dron vykonává definovanou úlohu, i přesto je ale vyžadována pozornost pilota, aby v případě nutnosti dokázal opět převzít nad dronem kontrolu, viz Úřad pro civilní letectví.

[2] Autonomní dron létá zcela sám a musí být schopný řešit náhlé situace bez pomoci pilota. Autonomní drony podléhají přísným omezením, kde a jak je možné je použít, viz Úřad pro civilní letectví.

- úsilí,
- výkonnost,
- mentální náročnost,
- úroveň frustrace.

Uživatelé byli podrobeni pěti testovým situacím, v nichž museli zvládnout následující aspekty letu:

- plyn (řízení vztlakové síly letadla),
- přemístění nebo pohyb pohyb (všesměrový pohyb),
- ustálený nebo stacionární let (udržování polohy a výšky),
- pohyb vpřed (stabilizace letadla),
- přistání.

Při testech bylo využíváno různých módů a nastavení simulátoru. Například při úloze korigovaní plynu mohla být vypnutá funkce automatického držení letové hladiny.

Před testovacími lety uživatelé absolvovali kurz práce se simulátorem v rozsahu jedné hodiny. Uživatelům byly zároveň počítány chyby za nedokončení úlohy nebo způsobení neopravitelných poškození dronu. Při vyhodnocení senzitivity letu se jako nejdůležitější faktor ukázala mentální zátěž, která měla oproti ostatním faktorům nejvyšší korelaci s úlohou přistávání a počtem chyb. Účastníci, kteří vyplnili v dotazníku vyšší mentální náročnost, měli problémy s dokončením úloh a splnění trénovacích fází v simulátoru pro ně bylo časově náročné.

Mouloua et al.[12] zkoumají ve svém článku tři konkrétní problematické oblasti letu: **pracovní zátěž**, **povědomí o situaci** a **otázky týmové spolupráce**. Zabývají se těmito oblastmi z pohledu designerů letových systémů, kteří musí vhodně adresovat tyto lidské chybové faktory. Článek se zaměřuje na řízení bezpilotních letounů v kontextu americké armády. Proto rozlišují bezpilotní vzdušné letouny - UAVs (Unmanned Aerial Vehicles) a UAVs modifikované k boji (UCAVs). Výstupem jsou doporučení pro maximalizaci efektivity operátora UAV, identifikování klíčových lidských faktorů, ergonomická doporučení a implikace pro zefektivnění tréninku.

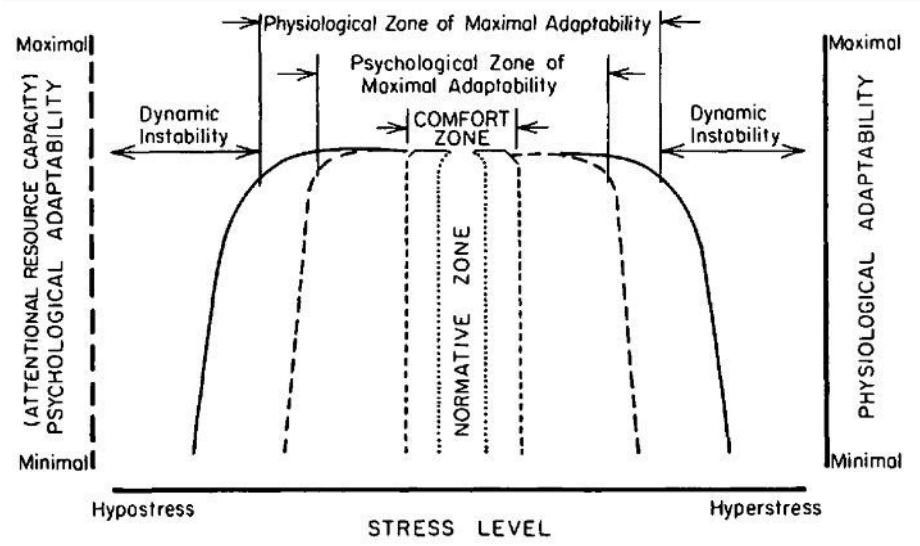
Povědomí o situaci znamená, že při určité úrovní automatizace by měl pilot mít dostatečný přehled o fungování systému, aby nebyl překvapen náhlými automaticky vykonanými úkony. Toho se dá například docílit oknem zobrazujícím detaily (příčiny) automatického rozhodování.

Koncept pracovní zátěže definují jako kombinaci vyžadovaných úloh, zátěžového faktoru a zodpovědnosti operátora dronu vůči těmto padavkům. Popisují, že při ovládání dronu jsou zátěžovými faktory: ovladač, čas letu, oblast a rychlost letounu. To, jakým způsobem pilot využije všechny řídící a informační prvky, záleží na jeho zkušenostech. Při velkém množství vjemů, které musí pilot vnímat, je možné využít strategii, kde se pilot snaží redukovat čas strávený na méně důležitých periferních prvcích. Popisují, že mentální zátěž lze odvodit dle intenzity srdečního tepu.

Ve své práci mimo jiné popisují automatické i manuální řízení. Při manuálním řízení je pilot plně zodpovědný za všechny systémové funkce letounu: letová poloha a rychlost, navigace, překážky a vyhýbání se překážkám, vyhledávání a rozpoznávání cílů, výběr a nasazení munice a další letové úkoly (normální i abnormální) vyžadují praktickou manipulaci s ovládacími prvky. Naopak plná automatizace funkcí systému může operátorovi znemožnit ovládání vyjmenovaných úkonů.

Pro UCAV operátory jsou nejvíce efektivní systémy, které dovolují optimální rovnováhu mezi prací operátora a operačními úlohami UCAV samotného. Tyto systémy se pak typicky nachází na pomezí dvou extrémů - manuálního řízením a kompletní automatizace. Při jízdě autem se tento způsob automatické pomoci již osvědčil například posilovačem řízení nebo protiskluzovým systémem brzd. V případě UAV se nabízí regulace rychlosti pomáhající řidiči udržovat asistovanou kontrolu nad letounem.

Jednou z popisovaných náročných situací je například nutnost splnění více úloh najednou v omezeném čase. Popisují také fenomén, kdy příliš vysoká nebo naopak příliš nízká zátěž způsobená nadprůměrnou automatizací (pilot funguje už jen jako pozorovatel) se značně negativně podepisuje na efektivitě pilota. Monitorování systému, kontrola chybových hlášení, vizuální kontrolovaní indikátorů stavu, spouštění počítačem asistovaných diagnostik a mnoho dalšího - tyto úlohy vyžadují plnou pozornost operátora a v průběhu času nabýt repetetivního rázu, což může a zároveň vede k degradaci výkonu a operačním chybám. Tuto myšlenku v článku převzali od Hancock a Warm [7]. Hancock a Warm se ve své práci věnují dynamickému modelu stresu a pozornosti. Na obrázku 2.1 je zobrazena hlavní myšlenka jejich práce. Pro získání největší pozornosti je potřeba, aby se stres pohyboval ve střední hladině. Hancock a Warm došli k závěru, že oba extrémy stresové škály vedou k degradaci pozornosti a stím souvisejícího výkonu. Ideálním stavem je tedy komfortní zóna.



Obrázek 2.1: Dynamický model stresu a udržení pozornosti [7].

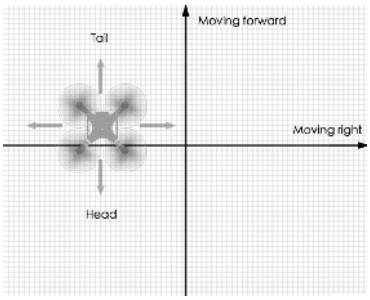
V této kapitole bylo rozebráno vnímání mentální a pracovní zátěže pilota. Analyzovaly se možnosti měření této zátěže (dotazník, chybovost, fyziologická odezva těla). Shrnulo se, jakými přístupy je vhodné k těmto lidským faktorům přistoupit. Byl zde vyzdvižen přístup nacházející kompromis mezi plně autonomním a plně manuálním přístupem k ovládání. Takový přístup dokáže získat maximální výkon od pilota bez jeho přetížení.

2.2 Existující aplikace pro řízení dronu

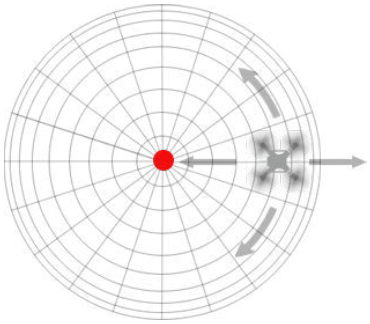
Doposud vznikla řada aplikací pro létání s drony, které se různými způsoby snaží pilotovi ulehčit plnění cíle mise. Při takové misi pilot často zápasí s obtížnou orientací v prostoru. Není pro něj lehké správně odhadnout rychlost dronu a vzdálenosti od překážek. V této kapitole představím aplikace, ze které jsou zajímavé s ohledem na řešený problém.

Obrázek 2.2: Waypoint mód - Snímek z oficiálního videa produktu DJI GO[3].

DJI GO je oficiální mobilní aplikace pro drony vyvinutá aktuálně dominující společnosti v tomto odvětví. Aplikace se dá považovat za state-of-the-art v designu GUI i v letových módech, které nabízí. Kromě základního letového módu, kde„dopředu“ znamená let ve směru čela dronu, aplikace disponuje i dalším tzv. inteligentní letové módy. **Waypoint mód** (viz obrázek 2.2) umožňuje automatický let po předdefinované dráze. Dron se pohybuje po předem definované dráze, případně vykonává přednastavené definované úkony. Výhodou je, že si pilot může dráhu připravit předem.



(a) Course Lock



(b) Home Lock

Obrázek 2.3: Inteligentní letové módy aplikace DJI GO. Převzato od DJI [4]

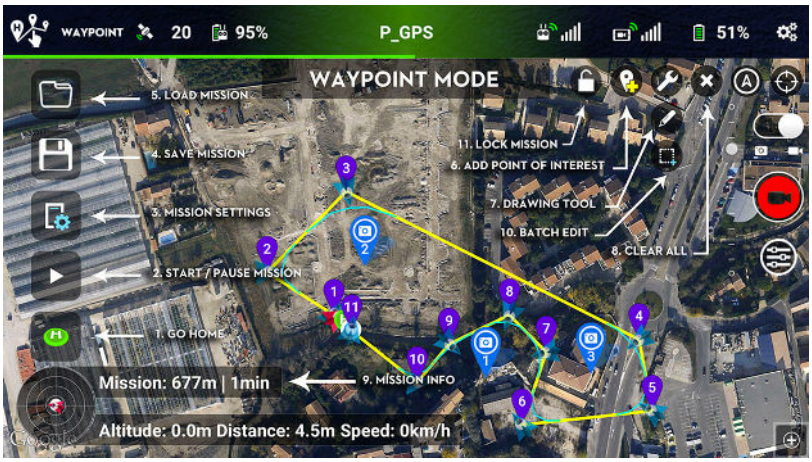
Další módy se už soustředí na různé situace, ve kterých se vždy hodí jiné ovládání. **Course Lock** viz obrázek 2.3a, kde „dopředu“ znamená na sever, je vhodnější při orientaci podle mapy než klasický mód, který je uzpůsoben k orientaci podle videa z kamery,

[3] https://www.youtube.com/watch?v=zvvbMxQ9Hj0
[4] https://developer.dji.com/mobile-sdk/documentation/

naopak u módu **Home lock** pokyn „páčkou dopředu“ vyvolá pohyb pryč od místa startu (viz obrázek 2.3b).

Aplikace **Litchi** je alternativnou DJI GO, poskytující webové rozhraní a mobilní aplikaci. Pro uživatele je dostupný centrální hub misí, ze kterého si může uživatel stáhnout již hotové mise a létat podle nich ve waypoint módu (viz obrázek 2.4). Litchi nabízí následující funkce a letové mody.

- FPV - let z pohledu první osoby.
- Waypoint - automatický let mise podle předdefinované trajektorie.
- Mission Hub - cloudová databáze dostupných misí ke stažení s možností sdílet své vlastní mise.
- Follow - následování určeného objektu, kde objektem je buď spárované mobilní zařízení, nebo marker v mapě.
- Orbit - kroužení okolo cílového bodu za definovaných parametrů (poloměr, výška, rychlost, směřování kamery, atd.)
- Virtual Reality - režim uzpůsobený pro použití VR brýlí.
- Focus - režim, ve kterém bude dron vždy natočený k bodu zájmu.
- Panorama - režim pro snadné vytvoření panoramatu.
- Track - pomocí detekce objektu bude sledován označený objekt a dron za ním automaticky poletí.



Obrázek 2.4: Aplikace Litchi - Waypoint Mise.[5]

Jak DJI GO a tak aplikace Litchi jsou určené pouze pro drony DJI. DJI GO je více spolehlivé, uživatelsky přívětivé a orientované na snadné vytvoření záběrů. Naopak Litchi je komplexnější, jelikož počítá s již znalým uživatelem.

Aplikace **DroCo** byla vyvinuta pro řízení více-dronových misí. Jedná se o experimentální vývojovou aplikaci, kterou vyvíjí výzkumná skupina robotiky na FIT VUT v Brně.

[5]Litchi.com: User Guide https://flylitchi.com/help

Drony se registrují do centrálního systému DroCo - jsou navzájem viditelné. Na obrázku 2.5 je vidět pohled na dron v režimu třetí osoby. Pilotovi se zobrazují nejen budovy v aplikaci korespondující s budovami reálného světa, ale zároveň zde vidí vše podstatné jako je mapa, video stream z dronu nebo další drony, s nimiž by měl kooperovat. Je také možné do rozšířeného světa přidávat různé interaktivní prvky misí nebo zóny - právě ony napomáhají s orientací a bezpečností. Aplikace je schopna komunikovat s ROS drony, DJI drony a nebo se simulovanými drony přes jejich vlastní protokol [10, 15, 16].

Obrázek 2.5: Ukázka z aplikace Droco [15], dron umístěný do virtuálního prostředí s promítnutým záběrem z jeho kamery.

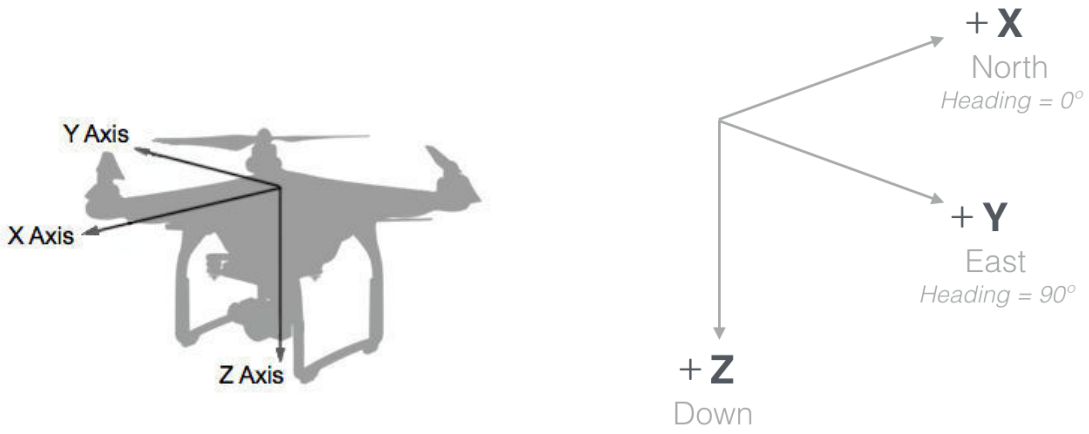
U existujících řešení chybí mezistupeň mezi automatickým, co nejpřesnějším letem po předdefinované trajektorii, a letem volným. Výhodou navrhovaného řešení je volnost letu pro pilota, kdy pilot letící v bezpečné zóně ani nepozná, že mu do řízení algoritmus zasahuje. Současně se ale pilot může spolehnout, že pokud by z nepozornosti zadal špatný řídící pokyn, bude zachycen a upraven.

2.3 Ovládání dronu

V této kapitole představím dva hlavní souřadné systému používané společností DJI pro řízení dronu. Vysvětlím, jak je možné, že pokyn pilota je správě namapován na jednotlivé motory a dron vykoná pohyb.

Souřadné systémy při řízení dronu

Pro popis pohybu UAV existuje velké množství **souřadných systémů**. V DJI SDK[6] (software development toolkit) se používají právě tyto dva typy: systém relativní vůči tělu vozidla (**body frame**) a systém (**world frame**) relativní vůči zemi. Souřadnicový systém podle těla vozidla má střed v těžišti tělesa (viz obrázek 2.6a). Osa x směruje ve směru čela dronu, osa y směruje doprava a osa z dolů. Světový souřadný systém se řídí pomocí světových stran. Tak jak je vidět na obrázku 2.6b, osa x směřuje na sever, osa y směřuje na východ a osa z směřuje dolu. Tento systém je také znám pod zkratkou NED z anglických slov North-East-Down. Každý z těchto systémů má své využití. U **world frame** systému nezávisí na natočení dronu, takže pilot může držet rovný kurz i při provádění rotace dronu, u **body frame** je natočení dronu pro pokyn rozhodující. Pilot se při řízení podle **body frame** řídí primárně kamerou dronu.



(a) Souřadný systém relativní vůči tělu vozidla: **body frame**

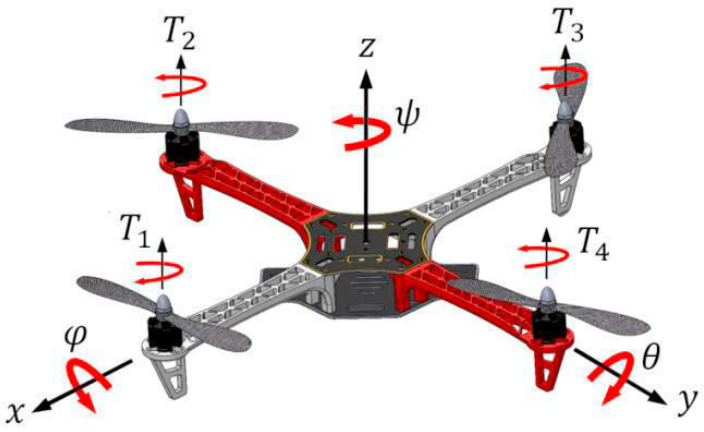
(b) Souřadný systém relativní vůči zemi: **world frame**.

Obrázek 2.6: Schémata k ovládání dronu převzatá z DJI MOBILE SDK Dokumentace[7].

Při **ovládání dronu** je stěžejní koordinace jednotlivých motorů. Pokud by ji měl zajišťovat uživatel, let by byl takřka nemožný. Jak je vidět na obrázku 2.7, dron samotný musí správně využít rotaci všech čtyř motorů, aby udělal cílový pohyb. Díky internímu kontroléru dronu pilot nemusí řešit jednotlivé motory zvlášť. Kontroler dronu je složen z několika PID regulátorů, které správným způsobem mapují pokyn pilota na výkon motorů, aby dosáhly tíženého stavu bez jeho překročení.

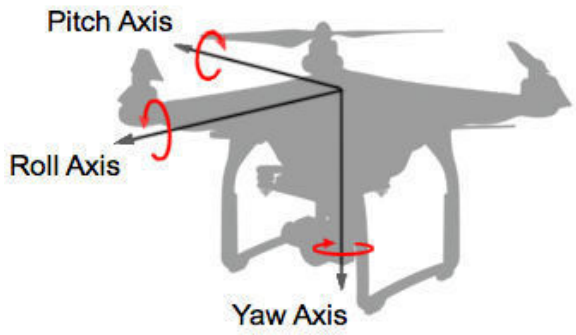
Základními pohyby dronu jsou stoupání a klesání. Pro vykonání dalších pohybů má dron k dispozici tři osy (viz obrázek 2.8) okolo kterých rotuje. Dron je schopen celkem 4 různých druhů pohybů:

[6] https://developer.dji.com/mobile-sdk/
[7] https://developer.dji.com/document/220a6eee-aa08-4f26-8235-29aeab9bb5ff



Obrázek 2.7: Možnosti rotací okolo os u quadrokoptéry. Převzato z [4].

1. **Pitch** je rotace okolo osy y způsobující předozadní náklon - pohyb dopředu nebo do zadu.
2. **Roll** je rotace okolo osy x. Umožňuje dronu letět přímo do boku.
3. **Yaw** je otáčení okolo osy z. Jeho důsledkem je změna směrování čela dronu. Polovina vrtulí se točí jedním směrem a polovina opačným. Díky tomu je možné, že dron samovolně nerotuje. Rotaci lze vyvolat přidáním výkonu vrtulím v jedné dvojice a ubráním té druhé.
4. Pohyb **throttle** není přímo závislý na konkrétní ose. Pokud je dron vyrovnaný, jedná se o stoupání, či klesání v závislosti na výkonu motorů. Pokud je ale dron nakloněný (Pitch/Roll), zesiluje se efekt horizontálního pohybu.



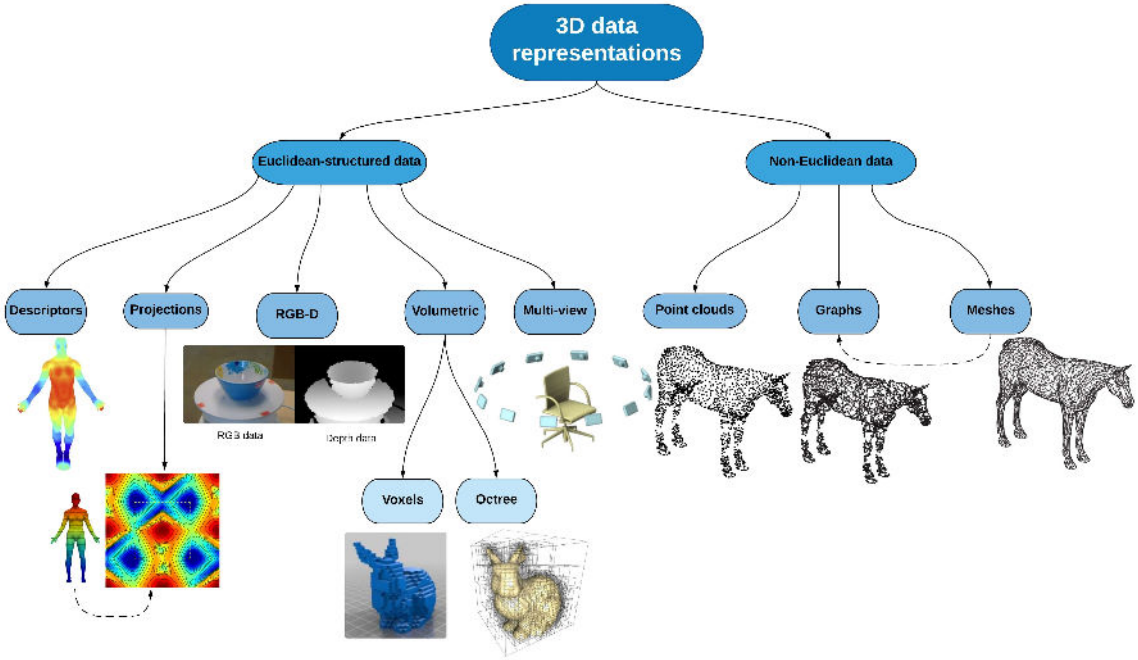
Obrázek 2.8: Osy rotačních pohybů dronu (převzato z DJI MOBILE SDK Dokumentace[8]).

[8] https://developer.dji.com/document/220a6eee-aa08-4f26-8235-29aeab9bb5ff

2.4 Přístupy k reprezentaci 3D objektů

Při vývoji algoritmu pro let s dronem je potřeba se na dron dívat jako na objekt ve 3D prostoru. Pokud je nezbytné využívat nějaký prostor, i ten je reprezentován 3D objektem. V této kapitole provedu rešerši možných reprezentací 3D objektu a popíši ty nejvhodnější pro navrhovanou metodu. Reprezentace vychází primárně z počítačové grafiky, kde se hojně používají. Důležitou vlastností, která bude třeba pro danou metodu, je výpočet kolize a výpočet vzdálenosti od okraje. To je zcela zásadní, jak v případě, že se dron musí držet uvnitř bezpečného objektu, nebo vně nebezpečné zóny.

Práce, které se nějakým způsobem dotýkají tématu 3D objektů vždy uvádí přehled možných reprezentací těles v prostoru. Ahmed et al. [1] rozdělují reprezentace na eukleidovské a ne-eukleidovské, viz obrázek 2.9. Eukleidovská data jsou strukturovaná do mřížky, kde je těleso popsáno ve společném souřadném systému. Ne-eukleidovské prostory nemají globální parametrizaci, nebo společný souřadný systém.



Obrázek 2.9: Různé reprezentace pro 3D data podle Ahmed et al. [1]. Rozdělení na Eukleidovské a ne-eukleidovské reprezentace.

Henderson [9] ve svém článku popisuje 3 reprezentační schémata: objemové, povrchové a kosterní. Popisuje mimo jiné například reprezentaci pomocí zobecněného válce (metoda šablonování), nebo zobecněné bloby[9].

Nejrozsáhleji přehled 3D objektů zpracovali na univerzitě v Princeton [6]. V přehledu zmiňují většinu relevantních reprezentací - již zmíněných i další. U jednotlivých reprezentací hodnotí následující vlastnosti: intuitivní specifikace, zaručená kontinuita, zaručená platnost, efektivní vykreslování, efektivní logické operace, přesnost, výstižnost a struktura.

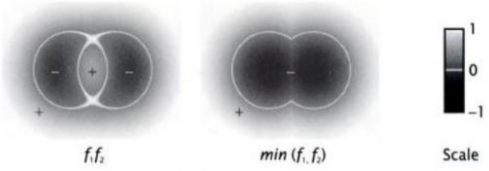
[9]Blob je možné si představit jako kulovité těleso. Jedná se o shluk částic okolo definovaného středu. Pokud jsou dva bloby blízko sebe, dochází k jejich slévání.

1. Surová data
   (a) **Point cloud** - nestrukturovaný set (mračno) bodů (získá se z dálkoměru, počítačového vidění atd.).
   (b) **Range image** - set 3D bodů mapovaných na hloubkovou mapu (získává se ze skeneru vzdáleností).
   (c) **Voxely** - jednotná mřížka objemových vzorků (získává se z počítačové tomografie, magnetické rezonance atd.).
   (d) **Polygon soup** - nestrukturovaný soubor polygonů (vytvořeno pomocí interaktivních modelovacích systémů).
2. Povrchem reprezentovaná tělesa
   (a) **Mesh (Síť)** - propojený soubor polygonů - obvykle trojúhelníků (efektivní vykreslování).
   (b) **Dělení povrchu** - definice sekvencí povrchů se zjemňovanou přesností.
   (c) **Parametrický povrch** - tenzorový součin spline plošek.
   (d) **Implicitní povrch** - body vyhovující: F(x,y,z) = 0 (zaručená spojitost, zaručená platnost, efektivní logické operace) .
3. Objemem reprezentovaná tělesa
   (a) **Octree** - binární rozdělení prostoru s pevnými buňkami (zaručená platnost, efektivní logické operace).
   (b) **BSP tree** - binární rozdělení prostoru s pevnými buňkami stejně jako u octree, jen se liší stromová struktura.
   (c) **CSG (Constructive solid geometry)** - hierarchie operací s logickými množinami (sjednocení, rozdíl, průnik) aplikované na jednoduché tvary (intuitivní specifikace, zaručená platnost a efektivní logické operace).
4. Vysokoúrovňové struktury
   (a) **Scene graph** - stromová struktura komplexních objektů, sjednocení objektů v listových uzlech (efektivní vykreslování a vysokoúrovňová struktura).

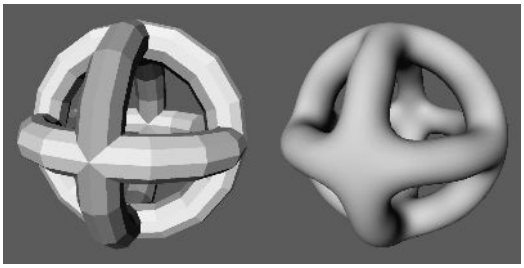
Vzhledem k zaměření článku popíši pouze relevantní reprezentace ze všech představených.

Voxely a octree voxelů

Objemová reprezentace pomocí voxelů rozděluje prostor na obsazené a neobsazené voxely. Kde voxel je nejmenší jednotka objemu v daném místě. Jejich nevýhodnou je, že reprezentují jak obsazené tak i neobsazené území, a proto mohou být neefektivní. Pro zvýšení efektivity je možné uspořádat voxely do struktury octree, čímž se umožní různá velikost voxelů. Voxely také nenesou konkrétní informaci o povrchu, ale pouze o objemu. Povrch by musel být odvozen [1]. Pro reprezentaci letových oblastí voxely nabídnou možnost zadefinovat velmi komplexní oblasti.



Obrázek 2.10: Slévání dvou 2D blobů definovaných funkcí $f_1 = |x - c_1| - r$ [3].



Obrázek 2.11: Interpolace polygonálního modelu pomocí implicitní funkce [19].

Mesh síť

Mesh síť reprezentuje objekt pomocí vrcholů, mezi kterými se tvoří vazby. Tři propojené vrcholy pak tvoří plošku (face), většinou trojúhelník. Jedná se o hraniční reprezentaci. Je definována pouze povrchem tvořeným trojúhelníky. Mesh síť dokáže velmi přesně reprezentovat komplexní povrchy. Tato reprezentace je vhodná především pro počítačovou grafiku. Při výpočtu kolize je ale nevýhodná. Pro komplexní těleso je potřeba zkontrolovat polohu vůči všem trojúhelníkům. V herních enginech se toto řeší obalovými tělesy, které objekt zjednoduší.

Implicitní plochy

Podle Turk et al.[19] je implicitní plocha definována implicitní funkcí počítanou nad $\Re^3$. Například jednotková koule by byla reprezentována rovnicí $f(x) = 1 - |x|$ pro $x \in \Re^3$. Body povrchu koule jsou pak takové, kde $f(x) = 0$. Pozitivní hodnoty této implicitní funkce se nachází uvnitř koule a negativní naopak vně. Důležitou třídou pro implicitní plochy je tzv. *meta ball* nebo *blob*. Jedná se o objekt definovaný Gaussovou funkcí $g_i(x)$. Díky *blobu* je možné řešit situaci, kdy se dvě takové koule dostanou blízko sebe. V tom případě dochází k jejich slévání a je potřeba určit jejich společný povrch. Ten se určí sumou Gaussových funkcí obou objektů a porovnání s hranicí t, jak je popsané v rovnici 2.1:

$$f(x) = t \sum_{i=1}^{n} g_i(x) \tag{2.1}$$

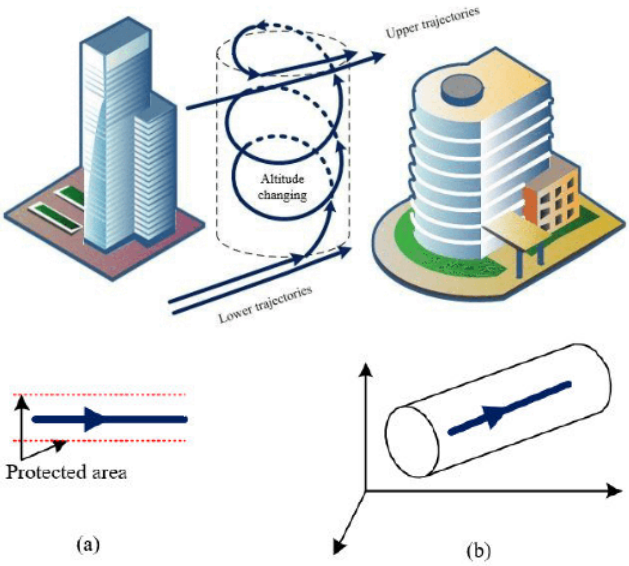
Implicitní funkce je vždy variantou vzdálenostní funkce od středu blobu. Podle jejího nastavení bude definováno chování při stékání blobů. Na obrázku 2.10 je vizualizace výpočtu implicitního povrchu zjednodušená do 2D. Dva bloby stékají do jednoho objektu, kde kladné hodnoty jsou vnější prostor a záporné hodnoty jsou vnitřní prostor. Hodnota 0 reprezentuje povrch tělesa. Implicitní funkce je definována $f_1 = |x - c_1| - r$, kde x je bod v prostoru, c_1 je střed blobu a r je poloměr. Neboli přesně ve středu blobu je hodnota -1, poté stoupá, na povrchu je 0 a následně narůstá do kladných čísel. Na obrázku 2.10 je ukázáno, že pokud by byl použit pouze součin, bude špatně určen průnik. Pokud se ale použije minimum, bloby se spojí správně. Kromě jednoduchých blobů je možné aplikovat implicitní povrchy na kostru, nebo i na polygonální síť (viz obrázek 2.11).

Zásadní výhodou implicitní funkce pro navrhovanou metodu je možnost slévání objektů pomocí vzdálenostní funkce. Pokud by například kostra byla tvořena přibližnou dráhou, která se na jednom místě kříží, je toto překřížení snadno řešitelné.

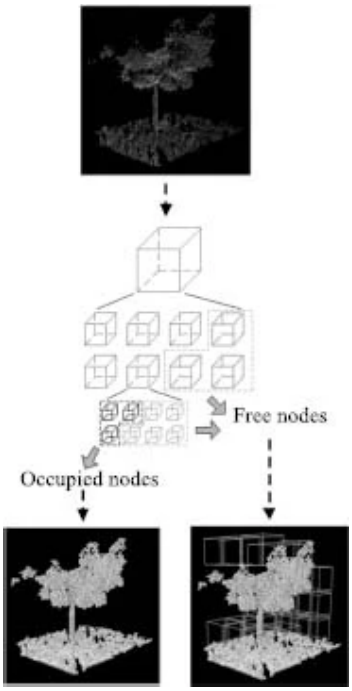
Reprezentace oblastí pro dron

Oblast řízení dronů a jejich autonomního pohybu je stále ve vývoji. Následující dva články se zabývaly tím, jak definovat dráhu či oblast pro let dronu.

Nguyen et al.[13] v článku řeší problémy okolo zavedení dronů do klasické městské infrastruktury. Navrhují, jak definovat trajektorie, pravidla provozu a případnou detekci kolize pro předcházení haváriím. Pro vytvoření bezpečné infrastruktury je třeba vytvoření letových koridorů. Ty mohou být v různých letových hladinách a pro přechod mezi nimi bude sloužit vertikální koridor (viz obrázek 2.12). Koridor samotný je definovaný jako válec o daném průřezu okolo centrální dráhy.



Obrázek 2.12: Návrh reprezentace koridorů ve vzdušném prostoru města, znázornění různých letových hladin a prostoru určeného pro stoupání. Koridor má stanovené hranice (a), ve 3D prostoru je reprezentován válcem (b) (převzato z [13]).



Obrázek 2.13: Struktura voxelů rozdělující prostor na volné a obsazené oblasti (převzato z [21]).

Druhou možností reprezentace dráhy bezpečného letu jsou voxely. Fei Yan et al. [21] ve svém článku představili možnosti plánování trajektorií ve 3D prostředí. Popisují jakým způsobem je možné vytvořit plánovací algoritmy pro bezpečnou cestu mezi překážkami. Co je především zajímavé, je jejich reprezentace prostředí. Pokud je prostředí naskenované, jedná se o 3D point cloud. Ten je ale pro možnost použití dalších algoritmů potřeba předělat na jiný model prostředí. Prostor reprezentují voxely (obsazené a volné), které tvoří tzv. mapu obsazenosti (occupancy map) a jsou dále uspořádány do octree struktury. Nejmenší možná velikost voxelu je pak mírně větší než velikost daného UAV. Prostor je tvořen voxely obsazenými nebo volnými, jak je vidět na obrázku 2.13. V článku je cílem vytvořit nejvhodnější cestu spojující cílová místa. Finální cesta je reprezentované sekvencí dotýkajících se volných voxelů.

2.5 Principy řízení a stabilizace dronu

Kromě vhodného definování dráhy bude potřeba navrhnout kontrolní algoritmus, který umožní přesně a stabilně řídit dron. V této kapitole představím postupy, které se věnují stabilnímu řízení dronu vůči dráze, nebo nějaké jeho cílové pozici. Budou zde představeny a vysvětleny principy Kalmanova filtru, PID regulátoru.

Kalmanův filtr pro drony

Kalmanův filtr je algoritmus používaný k přesnému určení stavu systému při zašuměných datech ze senzorů na vstupu (například k určení přesnější polohy dronu). Zkráceně, opakovaně zpracovává vstupní signál a porovnává ho s odhadovaným výsledkem podle fyzikálního modelu. Uvnitř si pak modeluje „váhu spolehlivosti“ obou možných stavů (model vs senzory). Tím je schopný vyrovnat výchylky, případně výpadky. Má mnoho aplikací, ale v tomto kontextu je nejvíce zajímavé určení polohy pohybujícího se tělesa. Následující rovnice vychází ze článku od Welch et al. [20], který popisuje základní, případně rozšířený, Kalmanův filtr. Článek popisuje Kalmanův filtr pro diskrétní čas, proto bude zachováno originální značení.

Cílem je tedy určit stav $x \in \Re^n$ v procesu s diskrétním časem, který je definován rovnicí:

$$x_{k+1} = A_k x_k + B u_k + w_k, \tag{2.2}$$

s měřením $z \in \Re^m$ které je

$$z_k = H_k x_k + v_k. \tag{2.3}$$

Náhodné proměnné w_k a v_k jsou nezávislé. Modelují šum a jsou tvořeny normálním rozložením. A je transformační matice stavového vektoru x_k a B transformační matice kontrolního vektoru u_k.

Pokud by se například jednalo o pohybovou rovnici. Tvar matic A a B se odvíjí od uspořádaní stavového vektor u x_k. Ten může být například $x = [d_x, d_y, v_x, v_y]$ pro 2D pohyb, dvojice (d_x, d_y) představuje polohu a (v_x, v_y) je rychlost. Pohyb má následující rovnici:

$$x_{k+1} = x_k + v_k \cdot \Delta t + \frac{1}{2} a_k \cdot \Delta t^2 \tag{2.4}$$

$$v_{k+1} = v_k + a_k \cdot \Delta t, \tag{2.5}$$

kde x je poloha, v je rychlost a a je zrychlení. Matice A zařídí, že do výsledného stavu přispěje rychlost $(v_k \cdot \Delta t)$ a matice B zařídí, že ve stavovém vektoru se poloha aktualizuje i podle zrychlení $(\frac{1}{2} a_k \cdot \Delta t^2)$ a že se rychlost změní podle zrychlení, viz rovnice 2.5.

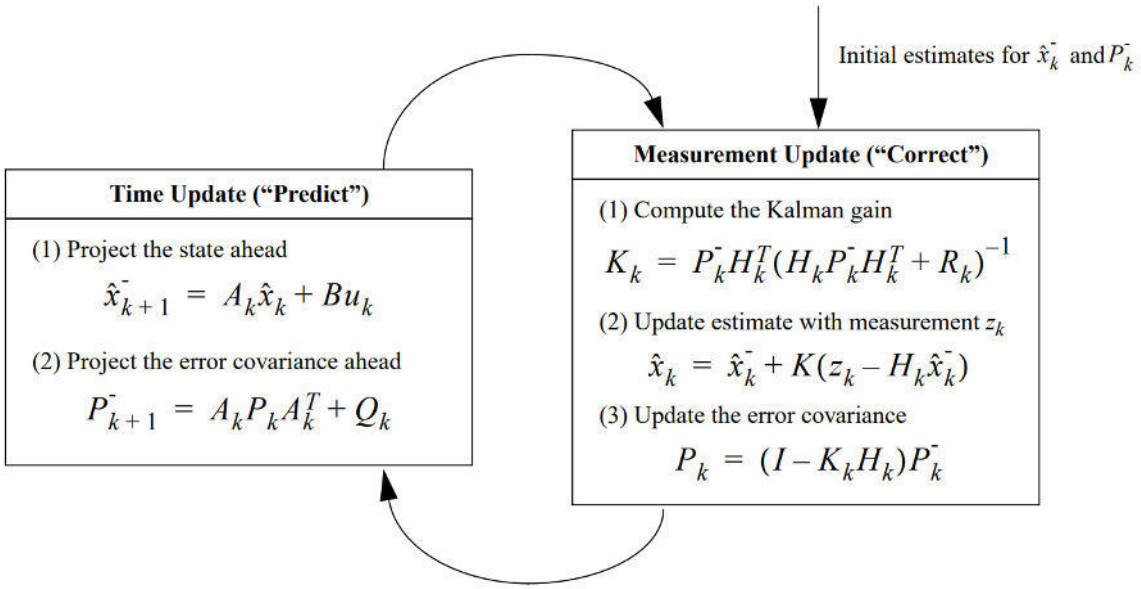
Kalmanův filtr (viz obrázek 2.14) se skládá ze dvou fází, mezi kterými neustále cyklí:

1. Aktualizace měření (korekce)
2. Časová aktualizace (predikce)

V následujícím textu se budu odkazovat na rovnice z obrázku 2.14.

Časová aktualizace (predikce)

Nejdříve se předpoví odhadovaná poloha x^-_{k+1}. Poté se vypočítá předpověď kovariance chyby P^-_k, kde Q_k je rozptyl normálního rozložení chyby predikce.



Obrázek 2.14: Kompletní přehled operací v Kalmanově filtrů (převzato z [3]).

Aktualizace měření (korekce)

První krok v bloku měření je výpočet K, což je ona „váha spolehlivosti“ zmiňovaná v úvodu. Rovnice jde „přehledněji“ přepsat následovně:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \tag{2.6}$$

H_k je transformační matice, H_k^T je její transponovaná verze, P_k^- je kovarianční matice chyby odhadu, R_k je kovarianční matice chyby měření. Pro aktualizaci měření je zásadní rozdíl $(z_k - H_k\hat{x}_k^-)$. Nazývá se *residuál* a určuje nesoulad mezi naměřenou hodnotou z_k a predikovanou hodnotou $H_k\hat{x}_k^-$. R_k a P_k^- ovlivní to, jak bude velké K_k a od toho se odvíjí význam *residuálu* pro výsledek. Pokud s R_k blíží nule, Kalmanův zisk K_k bere v potaz *residuál* více. Pokud se ale na druhou stranu P_k^- přiblíží nule, K_k se sníží a *residuál* bude zanedbán.

Vypočte se nový stav $\hat{x}_k$ s ohledem na měření z_k a aktuální hodnotu K_k

Nakonec se aktualizuje kovariance chyby měření P_k. I je matice identity, K_k je Kalmanův zisk a P_k^- je kovariance chyby odhadu.

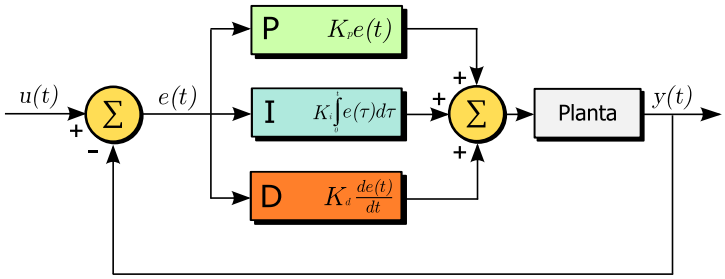
Kalmanův filtr se nachází ve většině dronů. Zároveň to, jak je navržen, může být inspirací pro návrh řešení.

Stabilizace Quadrokoptéry pomocí PID kontroleru

Jedním z nejpoužívanějších stabilizačních prvků v robotice je PID kontroler. Proportional-Integral-Derivative kontroler si klade za cíl na základě vstupních parametrů (aktuální chyba) co nejdříve ustálit stav objektu (v tomto případě dronu) na stavu cílovém. Výstupem PID kontroleru jsou pokyny pro aktuátory (akční členy - v tomto případě motory). PID kontroler má za cíl dosáhnout tohoto ustáleného stavu co nejrychleji bez zbytečné oscilace.

Trenev et al. [18] se ve svém článku věnují stabilizaci pohybu quadrokoptér podél poskytnuté trajektorie. Dron má letět určitou dráhu a jejich cílem je co nejpřesnější průlet. Dráhu definují jako výšku, odchýlením na ose X a odchýlením na ose Y, tyto složky jsou dále řešeny samostatně. Kromě toho řeší i cílové natočení. Pro každou z těchto složek používají PID regulátor a jejich cílem je vhodně reagovat na chybu vůči dráze a stabilizovat dron na cílové pozici.

Cílem **PID regulátoru** je dosáhnout cíleného stavu systému pomocí iterativního výpočtu.



Obrázek 2.15: Schéma PID regulátoru. Převzato od Arturo Urquizo[2].

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d_\tau + K_D \frac{de(t)}{dt}, \tag{2.7}$$

$$e(t) = x_d(t) - x(t), \tag{2.8}$$

kde $u(t)$ je kontrolní vstup (pokyn), $e(t)$ je rozdíl mezi cílovým stavem $x_d(t)$ a aktuálním stavem $x(t)$; K_P, K_I, K_D - proporční, integrační a derivační váhové koeficienty pro PID kontroler.

PID kontroler má následující strukturu:

1. Proporční člen P vrací výstup závislý na aktuální chybě. Vždy zde ale zůstane tzv. steady-state error. Ten má za následek, že upřesňovaní se zastaví zbytečně brzy s malou chybou.

2. Integrační člen I řeší upřesnění výsledku (steady-state error[10]). Akumulováním i malé chyby dojde k dalšímu upřesnění.

3. Derivační člen D se snaží predikovat budoucí chybné chování. Na základě derivace měnícího se stavu předchází budoucí chybě.

Výhodou využití PID kontroleru je možnost variability. Na danou situaci je vždy možné vyladit váhové koeficienty K_P, K_I, K_D, tak, aby rychlost reakce byla dostatečná a stabilizace adekvátně silná. Díky tomu lze dosáhnout pohotových reakcí na změny bez přestřelení cíle a s minimem oscilace.

[10]Steady-state error je chyba mezi aktuálním a cílovým stavem v systému v limitě, kde stav jde do nekonečna.

2.6 Simulátory

Při vývoji algoritmů pro řízení dronů je důležité nejdříve testovat na simulátorech. Reálný dron je velice nákladný a je tedy lepší neriskovat škodu. Zároveň simulátor umožní rychlejší vývoj. V následující sekci jsem zpracoval dostupné simulátory a jejich možnosti. Simulátory se liší v mnoha aspektech. Každý simuluje různé prostředí. Některé mohou mít překážky, některé létají ve volném prostoru a jedinou pevnou částí je zem, kde dron přistává. Důležitým aspektem výběru simulátoru je také způsob ovládání dronu a jak je v simulátoru modelována fyzika pohybu.

Posledním aspektem výběru je integrace a propojení s externími programy. Tato poslední informace je klíčová. Vhodný simulátor musí umožnit obousměrnou komunikaci s externím řídícím programem. Přijímat data o řízení, nechat se jimi ovlivnit a také posílat zpět polohová data, případně další jako je směr, rychlost a akcelerace. Zároveň, některé simulátory jsou zpoplatněné, případně je k nim potřeba získat licenci.

U simulátorů budou tedy posuzované následující vlastnosti:

1. letové prostředí,
2. ovládání dronu,
3. fyzikální model,
4. možnosti integrace.

DJI Flight Simulator

Jedná se o oficiální simulátor od společnosti DJI[11]. Aplikace simulátoru umí simulovat všechny drony od DJI. Létá se ve virtuálním prostředí, ve kterém se vyskytují překážky. Pro ovládání je potřeba zapojit dron, nebo alespoň ovladač. V základní verzi je simulátor omezený, se zakoupeným produktem se ale obvykle dostává licence. Běží na DirectX. Dron umožňuje simulovat v režimu HITL (Hardware In The Loop), kdy se do počítače připojuje fyzický kontrolér dronu, aby se dosáhlo největší podoby s reálným dronem. Testování pak funguje následovně:

Obrázek 2.16: Ukázka napojení Aplikace DJI Simuulator na reálný dron.

1. Dron se sundanými vrtulemi se připojí k počítači, na kterém je zapnutý simulátor.
2. Aplikace se připojí do ovladače a ten se spojí s dronem.

[11] https://www.dji.com/cz/simulator

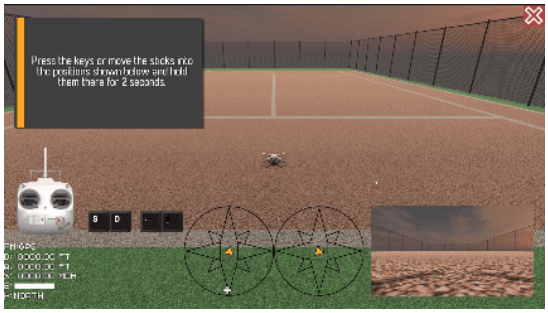
3. Veškeré vstupy z ovladače jsou posílané do dron.

4. Jedná se o hybridní simulaci, dron sice létá v simulátoru, ale některé úkony se dějí na reálném dronu, jako je například pohyb gymbalu, nebo pořizování fotek.

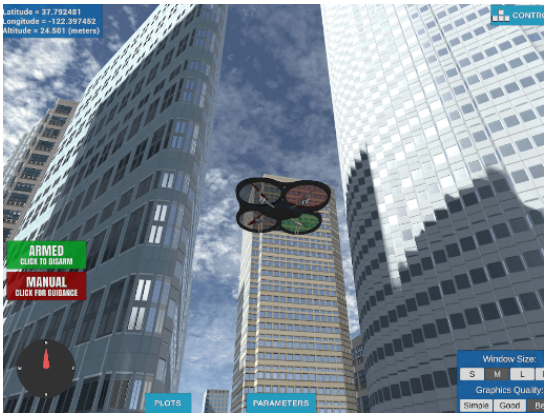
Aplikaci tedy lze použít pouze k testování oficiálních aplikací a aplikací využívajících DJI SDK, ve kterém je možné vyvíjet pouze pro Android, IOS a Windows aplikace. Tato uzavřenost značně limituje možnosti vývoje.

Zephyr drone simulator

Je simulátor určený především pro trénink pilotů dronu. Má placenou i volnou trial verzi. Podporuje různé ovladače dronů a je možnost ovládat simulovaný dron i klávesnicí. Nabízí virtuální letové prostředí (viz obrázek 2.17) včetně překážek a možnosti nastavitelného počasí. Bohužel konektivita k externímu programu není, a proto je pro účely této práce nevhodný.



Obrázek 2.17: Snímek prostředí Zephyr drone simulator převzatý z oficiálního videa[12].



Obrázek 2.18: Udacity drone simulator: snímek z ukázkové simulace[13].

Udacity Drone simulator

Tento simulátor (viz obrázek 2.18) je napsaný v Unity a je součástí kurzu Udacity, jenž pojednával o plánování trasy ve městě. Je připravený tak, aby se do něj dalo připojit přes mavlink (komunikační protokol) z Python skriptu. Dron v simulátoru se vyskytuje ve dvou módech. První je „guided“, kdy má plnou kontrolu nad pohybem vytvořený program. Druhý pak je „manual“, kdy se dron řídí podle pokynu ze šipek. U změny stavu ale nastává problém v tom, že nejde mixovat manuální řízení a řízení programem. Posledním problémem je, že fyzikální pohyb dronu není dostatečně realistický.

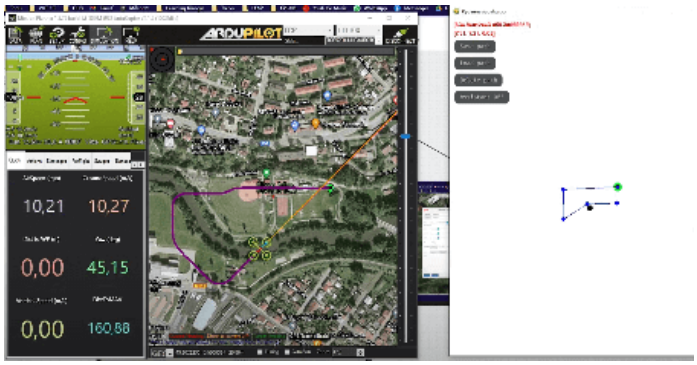
SITL od Ardupiot

Simulátor typu SITL (Software In The Loop) od Ardupilot[14] realisticky modeluje kontroler dronu. Jedná se o softwarovou alternativu simulování pomocí HITL (Hardware In The

[13] https://zephyr-sim.com/
[13] Udacity Drone simulator - https://github.com/udacity/FCND-Motion-Planning
[14] https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

Loop). Tento simulátor je možné v jazyce Python ovládat pomocí knihovny DroneKit[15]. Stav dronu je možné vizualizovat programem Mission Planner[16]jako 2D pohled na mapu a vizualizaci polohy, rychlosti a orientace.

Mission Planner má v sobě dokonce SITL vestavěný, což znamená, že lze pustit simulovaný dron přímo z aplikace. Vizualizace zde probíhá ve 2D mapě pohledem shora. Jedná se o prostředí reálného světa, nejsou zde ale překážky, pouze povrch země. Ostatní letové parametry jako je náklon, výška, zrychlení atd., jsou vizualizovány především textově, jak je vidět na obrázku 2.19. Z tohoto prostředí lze dronu dávat také základní pokyny (například RTL - návrat na místo přistání).



Obrázek 2.19: Snímek z testování vlastního programu se simulátorem SITL a vizualizací Mission Planner.

DJI Tello

Dron DJ Tello (viz obrázek 2.20) nabízí možnost napojit Python program na fyzický dron. Jedná se o malý jednoduchý dron od společnosti DJI určený spíše ke kreativnímu tvoření. Jeho výhodou je otevřené SDK[17] přístupné přímo z Pythonu. Slabou stránkou je absence GPS lokátoru a malá spolehlivost. Silnou stránkou je jednoduchost a možnost rychlého otestování programu na reálném dronu. Oproti velkým dronům se také v případě havarie riskuje řádově menší peněžní ztráta.

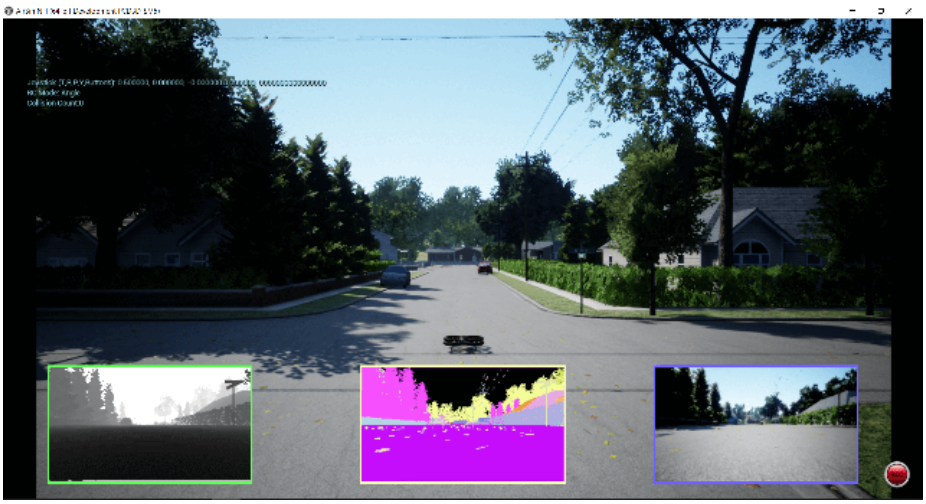
Obrázek 2.20: Tello Drone (obrázek převzat od DJI[18]).

[15] DroneKit - https://dronekit.io/
[16] Mission Planner Simulation - https://ardupilot.org/planner/docs/mission-planner-simulation.html
[17] https://github.com/damiafuentes/DJITelloPy
[18] https://store.dji.com/cz/product/tello?vid=38421

AirSim

Simulátor AirSim [17] umožňuje umístění simulovaného dronu do virtuálního světa. Je vyvíjen firmou Microsoft a slouží k vývoji autonomních vozidel (aut, nebo dronů). Celý simulátor běží na Unreal enginu. Poskytuje API pro Python, je tedy možné se na něj napojit pomocí knihovny AirSim. Ve virtuálním 3D prostředí má dron vlastní GPS souřadnice. AirSim má vlastí fyzikální model dronu (tedy i simulátor jeho kontroleru), který je dostačující. Dále je možné místo něj zapojit SITL (například od ArduPilot), nebo i HITL. Prostředí, ve kterém se s dronem létá, je zobrazeno na obrázku 2.21.

AirSim se svým základním modelem dronu je nejvhodnější variantou umožnující uživatelské testy. Hlavním kriteriem výběru simulátoru je možnost získání obrazu kamery, ideální model prostředí pro testování a dostatečně komplexní API.



Obrázek 2.21: Okno simulátoru AirSim [17], pohled třetí osoby na dron a tři malá okna kamery dronu (hloubková mapa, segmentace, základní obraz).

Kapitola 3

Korekce letu s využitím bezpečných trajektorií

Cílem práce je navrhnout a otestovat metodu, která umožní mapovat ovládání pilota v předem definovaném bezpečném území. Aby výsledná metoda mohla získat vstup, je nutné vhodně pracovat s definováním trajektorie a to buď pomocí vytvoření ze záznamu letu, nebo pomocí manuálního vytvoření dráhy. Bude potřeba vytvořit modul zodpovědný za modifikaci pokynů v závislosti na stavu dronu a prostředí. Navrženou metodu je potřeba uživatelsky otestovat. Z toho důvodu bude vytvořeno experimentální GUI s vizualizačními prvky. Pro zajištění co největší důvěryhodnosti testů bude nezbytné do návrhu architektury zapracovat napojení aplikace externí simulátor či dron.

Vzhledem k tomu, že aplikace řeší konkrétní uživatelský problém - letět bezpečně v předem definované oblasti, je zásadní vhodně definovat případy, kdy se řešení použije. V kapitole 3.1 uvedu příklady užití, které si metoda klade za cíl vyřešit. V této kapitole vysvětlím, jak je definován problém chytrého letu pomocí bezpečnostních trajektoriích, dále princip poloautomatického ovládání dronu pilotem i programem, který zasáhne do řízení jen v případě nutnosti.

Před řešením problému do hloubky představím koncept a navrhnu architekturu systému (viz kapitola 3.2). Nejdůležitější částí návrhu řešení je korekční metoda, kterou rozeberu v kapitole 3.4. Korekční metoda bude určovat výsledné chování dronu v různých situacích. Dále navrhnu aplikaci, díky které bude možné řešení testovat. Kvůli škálovatelnosti musí být architektura modulární, což umožní využití více různých dronů a simulátorů.

K aplikaci bude potřeba navrhnout vhodné GUI. Cílem práce není vytvořit GUI dokonalé pro koncového uživatele - jedná se o experimentální GUI, zobrazující prvky související s navrhovanou metodou. V kapitole 3.5 pak rozeberu návrh a účel jednotlivých prvků.

3.1 Náročné aspekty dohlídkových misí

V kapitole 2.1 jsem analyzoval práce věnující se mentální zátěži při řízení bezpilotních letounů. V této kapitole vyjdu ze získaných informací a aplikuji je na konkrétní případ užití, na který se tato práce zaměřuje - průzkumný let určité oblasti.

V oblasti může být výskyt lidí nebo jiných překážet. Pilot má za úkol v rámci mise v dané oblasti pořídit snímky či video záběry cíle. Přitom musí plnění celé mise proběhnout bezpečně. Hlavní motivací práce je pilotovi při plnění této mise snížit mentální zátěž tak, aby se mohl více soustředit na pozorování nebo například komunikaci s dispečinkem.

Obrázek 3.1: Záběry z úvodního videa aplikace DJI Simulator[1]zobrazující případ užití: Kontrola elektrického vedení.

Případy užití bezpečného letu

Všechny případy užití vychází z toho, že pilot oblast letu zná a je schopný v ní rozlišit bezpečnou a nebezpečnou oblast. Předpokládám, že by pilot dokázal při plnění své mise využít přesnou znalost prostředí, tudíž nebyl nucen plně se soustředit na bezpečnost letu. Ukázkou jsou následující situace:

- **Kontrola lanovky, nebo elektrického vedení** (viz obrázek 3.1). Takovou kontrolu provádí pilot pravidelně, například 1x týdně. Pokud by se při této činnosti mohl spolehnout, že nenarazí do překážky, protože bezpečné území si velmi pečlivě předdefinoval, výrazně by mu to zjednodušilo práci.
- Při velkých koncertech, fotbalových utkáních a demonstrací je náročné dohlížet na celý areál konání. Pro zefektivnění práce **při dohledu na velké zalidněné území** je možné použít dron. Řešení by mohlo pomoci pilotům létat vždy v bezpečné vzdálenosti od lidí.
- Ve filmovém odvětví by se řešení mohlo využít při **průzkumu vhodné scény pro natáčení**. Studia často posílají velmi drahé průzkumníky, aby tyto scény v zahraničí natočili a prozkoumali. Díky tomuto řešení by mohl být na průzkum poslán méně zkušený pilot, nebo by dokonce sám režisér mohl vzdáleně bezpečně ovládat dron. Asistent bezpečného letu by zajistil, že by průzkumný pilot létal pouze po předdefinované dráze a nemohl vyletět ven. To by ujistilo nezkušeného pilota například právě režiséra.
- **Bezpečný zkušební let dronem**: Poslední případ užití se odlišuje od ostatních. Drony poutají velký zájem i u dětí. V nesprávných rukou je dron schopný napáchat škody materiální či škody na zdraví. Pokud by ale dítě mohlo létat pouze po zadefinované dráze se zapnutým asistentem, byl by mu zprostředkován volný a hlavně bezpečný let.

Mentální zátěž pilota

Předtím než bude možné navrhovat prvky řešení, je třeba definovat, co konkrétně je mentální zátěž. Jedná se o situaci při plnění mise, vyvolávající stres, či působící vyčerpání. Mentální zátěž se dá změřit (viz článek [14]). Zde je výčet několika elementů, jež mohou být pro pilota náročné:

[1]DJI Simulator - Introducing DJI Flight Simulator - https://www.dji.com/cz/simulator/info

1. **Orientace v prostoru** - Vzhledem k volnosti, s jakou může dron rotovat okolo své vertikální osy, se často pilot dostává do situace, kdy je dron otočený a ovládání je tedy zrcadlové. Nezkušený nebo rozptýlený pilot v takovém případě může pokyn splést a nabourat.

2. **Vykonávání více pohybů najednou** - Určitě by bylo možné ovládat dron pouze jedním pohybem v jeden okamžik. Jednalo by se ale o neplynulý a neefektivní let. Pilot běžně udává několik pokynů najednou k vykonání komplexního pohybu. Například stoupá, letí dopředu a zároveň rotuje dron mírně vlevo, čímž způsobí plynulé zatáčení.

3. **Hlídání více prvků ovládání najednou** - Podobně jako při řízení auta, je třeba se na aplikaci GPS navigace dívat pouze letmo, i u dronů by dlouhé zahledění do mapy mohlo skončit kolizí. Pokud by se ale pilot chtěl řídit více mapou, kde má definovanou misi, potřeboval by něco, co mu zajistí, že v realitě nenarazí.

4. **Cíle mise a dispečink** - Pilot vetšinou nelétá jen pro zábavu. Většinou plní nějakou misi (cíl letu). Ten může mít například zadefinovaný na druhé obrazovce. Zároveň může komunikovat s dispečinkem, jenž mu dává další pokyny. Je zásadní, aby krátká nepozornost nezpůsobila nebezpečí.

Prvky snižující mentální zátěž

Existující řešení se tyto náročné aspekty letu snaží různými způsoby vyřešit, jak bylo popsáno v kapitole 2.1. Jaké jsou možnosti snížení mentální zátěže pilota?

- Zvolení **vhodného letového modu** s ohledem na situaci. Pilot se tak bude lépe orientovat v prostoru viz kapitola 2.3.

- **Vizuální prvky** - pilot by měl dostat informaci o své poloze, výšce, rychlosti a kudy má letět.

- **Semi-autonomní let** - uživatel létá samostatně, ale v případě hrozby kolize mu do řízení zasáhne algoritmus. Hrozbu kolize lze poznat pomocí senzorů, nebo definováním modelů prostředí.

- **Automatický let** by pilotovi ušetřil hodně práce. Proto také existuje spousta řešení snažící se drony automatizovat. Pro případy užití, kterým se práce věnuje, je ale zásadní možnost volného ovládání, protože mise je natolik komplexní, že ji nelze zcela automatizovat. Nevýhodou automatizovaného letu je situace, kdy je pilot znenadání nucen například přejít do manuálu. V takovém případě pilot přechází z jednoho extrému (automatický) do druhého (manuální) a to bez existující možnosti kombinace obojího.

Na základě rozebraných aspektů způsobujících mentální zátěž a na základě průzkumu v teoretické části jsem se rozhodl snížit mentální zátěž pilota redukováním důležitosti soustředění se na orientaci a pohyb drona v prostoru. Rád bych vše reprezentoval na až popularizačně-naučném, přesto názorném, příkladu vystihující podstatu věci.

Velmi zaneprázdněný manažer jde po visutém mostě nemající zábradlí. Na hlavě má sluchátka a telefonuje. V ruce drží telefon, do kterého něco zapáleně zapisuje. Aby nespadl z mostu, musí se každé dvě sekundy zastavit a zkontrolovat, zda se neblíží okraji. Pokud by uprostřed mostu bylo natažené lano, ke kterému by se manažer přivázal kladkou tak, aby

nebylo možné, že se dostane k okraji, mohl by se z devadesáti procent věnovat řešení své práce a směr svého pohybu upravovat pouze, když by ho lano tahalo.

V praktickém řešení bude ono lano definovat bezpečnou dráhu a úvaz s kladkou nahradí korekční síla pro pokyny pilota. Bude se tedy jednat o semi-autonomní let. Tím, že má pilot jistotu, že nemůže vyletět z bezpečné zóny, nebojí se vykonávat více pohybů najednou, má čas správně používat další ovládací prvky, může sledovat, jaké ho čekají cíle mise, a vše řešit s dispečinkem.

Uživatelské požadavky

Pilot při misi plní různé úlohy: sleduje scénu z určitého místa, pořizuje obrazový materiál, naviguje dron na cílovou pozici. Přitom mu hrozí kolize s případnými blízkými objekty. Předpokládá se, že pilot oblast předem zná, a proto je schopen vyznačit bezpečné i nebezpečné zóny. Příkladem může být pilot letící rutinní misi (vždy ji létá podobně), jistě by mu ulehčilo práci, kdyby nemusel řešit polohu tak přesně jako doposud. Zde přichází do hry navrhované řešení. Při jeho používání se pilot může nacházet ve dvou rolích:

- definice oblasti bezpečného letu (tvorba mise),
- chytrý let v bezpečném území.

Při vytváření mise, pilot musí určovat polohové (GPS) a výškové údaje. Pokud se na území bez problémů orientuje, dokázal by to udělat pouze za pomoci mapy. V některých situacích bude ale vhodnější, když proletí opatrně trasu nejdříve sám a nechá si zobrazit historii letu do mapy. Díky tomu pak dokáže lépe nadefinovat bezpečné území. Toto území si musí mít možnost uložit jako misi a později načíst.

Při letu je potřeba vycházet z toho, jak pilot létá průzkumné mise nyní a jak by se mu dalo pomoct. Průzkumná mise je manuální činnost, počítá se, že pilot bude reagovat například na pozorované objekty. Hlavním požadavkem na řešení je zachování manuálního ovládání při zvýšení bezpečnosti. Navrhovaná metoda by měla pilotovi poskytnout dostatek volnosti tak, jako kdyby létal volně. Dokud letí správně, tak by ani neměl poznat, že mu do řízení nějaký asistent zasahuje. Další situací, kterou uživatel často řeší je vzdálení dronu z jasného dohledu. V tu chvíli se pilot může spolehnout jen na stream z kamery a navigační prvky GUI. I přesto, že cílem práce není produkční GUI, musí se v experimentálním GUI vyskytnout takové navigační prvky (mapa s polohou dronu, orientace, rychlost a výška), aby jejich absence neovlivnila testování.

Může nastat situace, kdy uživatel s dronem na dohled a zároveň na jeho bezpečné dráze zjistí chyby ve vytvořeném bezpečném území. V takovém případě bude nejefektivnější možnost živé úpravy dráhy. Díky tomu uživatel ihned uvidí, jak dron na změny reaguje (pokud nechá asistenci zapnutou).

Volba letového modu vyšla z testování. V úvahu připadaly dva letové mody „Course Lock“ a „Body Lock“. Vzhledem k tomu, že se řešení zaměřuje na situaci, kde se létá mimo přímý dohled pilota na dron a blízko překážek, je pro orientaci zásadní obraz z kamery. Z toho důvodu byl zvolen letový mód „Body Lock“, který poskytne přirozené pohyby při sledovaní obrazu z kamery.

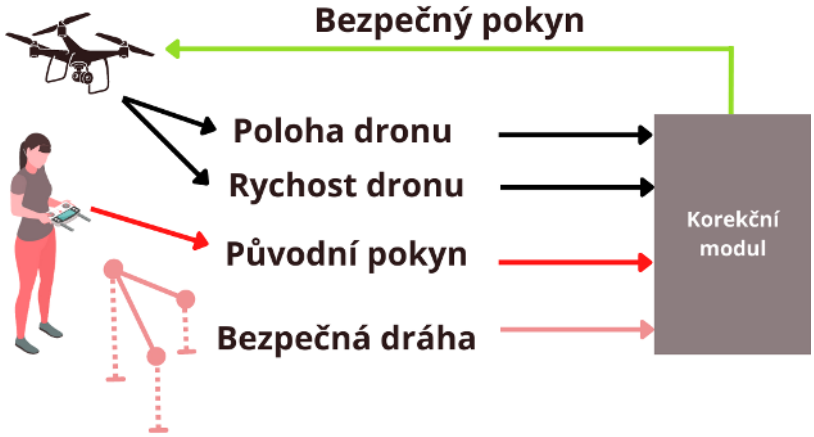
V této kapitole byly shrnuty situace pro jaké bude řešení navrhnuto. Představil jsem problémy, kterým pilot čelí, a požadavky na řešení, tak aby problémy vhodně adresovalo.

3.2 Princip letu s asistentem

V této kapitole navrhnu princip řízení dronu při použití korekční metody, kde se v celém systému daná metoda bude nacházet a jaké budou další zásadní části architektury korekčního systému. Koncept řešení a reprezentace bezpečného prostoru jsou dvě hlavní rozhodnutí, která spolu souvisí (musely se určit naráz). Proto již zde dopředu avizuji, že bezpečná oblast bude reprezentována dráhou a bezpečným rozsahem okolo ní. Podrobněji je pak toto rozhodnutí popsané v kapitole 3.3.

Koncept bezpečného řízení dronu

Při korekci letových pokynů pilota se nebude zasahovat přímo do řízení dronu (PID kontroler, který mapuje pokyn na jednotlivé motory). To bude bráno jako uzavřené řešení (vysvětleno bylo v kapitole 2.3), korekční metoda bude tudíž revidovat přímo pokyny pilota. Dron bude považován za hmotný bod, kde pokyn pilota znamená přímo zrychlení, které se aplikuje.



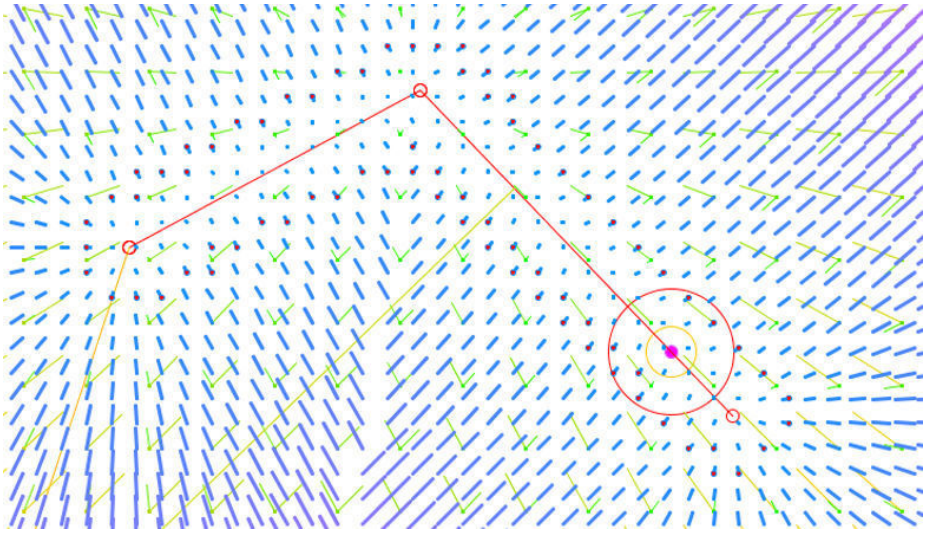
Obrázek 3.2: Základní schéma fungování korekčního modulu, který funguje jako filtr pokynů pilota.

Řešení bezpečného pohybu blízko trajektorie navrhuji jako filtr pokynů pilota. Tento filtr bude dále referován jako korekční modul, viz obrázek 3.2.
Vstupem jsou pokyny uživatele (ovladač), stav dronu (poloh, rychlost) a definovaná dráha. Výstupem této metody je bezpečný pokyn následně odeslaný dronu. Korekční metoda se na model světa dívá tak, že v každém místě dokáže určit míru nebezpečí pro dron a zároveň určit vektor směřující k nejbližší bezpečné poloze. Což se velmi podobá algoritmům pro výpočet implicitních povrchů, jež jsem rozebral v kapitole 2.4. V případě dronu, implicitní povrch znamená nepřekročitelnou hranici.

Na myšlenku implicitních ploch lze nahlížet i formou vektorového pole (viz obrázek 3.3). V každém místě prostoru lze určit vektor, který svou velikostí určuje míru nebezpečí a svým směrem ukazuje do nejbližšího bezpečného území. Tyto postupy jsou inspirací pro mé řešení (viz korekční metoda v kapitole 3.4).

Zásadní pro výpočet korekce je vektorové pole vytvořené díky vzdálenostní funkci.

[2]Processing: https://processing.org/



Obrázek 3.3: 2D vizualizace sil ovlivňujících výsledné řízení dronu na základě jeho polohy vůči bezpečné trajektorii. Obrázek z experimentálního programu napsaném v jazyce Processing[2]

Architektura korekčního systému

Nejproblematičtější chvíle plnění mise nastávají v případě, že má pilot sledovat více věcí najednou. Aplikace musí pilotovi ulehčit mentální zatížení tak, aby se mohl méně soustředit na let a více na průzkum, natočení kamery, gymbalu atd. Je třeba vytvořit aplikaci umožňující zpracování vstupů od pilota (z jeho ovladače) dříve, než jsou odeslány dronu. Při zpracování těchto vstupů je nezbytné, aby aplikace měla povědomí o tom, v jakém stavu se dron nachází či jaká je bezpečná dráha.

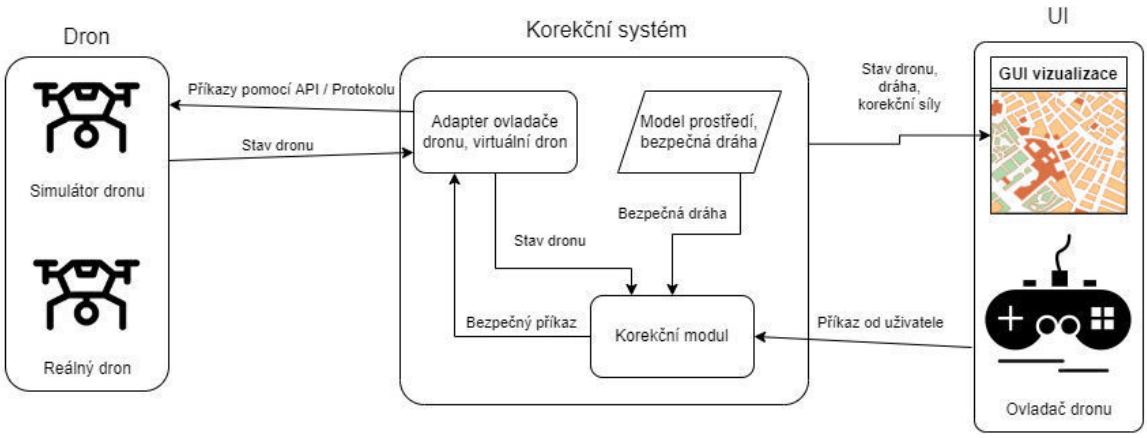
Řešení navrhuji jako systém, který bude mít následující architekturu (viz obrázek 3.4). Jádrem celé aplikace bude hlavní řídící program. Ten musí být schopný **přijímat uživatelské vstupy**, **vizualizovat stav dronu a prostředí**. Celé řešení **musí být modulární**, aby bylo možné nahrazovat především poslední komponentu, kterou je **dron (fyzický, nebo simulovaný)**. Každému simulátorů odpovídá jiný typ komunikace (protokol, API, knihovna) - řešení s tím musí počítat. Komponenty, které bude nutné navrhnout, uvnitř hlavního programu jsou **adapter ovladače dronu** (pro komunikaci s externím simulátorem), **korekční modul** (korekční metoda) a vhodně reprezentovaná **bezpečná oblast**.

Aby bylo možné řídit dron bezpečně vůči danému území, je třeba vytvořit návrh vhodné reprezentace bezpečného území. Bezpečné území a vhodná reprezentace stavu dronu poslouží jako vstupy pro navrženou korekční metodu.

Stav dronu je reprezentovaný následovně:

- poloha dronu v Eukleidovské souřadném systému (odvozená od GPS + nadmořská výška),
- vektor rychlosti dronu,
- orientace dronu (tři úly v osách pitch, roll, yaw - yaw je zásadní pro práci s pokyny).

Je nezbytné definovat, jaký typ dráhy pilot poletí a jak by měl dron reagovat. Létání pilota by mělo být v základu volné - nebude tedy přímo zakotvené k perfektní dráze. Bude



Obrázek 3.4: Blokové schéma architektury aplikace.

třeba jasně určit, co je a není bezpečné území. Navrhovaný korekční modul by měl pilotovi zasáhnout do řízení pouze tehdy, pokud by se jinak dostal do nebezpečí. Zároveň je zásadní, aby k zásahu do řízení docházelo postupně a zároveň byla predikována pozice dronu. Ten by se mohl dostat do situace náhlého, prudkého a hlavně nebezpečného brždění.

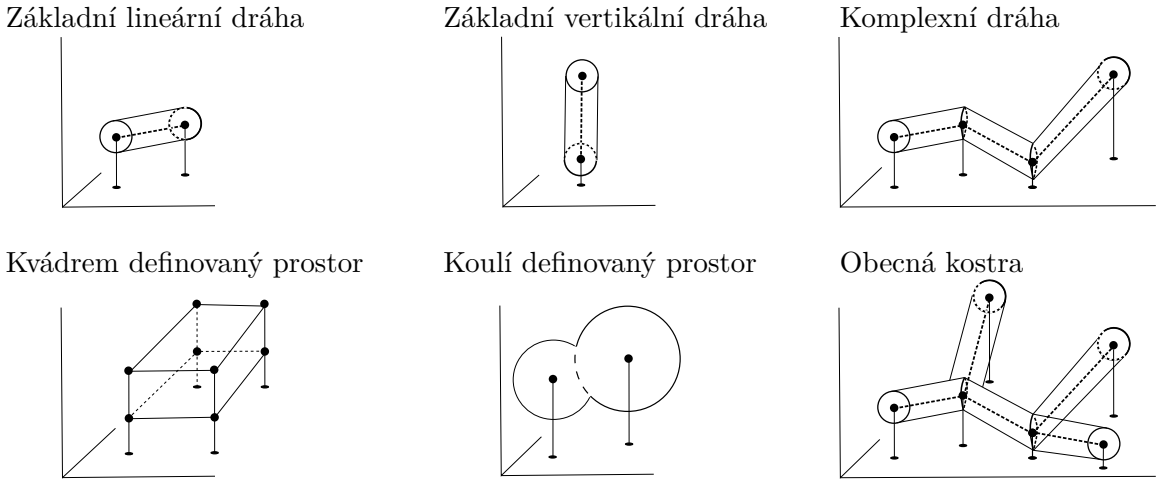
3.3 Reprezentace bezpečného prostoru

Základem celého systému je správně definovaná oblast pohybu dronu. V kapitole 2.4 jsem představil existující možné reprezentace prostoru. Může se jednat o oblast jednoduchou, nebo komplexní, zároveň by mělo být umožněno ji definovat pomocí dvou způsobů:

- Načíst uloženou dráhu z nějakého ideálně standardního formátu.
- Proletět danou bezpečnou cestu s reálným dronem, tu uložit a poté načíst do systému.

Systém pak takovou oblast použije pro kontrolu bezpečného letu. Při uvažování, jak taková oblast může vypadat, je možné brát v potaz následující typy drah a oblastí:

1. **Základní lineární dráha** - definována dvěma body, v každém bodě je definován průměr koridoru. Tvar koridoru postačí jako válec, pokud by ale bylo potřeba definovat ho jinou šířkou a výškou, mohla by se užít elipsa.
2. **Základní lineární vertikální dráha** - definována dvěma body, které jsou nad sebou - je třeba dát si pozor na změnu orientace.
3. **Komplexní dráha** složená z více bodů. Každý bod může mít jinou výšku a jinou šířku koridoru. Tato dráha by reprezentovala také záznam z letu dronu, pokud by se využíval pro definování. Šířku koridoru by bylo však potřeba stále doplnit.
4. **Kvádrem definovaný prostor** by dokázal pojmout větší objem prostoru. Byl by definován kostrou čtyř bodů. Otázkou je, jak vnímat stěny kvádru s ohledem na bezpečnou zónu. Logickou možností je, že stěny kvádru budou "odpuzovat dron", který se má držet uvnitř kvádru. Opačnou možností je kvádrem definovaná nebezpečná oblast, kdy dron létá vně kvádru. V případě potřeby definování letové plochy dronu by se přešlo k použití kvádru s velmi nízkou výškou.



Obrázek 3.5: Návrhy různých typů reprezentace bezpečného prostoru.

5. **Koulí definovaný prostor** je spíše podmnožinou lineární dráhy, jež je tvořena jedním bodem. Výhodnou je, že stačí pouze jeden bod a jeden poloměr koule pro definování oblasti relativně velké velikosti. Zároveň více překrývajících se koulí dokáže definovat libovolně komplexní útvar.

6. **Obecná kostra** je rozšířením komplexní dráhy. Podobně jako dráha je složená z více bodů propojených úsečkami. Rozdíl mezi ní a dráhou spočívá v tom, že body nemají dané pořadí - kostra se může různě větvit a vytvářet zatáčky, rozdvojení, ale i slepé uličky. Kostra značně komplexnější, proto nebude vhodná pro všechny představované metody.

Může dojít k situaci, kdy bude nutné reprezentovat také složitější oblasti. Toho se docílí metodou kombinující více definic těchto oblastí do sebe a jejich sjednocením tak vytvořit komplexní 3D mapu bezpečných drah a oblastí.

Ze dvou rozebraných přístupů k reprezentaci letové oblasti dronu (viz kapitola 2.4), jsem se rozhodl reprezentovat dráhu válcovými koridory. Ty jsem upřednostnil oproti voxelové reprezentaci , protože průzkumné mise jsou častěji vykonávány v delších a užších prostorech. Mnohem vhodnější možností je modelování dráhy pomocí trajektorie (množina bodů tvořící na sebe napojené úsečky). Takováto dráha bude jednoduše zadefinovatelná a je nejblíže logu dráhy letu, podle které se lehce vytvoří.

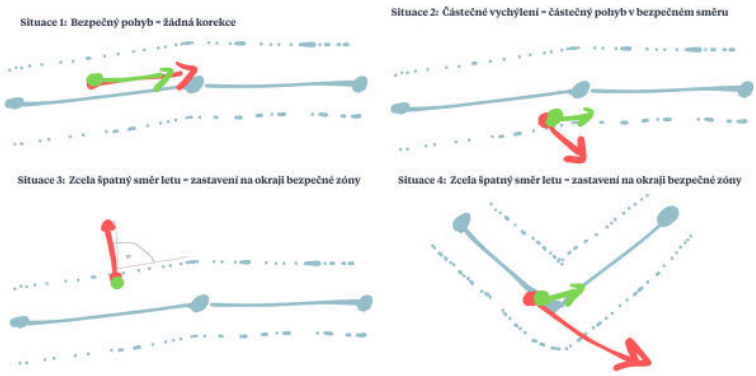
Bezpečná zóna S je definována jako ideální trajektorie dronu (množina úseček definována řídícími body) a maximální povolená vzdálenost od této trajektorie.

$$S = [p_0, p_1, p_2, p_3 ... p_n], \tag{3.1}$$

kde $p_0, p_1, p_2, p_3 ... p_n$ jsou jednotlivé řídící body. Každý bod obsahuje trojici: zeměpisná šířka, zeměpisná délka, nadmořská výška.

3.4 Korekční metoda

Pilot má tedy zadefinovanou dráhu a rozsah od ní, ve kterém je potřeba, aby ho metoda udržela. Pro lepší představu jsem zpracoval čtyři situace, ve kterých se pilot při řízení dronu může ocitnout. Obrázek 3.6 je úvahou nad chováním cílového řešení v různých situacích letu. V této kapitole navrhnu metodu, které tyto situace řeší. Červená šipka znázorňuje směr a rychlost dronu způsobenou pokyny pilota a zelená šipka znázorňuje, jaké by bylo požadované chování dronu při aktivním asistentovi (navrhovaná metoda).



Obrázek 3.6: Možné situace, do kterých se pilot může dostat při pohybu na dráze a jejich ideální řešení.

Při situaci 1 v obrázku pilot letí správně a zásah do řízení je zanedbatelný. V možnosti 2 částečně vybočuje z bezpečné zóny a pohyb je přeměněn na pomalé klouzání po okraji. Možnost 3 znázorňuje situaci, kdy pilot letí přímo pryč z bezpečné oblasti. V takovém případě je dron zastaven a algoritmus brání dalšímu pohybu. Poslední možnost znázorňuje dron, který v plné rychlosti letí směrem do zatáčky. Vhodným řešením v takovém případě je zpomalení a navázání na zatáčku pomalým pohybem.

Obecně lze popsat návrh metody následovně. Pilot dává povely dronu určující zrychlení daným směrem. Korekční metoda na základě polohy dronu vůči bezpečné dráze (definující bezpečnou zónu) a rychlosti dronu upraví řídící povely pilota tak, aby dron neopustil bezpečnou zónu.

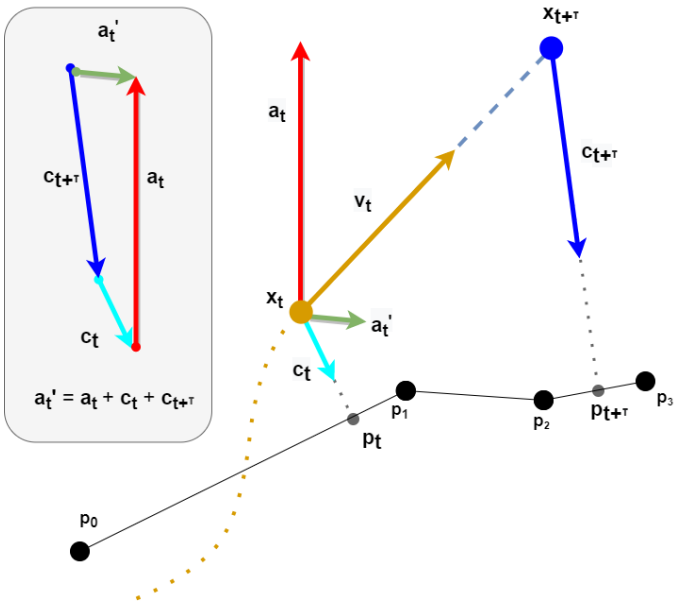
Existuje bod v prostoru s pozicí x_t. Ten představuje ovládaný dron. Letí rychlostí v_t a působí na něj zrychlení a_t. Aktuální zrychlení a_t určuje pokyn pilota.

Ve svém korekčním modulu (viz obrázek 3.7) pracuji s klíčovými proměnnými pro **polohu** x_t (oranžový bod), **pohyb** v_t (oranžový vektor) a **zrychlení** a_t (červený vektor). Dále se zde vyskytují dvě korekční síly, jejichž úkolem je usměrnit **původní nebezpečný pokyn** (zrychlení a_t). Světle modrá **lokální korekční síla** c_t vychází z aktuální polohy a tmavě modrá **prediktivní korekční síla** $c_{t+\tau}$ pak vychází z **predikované polohy** $x_{t+\tau}$. Výsledkem vektorového součtu $c_{t+\tau}$, $x_{t+\tau}$ a a_t je **vektor bezpečného pohybu** $a_t^{\cdot}$.

Vše probíhá v čase t, Δt uplynulý čas od posledního výpočtu stavu. Pro predikování polohy byla zavedena konstanta τ, kde $\tau > \Delta t$, $t + \tau$ je pak časový okamžik, pro který je počítána predikce.

Polohu a rychlost dronu vyjádříme rovnicemi pro změnu polohy hmotného bodu:

$$x_{t+1} = x_t + v_t \cdot \Delta t + \frac{1}{2} a_t \cdot \Delta t^2 \tag{3.2}$$



Obrázek 3.7: Korekce s ohledem na rychlost dronu: predikce polohy v čase $t + \tau$; grafický vektorový součet pro výpočet bezpečného pokynu (váhové koeficienty zanedbány). Oranžovou barvou je dron a jeho rychlost. Bezpečná dráha je reprezentována černými body.

$$v_{t+1} = v_t + a_t \cdot \Delta t, \tag{3.3}$$

kde Δt je uplynulý čas od poslední změny polohy.

$$c_t = r(L_2(x_t, p_t)) \cdot (x_t - p_t) \tag{3.4}$$

$$r(d) = \frac{(d-2)^3}{10} + 0.5, \tag{3.5}$$

kde p_t je nejbližší bod k x_t ležící na přímce nejbližšího úseku S, L_2 je Eukleidovská vzdálenost, r je normalizační funkce a c_t je korekční vektor.

Výsledné zrychlení (upravený povel pro dron) $a_t^{'}$ kombinuje jak původní povel pilota přepočten na zrychlení a_t, tak korekční vektor v daný čas c_t a především i korekční vektor $c_{t+\tau}$ odvozený na základě predikce polohy a rychlosti dronu v čase $t + \tau$.

Výpočet výsledného „bezpečného“ zrychlení lze popsat rovnicí:

$$a_t^{'} = w_0 \cdot a_t + w_1 \cdot c_t + w_2 \cdot c_{t+\tau}, \tag{3.6}$$

kde w_0, w_1 a w_2 jsou váhy a c_t je korekční síla pro aktuální polohu, $c_{t+\tau}$ je korekční síla pro predikovanou polohu a $a_t^{'}$ je výsledné zrychlení.

Grafické sčítání těchto vektorů je zobrazeno na obrázku 3.7, kde je vidět zásadní podíl korekčního vektoru $c_{t+\tau}$ na výsledku korekce.

3.5 Návrh UI pro korigovaný let

Pro testování metody je třeba vytvořit experimentální uživatelské rozhraní. To bude muset poskytnout možnost práce s bezpečnou dráhou a umožnit let s dronem. V této kapitole nejdříve popíši návrh vizuálního prostředí a účel jednotlivých prvků. Poté navrhnu, jakým způsobem se využije ovladač dronu (hardwarový kontroler).

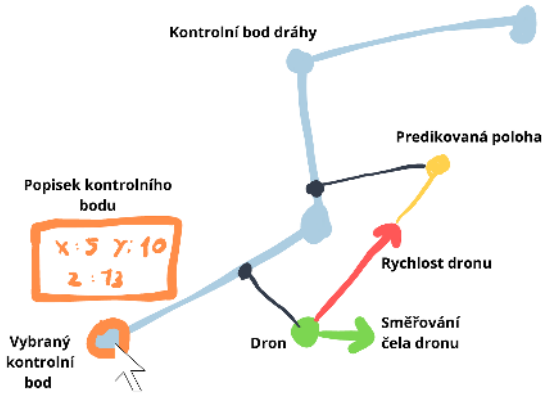
V **UI** se nachází dva typy prvků: **interaktivní** a **vizualizační**. Některé prvky jsou zde pro usnadnění letu, jiné pro editování bezpečné cesty. navíc jsou zde prvky, které slouží k vizualizaci chování korekční metody.

Interaktivní jsou zde určené pro umožnění zadávání uživatelských vstupů, nezobrazují sami o sobě žádnou komplexní informaci. Patři mezi ně tlačítka menu a přepínače v menu.

Vizualizační mají za úkol předat informaci o chování, nebo poloze dronu. Patří sem minimapa s bezpečnou dráhou a dronem, zobrazení výšky (pohled z boku), vizualizace korekčních sil, vizualizace páček a textové informace o poloze a rychlosti. Bezpečná dráha v minimapě jako jediný vizualizační prvek i interaktivní a umožňuje editaci dráhy.

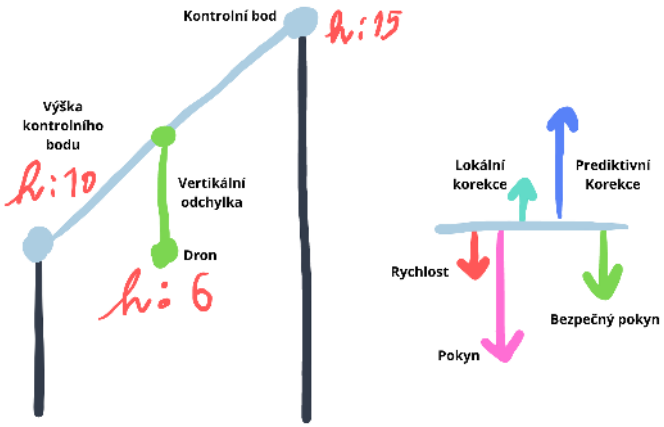
Nyní rozeberu jednotlivé prvky a popíši jejich účel a případně princip, pokud se jedná o složitější prvek.

1. **Tlačítka menu** - menu bude obsahovat následující tlačítka:
   - Načtení dráhy - otevře modální okno, kde uživatel vybere dráhu.
   - Uložení dráhy - uloží vytvořenou dráhu pro pozdější načtení.
   - Vymazání dráhy - vymaže aktuálně zobrazenou dráhu a začne vytvářet novou.
   - Přepínač asistenta - umožní zapnout a vypnout asistenta a zároveň zobrazuje zda je nebo není asistent zapnutý.
   - Přepínač nahrávání - umožní zapnout a vypnout nahrávání a zároveň zobrazuje zda je nebo není nahrávání zapnuto. Vypnutím nahrávání se záznam automaticky ukládá.
2. **Video z dronu** - mini pohled na video z přední kamery dronu. Bude možné ho mít skryté a nebo zobrazené. Kvůli užitečnosti při letu se bude přepínat přes ovladač.



Obrázek 3.8: Detailní nákres prvku minimapy v navrhovaném GUI.

3. **Minimapa** - pohled shora, který zobrazuje bezpečnou dráhu a polohu dronu. U **polohy dronu** je zde také zobrazen směr, kam je natočena příď a vektor rychlosti, který napoví, kam a jak rychle dron letí. Je zde také vyobrazená predikovaná poloha dronu (viz obrázek 3.8). Obě polohy jsou spojené jemnou linkou k zobrazenému nejbližšímu bodu. Prvek **bezpečné dráhy** v minimapě je zobrazen pomocí bodů, které spojují úsečky. Tyto bodu definují vlastnosti dráhy a je třeba s nimi dále manipulovat. Při najetí myší na daný bod se zobrazí jeho podrobnosti (poloha + výška). Při stisknutí a držení bude možné bod tahem přemístit. Výška bodu se změní kolečkem myši.



Obrázek 3.9: Pohled z boku - zobrazuje výšku; vpravo pak jsou síly vizualizující vertikální korekci.

4. **Pohled z boku** - zobrazení výšky dronu s přidanou hodnotou toho, že je zde vidět i nejbližší segment dráhy a výška jeho krajních bodů v porovnání s dronem (viz obrázek 3.9). Díky tomu uživatel dokáže poznat zda ho čeká stoupání či klesání a jak by měl uzpůsobit svou výšku, aby byla ideální.

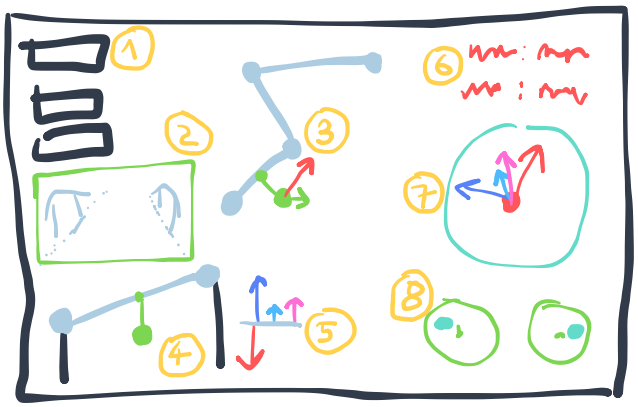
5. **Vizualizace korekce výšky** - pokud se dron začne oddalovat od ideální dráhy ve vertikálním směru, začne na něj působit korekce. Její sílu znázorní porovnání složky ve vektorech: lokální korekce, prediktivní korekce původního pokynu, rychlosti a výsledného pokynu. Síly nabudou podoby velkých proměnlivých sloupců vycházejících z nulové linie (ideální stav).

6. **Rychlost, GPS souřadnice a vektor pokynu (před a po korekci) jako textová informace** - tyto informace je potřeba zobrazit i textově, pro účely podrobnějšího chování celého systému. Například vektor pokynu před a po korekci jasně vypovídá o tom, jak se kdy systém chová. Pro běžného uživatele jsou pak relevantnější spíše rychlost a GPS.

7. **Vizualizace korekce polohy (v osách x a y)** - v osách x a y je korekce nejvíce znatelná na chování dronu. Aby bylo vidět, jak ovlivňují chování dronu jednotlivé korekční síly živě při řízení dronu, je třeba vizualizovat jejich směr a velikost. Na to slouží kruhová oblast, kde všechny síly vychází z jednoho bodu a je možné je porovnat.

8. **Vizualizace fyzických páček na ovladači** - jedná s eo poměrně intuitivní prvek - pilotovi dává vizuální informaci o tom, v jaké poloze se jeho páčky nachází.

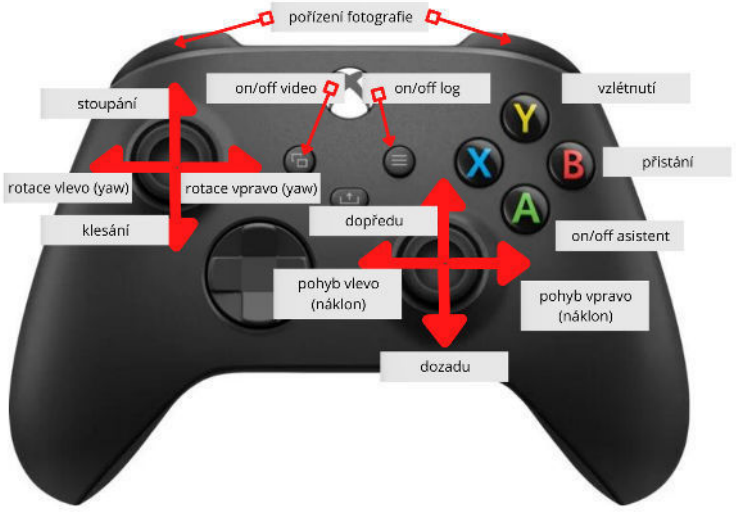
Všechny popsané prvky jsou zobrazené ve wireframe návrhu GUI (viz obrázek 3.10).



Obrázek 3.10: Wireframe experimentálního uživatelského rozhraní. Jednotlivé elementy GUI jsou popsané v textu.

Hlavním rozhraním pro řízení dronu je **ovladač**. Ten musí mít dvě páčky (joysticky), aby umožnil manévrování s dronem. V tomto případě je zvolen Xbox kontroler (viz obrázek 3.11), který umožní zadání velkého množství různých vstupů. Uživatel ho při letu samotném drží oběma rukama. Proto zde musí být dostupné, vše, co by mohl pilot při letu potřebovat.

Ovládání dronu je velmi specifické. Pro testování je potřeba, aby způsob ovládání vycházel ze standardních módů řízení a rozložení pokynů na ovladači. Ovladač samotný je také zásadní, protože řízení samotnou klávesnicí by nepřineslo dostatečnou přesnost. Existuje více možností, jak rozložit pokyny na ovladač. V tomto případě byl zvolen "Evropský režim", který se často používá u pilotů dronů, ale například u letadel není obvyklý [3]. Tento stejný mód je použit také u originální aplikace pro Tello dron[4], která byla inspirací.



Obrázek 3.11: Popis ovládacích prvků na ovladači Xbox.

[3] www.getfpv.com/learn/fpv-essentials/choosing-right-transmitter-mode/
[4] www.ryzerobotics.com/tello

Pilot ovládá let dronu páčkami. Levá páčka slouží ke stoupání a otáčený. Pravá páčka k předozadnímu pohybu a pohybu do stran. Tlačítka pro přepínač videa a logování jsou uprostřed, protože se používají jen na začátku či na konci letu. Oproti tomu tlačítko asistenta je pro svou důležitost velmi blízko pilotova dosahu - musí být schopen rychle reagovat. Pokud by nastala výjimečná situace a pilot by byl nucen rychle letět mimo bezpečnou zónu, musí mít tuto možnost vždy po ruce, aby asistent nezpůsobil náraz tím, že nepustí pilota udělat úhybný manévr.

Kapitola 4

Asistent bezpečného letu v blízkosti trajektorie

Pro implementování představeného návrhu je potřeba vytvořit systém, který je dostatečně modulární. To umožní, aby v průběhu vývoje a případně i v budoucím vývoji bylo možné měnit jednotlivé části. Aplikace se skládá ze tří hlavních částí Dron (simulovaný / reálný), Hlavní program a UI (GUI a letový fyzický ovladač). Každá z těchto částí je nahraditelná.

Nejdříve v kapitole 4.1 základní běh programu a hlavní struktury, se kterými se pracuje. Hlavní bloky programu mají svou reprezentaci, kterou popíši v kapitole 4.2. Do aplikace byly v průběhu vývoje integrovány dva simulátory (SITL a AirSim) a jeden reálný dron DJI Tello. V kapitole 4.3 rozeberu programové principy jejich napojení a případná specifika, se kterými je třeba při jejich použití počítat. Pro testování byla vyvinuta experimentální aplikace s GUI (viz kapitola 4.4). V kapitole 4.5 je popsán návrh experimentu, jehož průběh a výsledky jsou zpracovány v kapitole 4.6.

4.1 Cyklus aplikace a datové struktury

Každý objekt a výpočet má v algoritmu své místo. V této kapitole popíši, jak probíhá jeden typický cyklus programu a s jakými daty se při tom pracuje. Pokud se v následujícím textu bude vyskytovat `drone` jedná se o instanci třídy rozšiřující třídu `AbstractDroneModel`.

Program běží ve smyčce a vykonává následující kroky:

1. Aktualizuje si informace o dronu (poloha, rychlost, orientace).
2. Zpracuje stav ovladače, tedy vstupy od uživatele.
3. Pokud je zapnutý asistent, použije výše zmíněné aktuální hodnoty pro vykonání korekčního algoritmu.
4. Odesílá se pokyn dronu vhodným protokolem.
5. Je vykresleno GUI aplikace.

Veškeré zpracování vstupů je pomocí knihovny pygame. Na začátku cyklu jsou zpracovány veškeré události a je na ně reagováno. Kromě ovládání GUI myší, jde většina příkazů z Xbox ovladače. Nejzásadnější jsou páčky, které určují pohyb dronu. Jejich stav je uložen do pole 4 prvků `[pitch, roll, throttle, yaw]` a tvoří příkaz k pohybu. To, jakým způsobem se upraví původní rozsah páček $< -1, 1 >$, je určeno ve funkci `map_joystick_to_speed()`.

Pomocí modelu dronu aktualizuje svůj stav dotazem na simulátor a poskytne údaje o rychlosti, poloze a orientaci. Poloha je z GPS souřadnic transformována do Eukleidovského prostoru. Má tedy souřadnice `[x, y, z]` v metrech. Vzhledem k tomu, že používaný letový mód je relativní vůči natočení dronu, je potřeba rotovat vektor `command` okolo jeho počátku (poloha dronu) o úhel yaw pomocí funkce `drone.rotate_command_by_yaw([left_right, fwd_back])`.

Příkaz pro rotaci je společně s polohou dronu, strukturou dráhy a rychlostí dronu vstupem funkci: `corrector.adjust_command(location, velocity, command_speed, path)`.

Ve funkci `adjust_command` se vykoná představený korekční algoritmus, jehož výstupem je bezpečný příkaz. Ten je odeslán pomocí `drone.move(command)` do simulátoru, nebo dronu. Na základě všech aktuálních hodnot je vykresleno GUI.

4.2 Objektový návrh programu

Řešení je implementováno v jazyce Python[1]. Hlavní program běží ve smyčce, kterou inicializuje knihovna PyGame[2]. Každý cyklus smyčky je vykreslen, komunikován s dronem a je vykresleno GUI. Externí dron (simulovaný nebo reálný) má vždy jiné komunikační rozhraní. V kapitole 4.3 popíši integraci jednotlivých nástrojů, které jsou zásadní pro vývoj aplikací s drony a byly použité.

Dalším krokem při tvorbě programu je vyjít z návrhu a transformovat ho na konkrétní implementaci. V této kapitole popíši, jakým způsobem byl program dekomponován do objektového návrhu. Vysvětlím, za co jsou dané třídy zodpovědné a jaké jsou jejich nejdůležitější metody. Popíši zde, i třídy, které bylo potřeba vytvořit navíc oproti návrhu, aby program mohl správně fungovat.

Jak bylo popsáno v návrhu, základní bloky v hlavním programu jsou:

- adaptér ovladače dronu,
- korekční modul,
- model prostředí.

Pro navrhnuté bloky systému byly implementovány následující třídy:

Třída `Waypoint`

Pro udržení polohy bodu by stačila jedna reprezentace a to GPS + nadmořská výška. V programu je ale potřeba pracovat ve více souřadných systémech - vzdálenosti se dobře počítají v Eukleidovském systému a vizualizace je zase 2D. Proto si objekt `waypoint` uchovává všechny tři typy reprezentace, jelikož jejich převod stačí udělat pouze jednou při změně, ale používají se při každém cyklu.

Tři reprezentace jsou:

1. Zeměpisná: `latitude`, `longitude`, `altitude`
2. Eulerovská: `metres_x`, `metres_y`, `metres_z`
3. Visualizační: `visual_x`, `visual_y` (v pixelech)

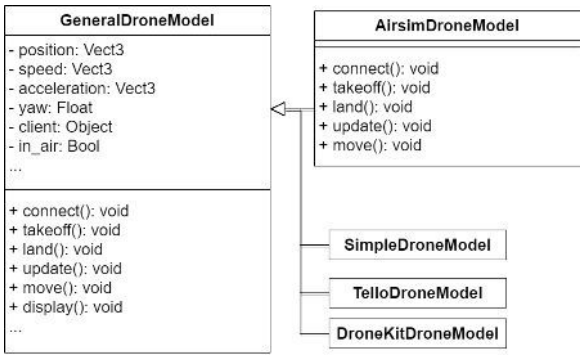
[1] https://www.python.org/
[2] https://www.pygame.org/

Třída `Path`

Její objekt zodpovídá za reprezentaci prostoru a práci s ním. Udržuje si pole `Waypointů`, dále ví, se kterým je aktuálně manipulováno v rámci GUI. Umožňuje načítání, ukládání operace a odstranění dráhy. Obsahuje metodu `get_segments()`, která místo seznamu bodů vrátí seznam úseček, jedná se o jeden ze vstupů korekční metody. Je zde implementováno vykreslení dráhy.

Třída `AbstractDroneModel`

Tato třída je pro program stěžejní. Z blokového schématu v návrhu implementuje funkcionalitu adaptéru ovladače dronu. Stará se tedy o veškeré odesílání pokynů do externího simulátoru a následné získávání stavu simulovaného dronu. Jak je vidět na diagramu tříd 4.1, třída je implementována jako abstraktní. Obsahuje paletu metod potřebné k fungování programu bez ohledu na to, jaký dron, nebo simulátor je použit. Většinu metod si tedy konkrétní `ConcreteDroneModel` implementuje sám. Nezměněné zůstávají pouze vizualizační metody. Každý `ConcreteDroneModel` tedy může mít například různé mapování citlivosti ovladače.



Obrázek 4.1: Třídy modelů dronu, kde konkrétní třída funguje jako adaptér pro komunikaci s externím simulátorem/dronem. Jednotlivé názvy tříd jsou odvozené od různých typů simulátorů nebo dronu použitých v projektu.

Kromě komunikace s externím dronem model obsahuje všechny získané informace o stavu dronu, aktualizující se na začátku každého cyklu voláním metody `update()`. Pro celý program je tím komunikace zapouzdřena a pracuje se s modelem, jako kdyby byl dron (hmotný bod), který dostává pokyny k pohybu.

Konkrétní implementace zásadních funkcí popíši v kapitole 4.3 u každého nástroje zvlášť.

Třída `Transformer`

Jak už bylo zmíněno u třídy `Waypoint`, v celém řešení se operuje s více souřadnými systémy. Pro jejich vzájemné převody a uchování přiblížení GUI slouží třída `Transformer`. Nejzásadnější převod, jenž třída implementuje, je mezi GPS souřadnicemi a Eukleidovským prostorem pro výpočet vzdáleností. Převod se provádí nejdříve z GPS souřadnic na UTM (Universal Transverse Mercator)[3] pomocí knihovny utm[4]. Všechny objekty a metody pracující se souřadnými systému využívají svůj objekt `transformer`.

[3] https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
[4] https://pypi.org/project/utm/

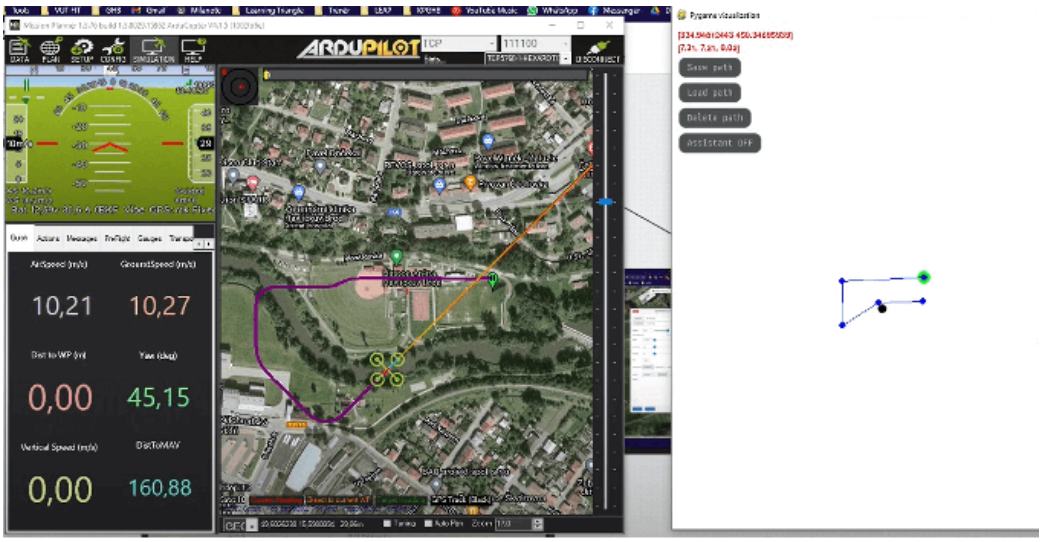
Třída Logger

Některé simulátory (např. AirSim) poskytují možnost logování, jiné nikoliv (SITL). Tyto logy ale nejsou dostatečné komplexní a především neobsahují informace o poloze vůči bezpečné dráze. Z toho důvodu je zásadní, aby existoval vlastní `Logger` zaznamenávající vše podstatné. Při návrhu jeho formátu jsem vycházel z logů společnosti DJI formátu a doplnil vlastní logované atributy.

4.3 Integrace řídicího modulu a simulátoru

V průběhu vývoje byla aplikace napojena na dva externí simulátory a jeden reálný dron. V této kapitole popíši, jaké byly výhody a nevýhody použití těchto nástrojů. Ke každému nástroji vysvětlím jeho možnosti a pomocí základního kódu ukážu principy ovládání dronu v daném nástroji. Uváděné fragmenty kódu jsou zjednodušenou variantou bez zabalení do objektového návrhu. Slouží především k lepšímu vytvoření představy o tom, jak se s nástroji pracuje a co umí.

Pro testování musel být zvolen takový simulátor, do něhož je možné zasahovat zvenčí. Je třeba, aby měl dostatečně realistický vlastní PID řídící kontrolér. Z pohledu testování aplikace na uživatelích je nutné, aby nabízel 3D prostředí a možnost získání záběru z kamery dronu. Účelem aplikace je totiž především být schopný bezpečně navigovat dron vzdalující se z dohledu (tedy kamera je hlavní orientační prvek) a efektivně s ním pořídit snímky nebo záběry.



Obrázek 4.2: Testování se simulátorem SITL v Mission Planneru.

DroneKit, SITL a Mission Planner

SITL je program, ve kterém běží fyzikální simulace modelu reálného dronu. Pro komunikaci se SITL se používá standardní MAVlink protokol. V tomto případě byl používán SITL, který reprezentoval Ardupilot[5] hexacopter. MAVlink protokol lze používat sám o sobě,

[5] https://ardupilot.org/ardupilot/

ale pro Python existuje knihovna Dronekit [6]. Dronekit nabízí nadstavbu nad protokolem a vytváří MAVlink zprávy za něj.

SITL je možné zapnout z příkazového řádku, přímo v programu pomocí Dronekit a nebo pomocí aplikace Mission Planner[7], viz obrázek 4.2. Jedná se o oficiální GCS (ground Control Station) pro Ardupilot. Jednou z jejích funkcí je spuštění SITL a rovnou se na něj napojit.

Pokud je předpoklad puštěného SITL, pak základní demopříklad v DroneKit bude vypadat následovně:

```
1  vehicle = connect(connection_string, wait_ready=True)
2  vehicle.simple_takeoff(aTargetAltitude)
3  while True:
4      (...)
5      # Get and transform location.
6      location_gps = [vehicle.location.global_relative_frame.lat, vehicle.location.global_relative_frame.lon]
7      location_xy = convert_latlon_xy(center, location_gps)
8      altitude = vehicle.location.global_relative_frame.alt
9      location = np.array([vehicle.location_xy[0], vehicle.location_xy[1], altitude])
10
11     # Get velocity.
12     velocity = np.array(vehicle.velocity)
13
14     # Calculate save command.
15     save_command_speed = corrector.adjust_command(location, velocity, command_speed, path)
16
17     # Send Command to the drone.
18     vehicle.set_velocity_body(save_command_speed[1], save_command_speed[0],
           save_command_speed[2])
19     (...)
```

Existují dvě hlavní možnosti, jakými jde SITL ovládat[8]:

- posílání příkazu, aby dron letěl na konkrétní místo,
- odesílání konkrétního rychlostního příkazy.

Vždy je potřeba zvolit, vůči čemu jsou příkazy relativní, souřadné systémy jsou podobné „Course Lock“ a „Body Lock“ od DJI.

Pro představenou metodu byly použité příkazy pomocí rychlosti, právě těmi je dron ovládán pilotem při klasickém letu. Všechny příkazy v DroneKit jsou asynchronní. Pro získávání informací o dronu je třeba nastavit callback funkci, například když se pošle dronu GOTO („leť na určité místo“) příkaz, je potřeba zjistit, že už tam dron doletěl.

SITL byl používán v první části vývoje, dokud nebylo nutné zobrazovat hezkou 3D scénu simulátoru. Pak by vystřídán simulátorem AirSim.

Tello

Aplikaci lze využít také s fyzickým dronem DJ Tello pomocí dostupného SDK [9]. Ve zjednodušené ukázce kódu je vidět, jak je možné s Tello dronem pracovat. Oproti ostatním nástrojům, zde není dostupné GPS. Na základě akcelerometru je ale možné získat aktuální rychlost. Myšlenka využití Tello dronu byla v dopočítávaní pozice z této rychlosti. Při

[6] https://dronekit-python.readthedocs.io/
[7] https://ardupilot.org/planner/docs/mission-planner-simulation.html
[8] https://dronekit-python.readthedocs.io/en/latest/guide/copter/guided_mode.html
[9] Software Development Kit

testování se ukázalo, že Tello není dostatečně spolehlivé. Po jednom okruhu (o průměru 2 metry) často docházelo k vychýlení pozice až o metr. Díky tomu byl dron nakonec vyhodnocen jako nepoužitelný pro otestování metody. Z testování nicméně vyplynuly důležité podněty do následujícího vývoje aplikace, kterým byla integrace se simulátorem AirSim.

```
me = tello.Tello()
me.connect()
me.streamon()
me.takeoff()
while True:
    (...)
    # Get velocity and orientation.
    yaw = me.get_yaw()
    velocity = np.array([me.get_speed_y(), me.get_speed_x(), me.get_speed_z()])

    # Get (compute) location.
    location = location + velocity * delta_time
    location[2] = me.get_height()

    # Calculate save command.
    save_command_speed = corrector.adjust_command(location, velocity, command_speed, path)

    # Send Command to the drone.
    me.send_rc_control(save_command_speed[0], save_command_speed[1], save_command_speed[2],
        yaw)
    (...)
me.land()
```

AirSim

AirSim je implementován v Unreal Enginu a je možné ho buď kompilovat a mít možnost tak zasahovat do samotného prostředí, nebo použít již zkompilovanou verzi. Pro účely projektu stačila již kompilovaná verze. Před spuštěním aplikace AirSim je potřeba vyplnit soubor `settings.json` s nastavením, co je simulovaný objekt. View mod nastaví pohled ze země, z třetí osoby. Multirotor nastaví, že je simulován dron - pro simulování dronu byl použit interní fyzikální model AirSim. U nastavení kamery bylo zmenšeno rozlišení, aby se usnadnil přenos do GUI.

```
{
  "SettingsVersion": 1.2,
  "SimMode": "Multirotor",
  "ViewMode": "Manual",
  "CameraDefaults": {
    "CaptureSettings": [
        {
          "ImageType": -1,
          "Width": 256,
          "Height": 144,
          "FOV_Degrees": 90,
          "AutoExposureSpeed": 100,
          "MotionBlurAmount": 0
        }
      ]
  }
}
```

Soubor `settings.json`, kterým se nastavuje simulátor AirSim před spuštěním.

Airsim nabízí API, které používá msgpack-rpc protocol[10] přes TCP/IP pomocí knihovny rpclib[11]. Odesílání zpráv tímto protokolem implementuje Python knihovna airsim[12]. Proto po vytvoření objektu dronu touto knihovnou je velké množství příkazů velmi intuitivní. Výrazně zjednodušená verze použití by mohla vypadat následovně:

```
# Connection.
client = airsim.MultirotorClient()
client.confirmConnection()
client.enableApiControl(True)
client.armDisarm(True)

# Takeoff.
client.takeoffAsync().join()

while True:
    (...)
    # Get Orientation.
    state = client.getMultirotorState()
    q = state.kinematics_estimated.orientation
    r, p, y = to_eularian_angles(q)
    yaw = np.rad2deg(y)
    pitch = np.rad2deg(p)
    roll = np.rad2deg(r)

    # Get speed.
    vel = state.kinematics_estimated.linear_velocity
    speed = transform.ned2utm(np.array([vel.x_val, vel.y_val, vel.z_val]))

    # Get position.
    lat = state.gps_location.latitude
    lon = state.gps_location.longitude
    x, y = transform.convert_latlon_metres_yx((lat, lon))
    z = state.gps_location.altitude - client.getHomeGeoPoint().altitude
    position = np.array([x, y, z])

    # Rotate command by yaw
    command_speed = rotate_command_by_yaw(command_speed,yaw)

    # Calculate save command.
    save_command_speed = corrector.adjust_command(location, velocity, command_speed, path)

    # Send Command to the drone.
    left_right, fwd_back, up_down, yaw = command
    command_ned = transform.utm2ned([left_right, fwd_back, up_down])
    vx, vy, vz = map_vec3_to_float_list(command_ned)
    duration = delta_time * 2
    client.moveByVelocityAsync(vx, vy, vz, duration, yaw_mode={'is_rate': True, 'yaw_or_rate': yaw})
    (...)
client.landAsync(timeout_sec=5).join()
```

Při získávaní rychlosti je nezbytné vertikální rychlost vynásobit -1, protože AirSim vychází z NED (North East Down)[13] systému, kde směr dolů je kladný. Oproti DroneKit

[10] https://github.com/msgpack-rpc/msgpack-rpc
[11] http://rpclib.net/
[12] https://microsoft.github.io/AirSim/apis/
[13] https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates

nástroji, kde šlo definovat v jakém kontextu se pokyn odesílá, zde jsem musel rotovat příkaz o úhel yaw funkcí. AirSim sice nabízí možnost pracovat s yaw, ale nechovala se podle představ. Jak je vidět v algoritmu, samotné získání yaw je složitější. Důvodem je, že AirSim si orientaci uchovává v quaternionech[14]. Je tedy potřeba provést převod do klasických Eulerovských úhlů[15].

AirSim v základu veškerou komunikaci provádí asynchronně. Pokud je potřeba příkaz provést synchronně, volá se nad ním metoda **.join()** - to se použije například u vzlétnutí. AirSim umožňuje odesílat příkazy pro pohyb na určitou dobu (například leť rovně rychlostí 10 m/s po dobu dvou sekund), nebo rychlost nastavit na neomezenou dobu a novým příkazem jí přepsat. Rozhodl jsem se posílat příkaz na dobu $2\Delta t$, aby v případě zaseknutí programu dron po chvíli zastavil. Δt je délka jednoho cyklu programu.

Při odeslání rychlosti je podstatné nastavit správně yaw_mode, aby se s rychlostí odeslal i příkaz k otáčení.

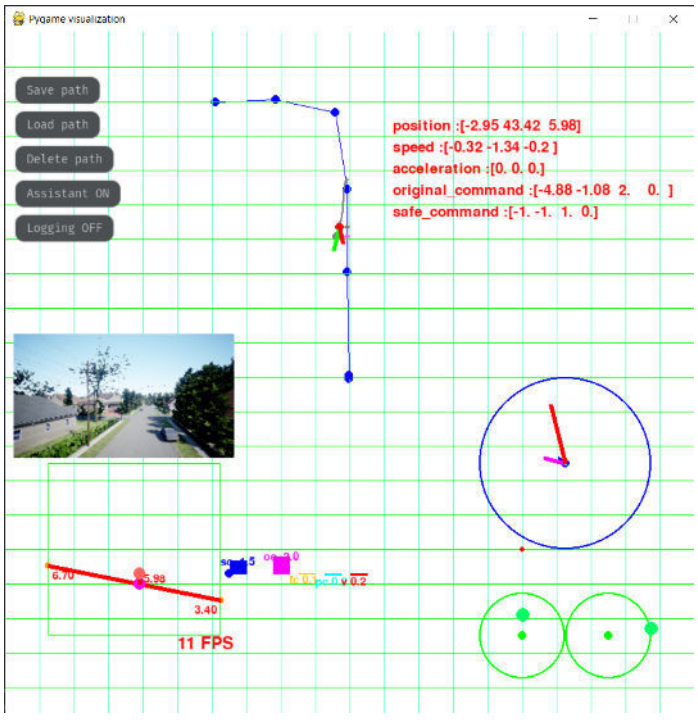
[14] https://en.wikipedia.org/wiki/Quaternion
[15] https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_$Euler$_$angles$

4.4 GUI Aplikace

Pro tvorbu GUI (viz obrázek 4.3) jsem vybral knihovnu Pygame. Alternativními možnostmi byly knihovny Tkinter a PyQt. První zmiňovaná ale nenabízí dostatek možností a druhý je naopak zbytečně komplexní na experimentální aplikaci. Hlavní vlastností, kterou PyGame nabízí je cyklicky kreslený canvas. Zde je možné implementovat veškeré vizualizační prvky popisované v návrhu. Nevýhodou takového vykreslování je efektivita a mimo jiné se vše kreslí přes CPU. Musí se tedy počítat s tím, že FPS mohou klesnout pod 60. To se ale v důležitých částech, například odeslání pokynu, řeší využitím Δt času mezi snímky.

V případě AirSim je možné získat aktuální obraz z kamery. Za normálních okolností by se pracovalo v OpenCV (Arsim se používá především pro computer vizion), ale v tomto případě se musí vykreslit video přímo do GUI. Přes dekódovávaní obrázku, prohození os, přeformátovaní na vykreslitelný objekt je možné video zobrazit. Nicméně důsledkem toho FPS klesne na 10 (to stále dovoluje řízení dronu). Funkce zobrazující video je implementována následovně:

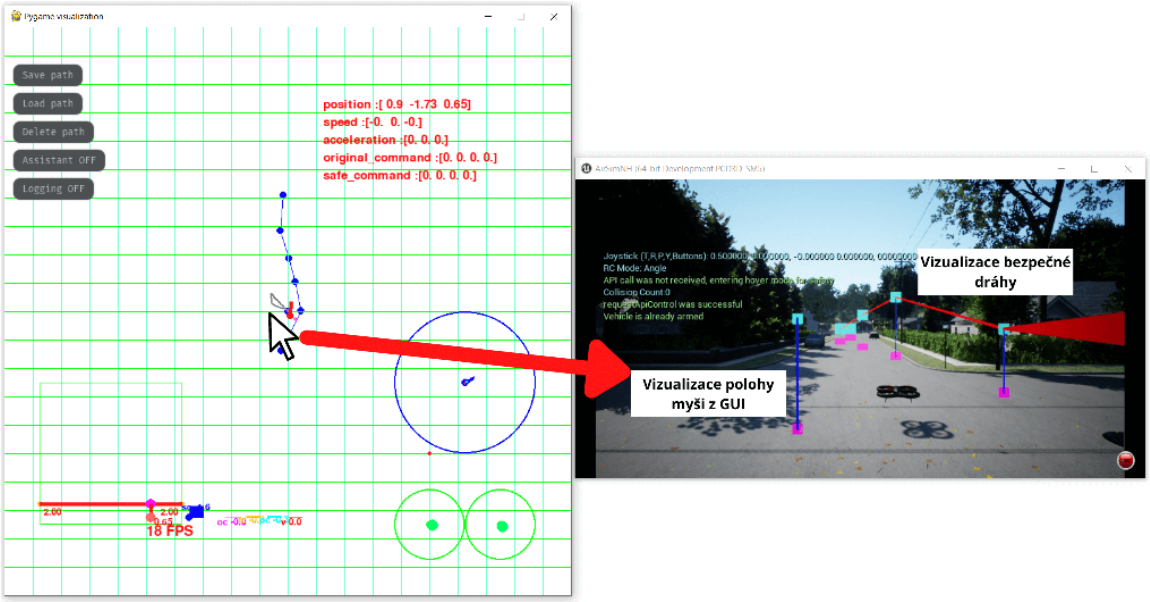
```
1 def display_video(self):
2     result = self.client.simGetImage("0", airsim.ImageType.Scene)
3     rawImage = np.fromstring(result, np.int8)
4     frame = cv2.imdecode(rawImage, cv2.IMREAD_UNCHANGED)
5     frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
6     frame = frame.swapaxes(0, 1)
7     frame = pygame.surfarray.make_surface(frame)
8     self.surface.blit(frame, (10, 350))
```



Obrázek 4.3: Samostatné okno GUI experimentální aplikace pro testování a vývoj - mapa, vytvořená dráha, dron, historie polohy, vizualizace měřítka (jeden čtverec je 10 metrů), vizualizace korekčních vektorů.

Pro práci s prostředím API AirSim nabízí velkou škálu funkcí. Hlavními možnostmi jak jde virtuální svět změnit je přidávání objektů (bude možné s nimi mít kolizi), nebo vykreslování bodů a čar. Toho jsem využil k vizualizaci bezpečné dráhy přímo do prostředí virtuálního světa. Díky tomu je možné lépe pozorovat chování dronu vůči viditelné dráze. Při experimentu ale samozřejmě byla tato funkcionalita vypnuta.

V řešení jsem implementoval možnost vykreslovat dráhu a polohu myši podle toho, kde se ve 2D GUI nachází (viz obrázek 4.4). Myš je v AirSim zobrazena společně s výškou, ve které by byl umístěn další kontrolní bod dráhy pokud by uživatel klikl s přidržením klávesy `shift`.



Obrázek 4.4: Do simulátoru AirSim je kromě bezpečné dráhy vizualizovaná také poloha myši ve světě podle toho, kde se myš nachází v GUI.

4.5 Experiment

Představená metoda bude ověřena v uživatelském testování testovacími piloty. V této kapitole se budu věnovat cíli a popisu experimentu, jeho provedení, úlohám, které při testu piloti budou plnit, a měřeným atributům pro určení výsledku testu.

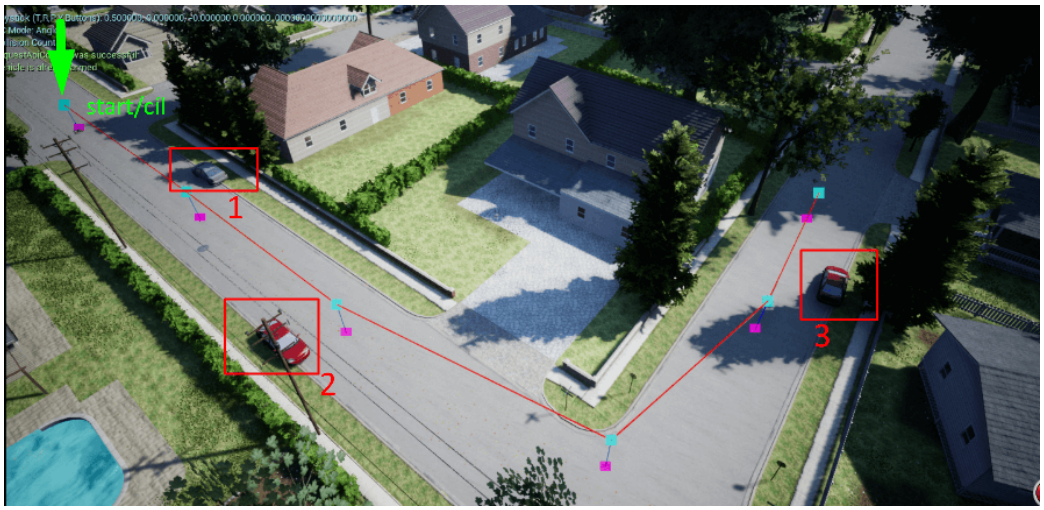
Cílem testování je ověřit splnění uživatelských požadavků na navrhovanou metodu. Motivací bylo snížit mentální zátěž pilota a zvýšit bezpečnost při vykonávání mise. Testováním je potřeba zjistit, jak se liší plnění mise při použití metody oproti běžnému standardnímu ovládání, a dále správně vyhodnotit jak subjektivní pocity uživatelů, tak míru nebezpečí (objektivní metrika).

Pilota je potřebné prověřit v situaci velmi podobné případům užití popsaných v kapitole 3.1. Je proto nezbytné navrhnout cíl mise - činnost vyžadující vyšší soustředění, než jen samotný let. Zároveň se musí vyhodnotit samotná kvalita splnění mise.

Testování si klade za cíl odpovědět na následující otázky ohledně využití asistenta pro průzkumnou misi:

1. Změní se rychlost splnění mise a jak?
2. Byl let více bezpečný?
3. Bylo plnění úloh v misi snadnější?
4. Jaké jsou situace, kdy metoda pomáhá nejvíce?
5. Jaké jsou situace, kdy metoda selhává?

Experiment je nutné navrhnout tak, aby napodoboval reálnou misi. Uživatel poletí misi (viz obrázek 4.5), při které je cílem vyfotografovat auta parkující na ulici, a to vždy ze tří směrů: šikmo zepředu, z boku a šikmo zezadu, jak to je zobrazeno na obrázku 4.7. Uživatel ví, že bezpečná dráha je uprostřed silnice a měl by se jí vždy držet (při letu s asistentem i bez). Po nafocení všech tří aut se tester vrací po bezpečné dráze na místo startu.

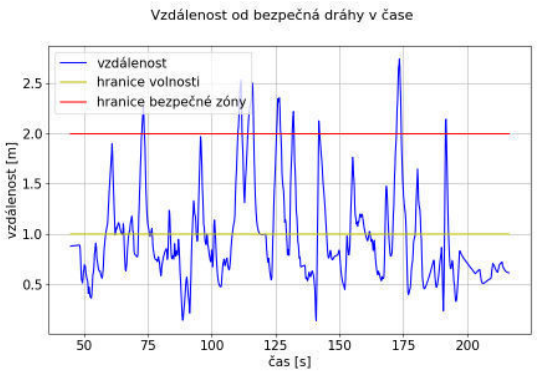


Obrázek 4.5: Pohled shora na testovací misi se zapnutou vizualizací dráhy a zvýrazněnými cílovými objekty (auty).

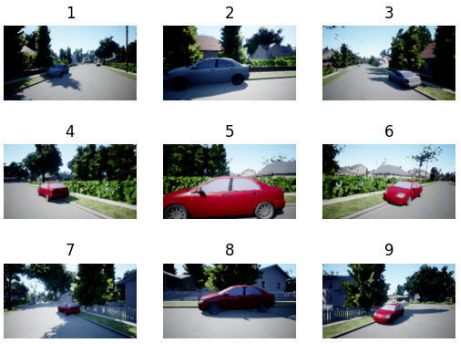
Let v uživatelském testu bude vyhodnocován na základě čtyřech metrik:

1. **Čas celého jednoho letu** se měří od chvíle, kdy je uživatel s dronem připraven k letu a stiskne tlačítko logování do chvíle, kdy se vrátí na místo startu a logování vypne. Tato metrika určí zda použití asistenta urychlí plnění mise, nebo zda ho zpomalí.

2. **Průměrná vzdálenost od ideální trajektorie** se počítá jako vzdálenost mezi polohou dronu a nejbližším bodem na trajektorii. Průběh této vzdálenosti z jednoho testovacího letu je zobrazen v grafu 4.6.



Obrázek 4.6: Graf vzdálenosti dronu od ideální trajektorie v průběhu testovacího letu s asistentem.



Obrázek 4.7: 9 fotografií aut potřebných ke splnění mise. Fotografie číslo 5 bude penalizována, jelikož není zachyceno celé vozidlo.

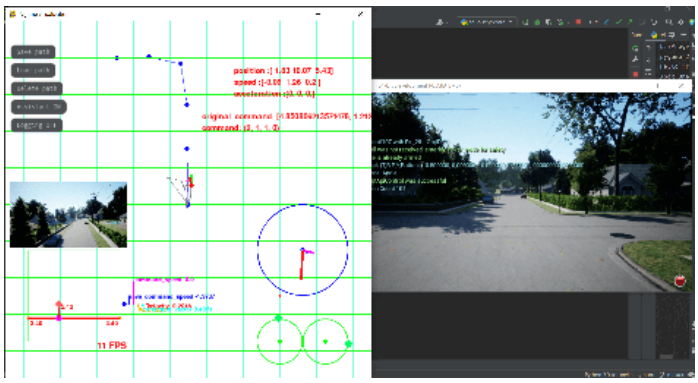
3. **Poměr času stráveného mimo bezpečnou zónu (v procentech)** určí jak často se uživatel vyskytoval mimo bezpečnou zónu. V grafu 4.6 jsou znázorněny dvě důležité hranice. První je "hranice volnosti" určující volnou oblast, ve které ještě není aplikován asistent (jeho aktuální korekční síla je nulová), druhou je "hranice bezpečné zóny". Pokud se za ní dron dostane, vyskytne se v nebezpečné zóně, což se podepíše ve výsledku testu.

4. **Počet chybných fotografií** určí kvalitu vykonání úkolu mise. Jelikož úkolem mise je fotografování tří aut, tak kvalitní fotografie je taková, kde se auto nachází celé a část není oříznutá. Zároveň musí být auto vyfoceno z dostatečné blízkosti a ze všech tří úhlů. Chybějící fotografie se počítá jako chybná. Ukázka fotografií, kde prostřední fotografie (5) je chybná je zobrazena na obrázku 4.7.

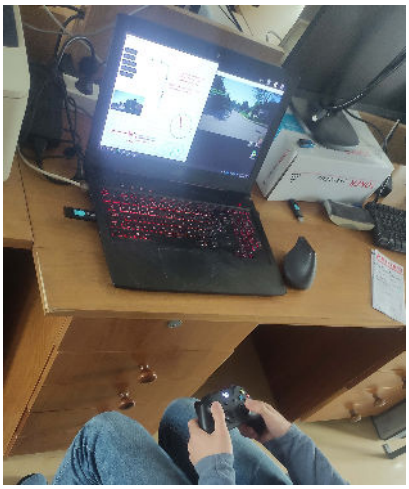
Subjektivní metrikou je míra snížení zátěže, které bude vyhodnocená pozorováním a dotazníkem. Otázky dotazníku jsou následující:

1. Připadal jsi si bezpečněji se zapnutým asistentem?
2. V jakém aspektu řízení asistent nejvíce pomohl?
3. V jakých situacích asistent selhal, nebo byl kontraproduktivní?
4. Zvolil by sis plnit misi raději s asistentem, nebo bez?
5. Co by bylo třeba zlepšit na korekční metodě asistenta?

Realizace experimentu musí umožnit testovacímu pilotovi vžít se do letu s dronem, jako kdyby byl dron opravdový. Pro testování bude použit simulátor AirSim, jenž nabízí ideální letové prostředí. Hlavní kamera simulátoru je nastavena na režim pohledu třetí osoby. Je umístěna na zemi tak, aby navozovala dojem reálného řízení dronu. Pokud se dron vzdálí od kamery, uživatel ho stěží vidí a pilot se musí spolehnout na představené GUI, ve kterém vidí obraz z kamery dronu. Kamera dronu nedisponuje gimbalem. Pohled testovacího uživatele je zobrazen na obrázku 4.8 a prostředí testování je zachyceno na fotce 4.9.



Obrázek 4.8: Experimentální GUI pro uživatelské testy v simulátoru AirSim.



Obrázek 4.9: Fotografie prostředí, ve kterém se provádělo testování.

Průběh experimentu s každým uživatelem musí začít představením experimentu a zaučením uživatele. Uživateli bude nejdříve představeno ovládání - seznámí se s chováním dronu krátkým volným letem. Poté si nanečisto proletí dráhu a vyzkouší si fotografování. Vyzkouší si také, jaké je, když na něj při letu působí asistent.

Před každým testovacím letem si uživatel podle pokynů připraví veškerá nastavení a s dronem vzletí. Když je dron ve vzduchu, spouští logování a zahajuje test. Po návratu test ukončuje vypnutím logování a resetuje program pro další let. Takto uživatel letí 4 lety nejdříve bez asistenta a poté 4 lety s asistentem.

4.6 Vyhodnocení

V této kapitole charakterizuji skupinu testovacích pilotů, popíši, jak testy probíhaly a prezentuji agregované výsledky. Na konci kapitoly provedu diskuzi nad výsledky a navrhnu možná zlepšení.

Výsledky testování byly získané nad těmito konkrétními testery (viz tabulka 4.1):

Charakteristika	Pohlaví	Věk	Zkušenosti s řízením dronů
Autor článku, student vš	muž	23	Menší zkušenost s létáním s drony
Student gymázia	muž	15	Žádné zkušenosti s ovládáním dronu, ale rychlé porozumění principu řízení
Student gymázia	muž	15	
Student gymázia	muž	17	

Tabulka 4.1: Tabulka charakterizující testovací uživatele.

Pro správné vyhodnocení testů je potřeba výsledky zpracovat a vhodně interpretovat. Výstupem každého nahraného letu je komplexní log obsahující základní letové informace, informace související přímo s bezpečnou dráhou a metodou samotnou. Nad těmito informacemi byla dále provedena sumarizace, kde se získaly potřebné statistiky (minima, maxima, průměry). Všechny tyto výstupy byly roztříděny a pro každý let byla provedena manuální kontrola kvality fotografií, při níž byla sumarizace doplněna o počat chybných fotek. Nakonec se lety zpracovaly pomocí vyhodnocovacího skriptu, jenž seskupil výsledky všech testerů.

	s asistentem	bez asistenta
Vzdálenost	0,92 m	2,29 m
Trvání nebezpečí	5,16 %	55,16 %
Kvalita fotografií	1,18 chyb	0,18 chyb
Čas letu	153,43 s	175,50 s

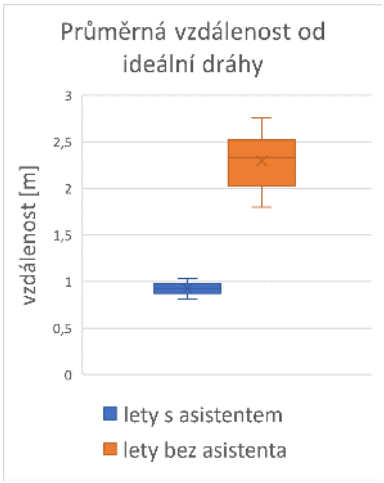
Tabulka 4.2: Tabulka průměrných hodnot měřených pro všechny lety.

Tabulka 4.2 zobrazuje **výsledky testování** s použitím asistenta oproti letů bez něj. Jedná se o průměrné výsledky všech testovacích pilotů, tudíž každá hodnota odpovídá průměru z 16 letů.

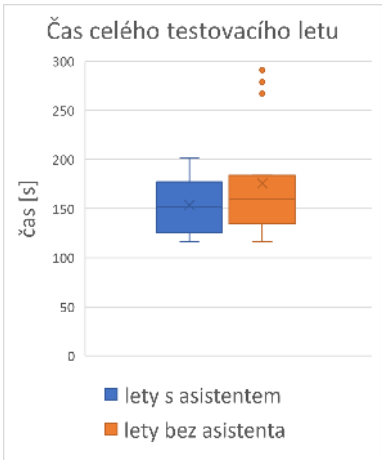
Výsledky testování ukázaly, že s použitím asistenta značně klesá míra nebezpečí a to z 55 % na 5 %. Průměrná velikost vychýlení od dráhy je pro let s asistentem o 1,37 metrů nižší.

Výsledky jsou dále zobrazené pomocí krabicových grafů 4.10 a 4.11. Z těch je možné vyčíst průměr, medián, minimum a maximum, horní a dolní kvantil a odchýlené hodnoty.

Časy testovacího letu se víceméně neliší (viz graf 4.10b), protože v obou letech uživatelé museli letět striktně po silnici a největší zdržení vždy způsobilo pořizování fotografie. I přesto se jedná o uspokojující závěr. Pokud by byl například čas bez asistenta mnohem rychlejší znamenalo by to, že uživatel letěl tyto mise nezodpovědně. Dá se předpokládat, že míra nebezpečí by se vyšplhala k závratným hodnotám a let by bylo nutné zneplatnit. Současně z výsledků vyplývá, že zde nebyl žádný pilot, jehož schopnosti by byly pro test nepřípustné, protože všichni piloti letěli podobně rychle.



(a) Vzdálenost

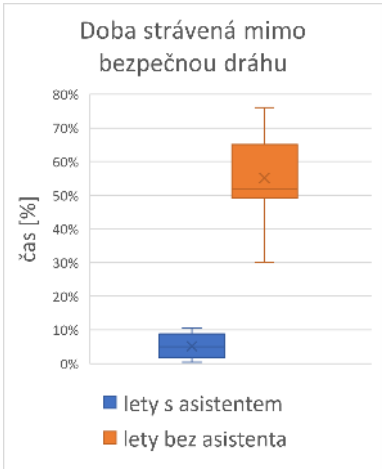


(b) Čas letu

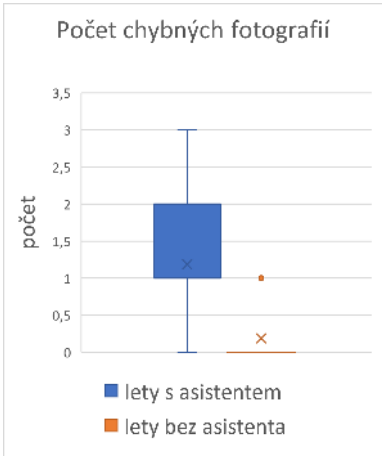
Obrázek 4.10: Krabicové grafy výstupních hodnot uživatelských testů.

Průměrná vzdálenost od bezpečné dráhy (viz graf 4.10a) byla při použití asistenta snížena z 0,92 na 2,29 metrů. Při letech bez asistenta bylo možné pozorovat, jak při pořizování fotek piloti zapomínají udržovat bezpečné území. Když pilot fotil auto šikmo zepředu, často se stalo, že zaletěl moc ke kraji silnice. Při použití asistenta mu v tomto pohybu bylo zabráněno a pilot si uvědomil, že musí dron pro pořízení fotky umístit jinak.

Doba strávená mimo bezpečnou dráhu (viz graf 4.11a) vypovídá, že piloti při letu bez asistentem trávili velkou část letu mimo bezpečnou zónu. U letu s asistentem se to stávalo jen zřídka a to převážně jen když pilot cíleně letěl na maximální výkon na okraj bezpečné dráhy kvůli pořízení fotografie.



(a) Trvání nebezpečí



(b) Kvalita fotografií

Obrázek 4.11: Krabicové grafy výstupních hodnot uživatelských testů.

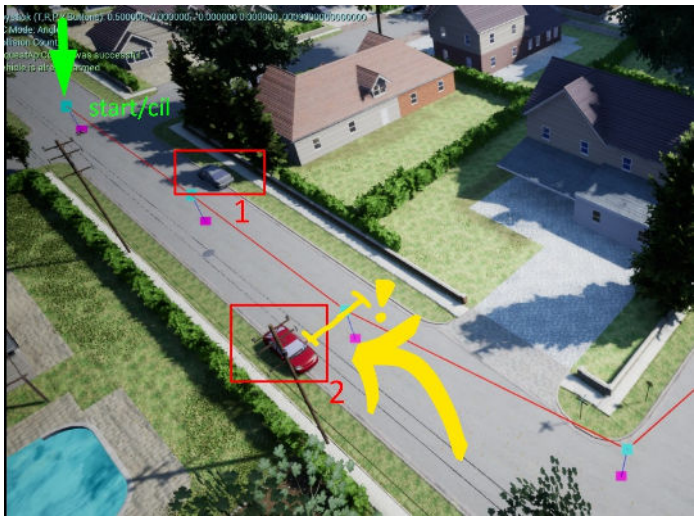
Počet chybných fotografií (viz graf 4.11b) je významně odlišný. To je způsobeno komplikací při fotografování dronem z hranice bezpečné zóny. Při testování uživatelům nejvíce dělalo problém pořídit kvalitní fotografie, pokud byli nuceni dostat se při letu s asistentem

na kraj bezpečné zóny a tam "zápasit" s asistentem. Proto let s asistentem obsahuje v průměru jednu a více chybných fotografií. U letu bez pilota si uživatel dron umístil vždy do ideální pozice (na úkor bezpečnosti), tudíž se chybná fotografie vyskytovala jen ojediněle.

Na otázky v dotazníku odpověděli tři účastníci testu (autor článku neodpovídal).

- Q1: Připadal jsi si bezpečněji se zapnutým asistentem?
- A1: „ Ano, jednoznačně.“, „ Ano, i když mi trochu bral svobodu.“, „Ano.“
- Q2: V jakém aspektu řízení asistent nejvíce pomohl?
- A2: „Udržení se ve středu cesty.“, „ Udržení správné výšky.“, „Udržení se uprostřed cesty.“
- Q3: V jakých situacích asistent selhal, nebo byl kontraproduktivní?
- A3: „Při přistávání.“, „ Trochu překážel při focení aut.“, „ Při focení aut mírně překážel.“
- Q4: Zvolil by sis plnit misi raději s asistentem, nebo bez.
- A4: „S asistentem.“, „S asistentem.“, „S asistentem.“

Z výsledků lze vyvodit, že navržené řešení pomohlo uživatelům provést misi stejně rychle a o 50 % bezpečněji, ale částečného omezení volnosti pohybu. Pokud je při pořizování fotografií nevhodně zadefinovaná dráha (viz obrázek 4.12), asistent zhoršuje schopnost pořídit fotografii z vhodného úhlu. Testovací uživatelé potvrdili, že by si zvolili pro let mise zapnutého asistenta. Mezi nedostatky uvádějí především „překáží při focení“, což odpovídá i pozorování. Jako nejužitečnější pomoc od asistenta uvádějí udržení výšky a udržení na středu cesty.



Obrázek 4.12: Situace, kdy pilot, aby dostal do záběru celé červené auto, musel zápasit s korekčním algoritmem, jelikož se nacházel na okraji bezpečné zóny.

Diskuze nad výsledky testování.

Při samotném testování bylo vidět, že jedním z problémů je „zápasení“ s asistentem v situaci (viz obrázek 4.12), kdy se dron nedokáže dostat do vhodné polohy pro zařízení fotografie. V takové situaci se testovací uživatelé snažili vyfotit fotografii spěšně ve vhodný okamžik. To pak vedlo k horším výsledkům viz 4.11b.

Na tento problém jsem navrhl koncept řešení s pracovním názvem *focus mode*. Ve chvíli, kdy uživatel potřebuje vyfotit fotografii a ví, že by se musel dostat do přetlačovaní s asistentem, aktivuje si stisknutím přední páčky na ovladači *focus mode*. V tu chvíli se sníží citlivost ovládacích pokynů, maximální rychlost dronu se omezí a hranice, kde začíná působit asistent se posune blíže nebezpečné zóně. Uživatel si opatrně, ale pohodlně vytvoří fotografii a poté vypuštěním páčky uvede vše do normálu.

Kapitola 5

Závěr

Cílem práce bylo navrhnout metodu, která sníží mentální zátěž pilota při plnění pozorovací mise. Navržené řešení staví na znalostech v oblasti řízení dronů, existujících aplikacích pro pilota a studiích o mentální zátěži pilotů. Nově navržená metoda je založena na předem definovaném bezpečném území (trajektorii).

Navrhl jsem řešení zaměřující se na řízení dronu s korekcí povelů (let s asistentem). Řešení staví na principu filtrování pokynu pilota ještě před jejich odesláním dronu. Jádrem je korekční metoda, jež na základě stavu systému (poloha a rychlost dronu, bezpečná dráha a pokyn pilota) provede korekci pokynu. Korekci určují dvě korekční síly (lokální a prediktivní) odvozené od vektorového pole vzdálenostní funkce okolo bezpečné trajektorie.

Zrealizoval jsem experimentální řešení v jazyce Python využívající knihovnu PyGame pro získaní řídících vstupů od pilota. Aplikace komunikuje se simulátorem AirSim poskytující fyzikální model dronu a virtuální prostředí pro testovací lety. Cílem experimentální aplikace bylo umožnit otestovat navrhovanou metodu na uživatelích. Experimenty jsem zaměřil na plnění pozorovací mise s úkolem fotografovat cílové objekty. Provedl jsem na testování na čtyřech uživatelích, kde každý letěl 4 lety bez asistenta a 4 lety s asistentem. Test jsem vyhodnotil pomocí měření, pozorování a dotazníku. Zjistil jsem, že se uživatelé cítí při letu s asistentem bezpečněji a jako nejužitečnější pomoc od asistenta uvádějí „udržení výšky a udržení středu cesty“. Podařilo se mi demonstrovat přínos navrženého postupu ve zvýšení bezpečnosti letu. Průměrný uživatel bez asistenta se vyskytoval v nebezpečné zóně 55 % trvání mise. Oproti tomu uživatel se zapnutým asistentem pouze 5 %. Odchylka od bezpečné dráhy byla při letu bez asistenta 2,29 m při bezpečné šířce 2 metry. Uživatel se zapnutým asistentem dosahoval průměrné odchylky 0,92 m za stejných podmínek. To je o 1,37 metrů menší odchylka. Současně experimenty odhalily, že pokud pro pořízení fotky uživatel musel umístit dron na okraj bezpečné zóny, bylo působení korekční metody nepříjemné. Užitečnost metody tedy velmi závisí na správném definování bezpečné dráhy. Na základě testů byla navrhnuta možnost “focus”, díky níž bude moct uživatel tlačítkem aktivovat zpomalený mód letu a lépe manévrovat na okraji bezpečné zóny při focení.

Největším přínosem práce je princip řešení poskytující uživateli volnost pohybu a přitom mu hlídá bezpečnost. Vytvořený program je modulární a dále může sloužit jako framework, ve kterém bude nadále možnost vyvíjet a testovat inovativní přístupy k řízení dronů. Do budoucna by bylo vhodné provést experimenty s reálným dronem v reálném prostředí, najít nestabilní situace navržené metody a zlepšit v nich stabilitu. Bylo by možné rozšířit řešení o další řídící prvky definující bezpečný prostor letu. Dosáhlo by se tak větší komplexnosti bezpečného území, jež je pro řešení klíčové. Zároveň se dá uvažovat o návrhu různých typů omezení odpovídajících povaze místa letu.

Literatura

[1] Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R. et al. *A survey on Deep Learning Advances on Different 3D Data Representations*. arXiv, 2018. DOI: 10.48550/ARXIV.1808.01462. Dostupné z: https://arxiv.org/abs/1808.01462.

[2] Arturo Urquizo. *Atomic force microscopy* [online]. 2013 [cit. 1.5.2022]. Dostupné z: https://en.wikipedia.org/wiki/PID_controller#/media/File:PID_en.svg.

[3] Bloomenthal, J., Bajaj, C., Blinn, J., Cani, M.-P., Wyvill, B. et al. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.

[4] Cedro, L. a Wieczorkowski, K. Optimizing PID controller gains to model the performance of a quadcopter. *Transportation Research Procedia*. 2019, sv. 40, s. 156–169. DOI: https://doi.org/10.1016/j.trpro.2019.07.026. ISSN 2352-1465. TRANSCOM 2019 13th International Scientific Conference on Sustainable, Modern and Safe Transport. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2352146519301875.

[5] De la Torre, G. G., Ramallo, M. A. a Cervantes, E. Workload perception in drone flight training simulators. *Computers in Human Behavior*. 2016, sv. 64, s. 449–454. DOI: https://doi.org/10.1016/j.chb.2016.07.040. ISSN 0747-5632. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0747563216305301.

[6] Funkhouser, T. Overview of 3d object representations. *Princeton University, C0S D*. 2003, sv. 597.

[7] Hancock, P. A. A dynamic model of stress and sustained attention. *Human factors*. SAGE Publications Sage CA: Los Angeles, CA. 1989, sv. 31, č. 5, s. 519–537.

[8] Hart, S. G. a Staveland, L. E. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: Hancock, P. A. a Meshkati, N., ed. *Human Mental Workload*. North-Holland, 1988, sv. 52, s. 139–183. Advances in Psychology. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. ISSN 0166-4115. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0166411508623869.

[9] Henderson, T. C. Efficient 3-D Object Representations for Industrial Vision Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1983, PAMI-5, č. 6, s. 609–618. DOI: 10.1109/TPAMI.1983.4767450.

[10] Hubinák, R. *Aplikace pro efektivní řízení dronu s využitím rozšířené virtuality.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav počítačové grafiky a multimédií. Dostupné z: http://hdl.handle.net/11012/194940.

[11] Ikeda, T., Bando, N. a Yamada, H. Semi-Automatic Visual Support System with Drone for Teleoperated Construction Robot. *Journal of Robotics and Mechatronics.* 2021, sv. 33, č. 2, s. 313–321. DOI: 10.20965/jrm.2021.p0313.

[12] Mouloua, M., Gilson, R., Kring, J. a Hancock, P. Workload, Situation Awareness, and Teaming Issues for UAV/UCAV Operations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting.* 2001, sv. 45, č. 2, s. 162–165. DOI: 10.1177/154193120104500235. Dostupné z: https://doi.org/10.1177/154193120104500235.

[13] Nguyen, D. D., Rohacs, J. a Rohacs, D. Autonomous Flight Trajectory Control System for Drones in Smart City Traffic Management. *ISPRS International Journal of Geo-Information.* 2021, sv. 10, č. 5. DOI: 10.3390/ijgi10050338. ISSN 2220-9964. Dostupné z: https://www.mdpi.com/2220-9964/10/5/338.

[14] Rostáš, J., Kováčiková, M. a Kandera, B. Use of a simulator for practical training of pilots of unmanned aerial vehicles in the Slovak Republic. In: IEEE. *2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA).* 2021, s. 313–319.

[15] Sedlmajer, K. *Uživatelské rozhraní pro řízení dronu s využitím rozšířené virtuality.* Brno, CZ, 2019. Master's thesis. Brno University of Technology, Faculty of Information Technology. Dostupné z: https://www.fit.vut.cz/study/thesis/16730/.

[16] Sedlmajer, K., Bambušek, D. a Beran, V. Effective Remote Drone Control Using Augmented Virtuality. In: *Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications 2019.* SciTePress - Science and Technology Publications, 2019, s. 177–182. DOI: 10.5220/0008349401770182. ISBN 978-989-758-376-6. Dostupné z: https://www.fit.vut.cz/research/publication/12006.

[17] Shah, S., Dey, D., Lovett, C. a Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Springer. *Field and service robotics.* 2018, s. 621–635.

[18] Trenev, I., Tkachenko, A. a Kustov, A. "Movement stabilization of the parrot mambo quadcopter along a given trajectory based on PID controllers". *IFAC-PapersOnLine.* 2021, sv. 54, č. 13, s. 227–232. DOI: https://doi.org/10.1016/j.ifacol.2021.10.450. ISSN 2405-8963. 20th IFAC Conference on Technology, Culture, and International Stability TECIS 2021. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2405896321018875.

[19] Turk, G. a O'brien, J. F. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG).* ACM New York, NY, USA. 2002, sv. 21, č. 4, s. 855–873.

[20] Welch, G., Bishop, G. et al. An introduction to the Kalman filter. Chapel Hill, NC, USA. 1995.

[21] Yan, F., Liu, Y.-S. a Xiao, J.-Z. Path planning in complex 3D environments using a probabilistic roadmap method. *International Journal of Automation and computing*. Springer. 2013, sv. 10, č. 6, s. 525–533.

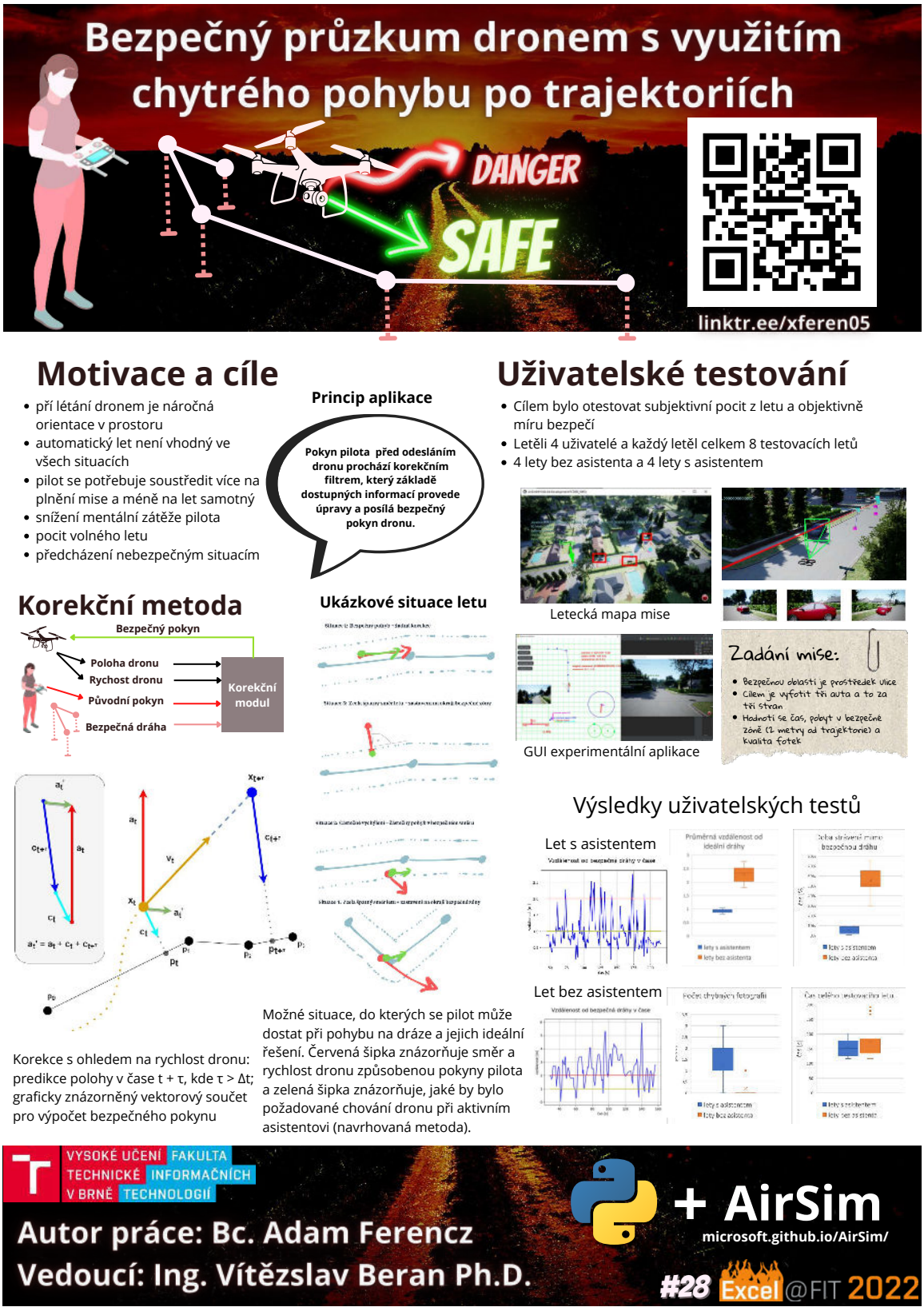
Příloha A

Obsah paměťového media

V rámci práce byl vytvořený plakát o velikosti A1. Náhled plakátu se nachází na další straně. Plakát se též nachází v přiloženém paměťovém mediu.

Paměťové medium má následující strukturu:

- `poster.pdf` – PDF soubor s plakátem (viz obrázek A.1),
- `video.mp4` – video prezentující práci,
- `text.pdf` a `text_print.pdf` – text technické zprávy,
- adresář `/tex` – zdrojové texty a soubory tecnické zprávy v jazyku LaTeX,
- adresář `/src` – zdrojové kódy k práci, nachází se zde soubor `README.md`, které blíže specifikuje strukturu souborů a obsahuje návod na spuštění programu,
- adresář `/src/log` se nachází kompletní výstupy provedeného uživatelského testování.



Obrázek A.1: Vytvořený plakát.