



FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

DIPLOMOVÁ PRÁCE

**Pokročilé algoritmy autonomního
přistávání bezpilotního letounu na plošině**

Autor:

Vojtěch Breník

Vedoucí práce:

Ing. Petr Neduchal, Ph.D.

18. května 2024

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:

Bc. Vojtěch BRENÍK

Osobní číslo:

A22N0105P

Studijní program:

N0714A150011 Kybernetika a řídicí technika

Specializace:

Umělá inteligence a automatizace

Téma práce:

Pokročilé algoritmy autonomního přistávání bezpilotního letounu na plošině

Zadávající katedra:

Katedra kybernetiky

Zásady pro vypracování

1. Proveďte rešerši v oblasti algoritmů pro autonomní přistání bezpilotního letounu (dronu) na plošině.
2. Proveďte rešerši dostupných simulátorů vhodných pro tuto úlohu.
3. Analyzujte možnosti simulace externích vlivů (vítr, teplota, ...) na přistávající bezpilotní letoun.
4. Navrhněte systém pro simulaci přistání bezpilotního letounu na plošině s využitím některé z metod uvedených v rešerši.



Rozsah diplomové práce:

40-50 stránek A4

Rozsah grafických prací:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Xin, L., Tang, Z., Gai, W., & Liu, H. (2022). Vision-based autonomous landing for the uav: A review. *Aerospace*, 9(11), 634.
2. Kakaletsis, E., Symeonidis, C., Tzelepi, M., Mademlis, I., Tefas, A., Nikolaidis, N., & Pitas, I. (2021). Computer vision for autonomous UAV flight safety: An overview and a vision-based safe landing pipeline example. *Acm Computing Surveys (Csur)*, 54(9), 1-37.
3. Saavedra-Ruiz, M., Pinto-Vargas, A., & Romero-Cano, V. (2022). Monocular Visual Autonomous Landing System for Quadcopter Drones Using Software in the Loop. *IEEE Aerospace and Electronic Systems Magazine*, 37(5), 2-16.

Vedoucí diplomové práce:

Ing. Petr Neduchal, Ph.D.

Výzkumný program 1

Datum zadání diplomové práce: **2. října 2023**

Termín odevzdání diplomové práce: **20. května 2024**


Doc. Ing. Miloš Železný, Ph.D.
děkan




Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 20. května 2024

ZÁPADOČESKÁ UNIVERZITA

Fakulta aplikovaných věd

Katedra kybernetiky

Abstrakt

Pokročilé algoritmy autonomního přistávání bezpilotního letounu na plošině

Vojtěch Breník

Český abstrakt

**Advanced algorithms for autonomous landing of an unmanned aerial vehicle
on a platform**

English abstract

Poděkování

Poděkování . . .

Obsah

Abstrakt	ii
1 Úvod	1
1.1 Bezpilotní letadlo	1
1.1.1 Komponenty malého vícerotorového letadla	1
1.2 Přistávání	2
1.3 Struktura práce	3
2 Definice úlohy	4
3 Plošina pro přistávání	6
3.1 Fiduciární markery	6
3.1.1 ARTag	7
3.1.2 Aruco	7
3.1.3 Apriltag	7
4 Detekce fiduciárních markerů	8
5 Metody přistávání	10
5.1 Obecné vlastnosti implementovaných metod	12
5.2 Přistávání po svislé přímce	13
5.3 Přistávání po svislé přímce s využitím Kálmánova filtru	14
5.4 Přistávání po skloněné přímce	16
5.5 Přistávání po skloněné přímce s využitím Kálmánova filtru	16
6 Simulátory bezpilotních letadel	18
6.1 Gazebo	19
6.2 RealFlight	20
6.3 jMAVSIM	20
6.4 FlightGear	20
6.5 JSBSim	20
6.6 Airsim	21
7 Návrh systému pro simulaci přistávání UAV na plošině	22
7.1 Komponenty systému	22
7.1.1 Letový řídicí software	22
7.1.2 Simulátor	22
7.1.3 Grafické uživatelské rozhraní (GUI)	23
7.1.4 Hodiny ze simulátoru	23
7.1.5 Kamera ze simulátoru	23
7.1.6 Detektor fiduciárních markerů	23
7.1.7 Metoda přistávání	24

7.1.8	Mise a jejich seznamy	24
7.1.9	Model světa	24
7.2	Vstupy systému	24
7.2.1	Počáteční poloha UAV a plošiny	24
7.2.2	Nastavení zastínění plošiny	25
7.2.3	Nastavení větru	25
7.2.4	Výběr přistávací metody	25
7.2.5	Uložené mise a jejich seznamy	25
7.2.6	Konfigurační soubor	25
7.3	Výstupy systému	25
7.3.1	3D vizualizace simulovaného prostředí	25
7.3.2	Pohled kamery s vyznačenými detekovanými markery	26
7.3.3	Stav mise a experimentu	26
7.3.4	Výsledek mise a experimentu	26
7.3.5	Uložené mise a jejich seznamy	26
7.4	Struktura systému	26
8	Grafické uživatelské rozhraní	29
8.1	Obrazovka Mise	29
8.2	Obrazovka Živě	31
8.3	Obrazovka Experimenty	33
9	Experimenty a vyhodnocení	35
9.1	Přesnost přistání	35
9.2	Přesnost přistávání	38
9.3	Úspěšnost přistávání	38
9.4	Doba trvání přistávání	40
9.5	Střední doba výpočtu v jednom kroku algoritmu	40
9.6	Vliv stínu na podlíd nedetekovaných markerů	41
10	Diskuze	42
11	Závěr	44
	Bibliografie	45

Seznam obrázků

1.1	Uspořádání rotorů rámů u čtyřrotorových letadel	2
1.2	Schéma zapojení součástí letadla	3
3.1	Příklady různých fiduciárních markerů	7
4.1	Postup při detekci AprilTagů	9
5.1	Činnosti obecného přistávacího algoritmu	12
5.2	Přistávání po skloněné přímce	16
7.1	Struktura navrhovaného systému	28
8.1	GUI: Obrazovka „Mise“	30
8.2	Přistávání na zastíněnou plošinu	31
8.3	GUI: Obrazovka „Živě“	32
8.4	GUI: Obrazovka „Experimenty“	34
9.1	Chyba před dosednutím	36
9.2	Chyba přistání	37
9.3	Příklad porovnání přistání	38
9.4	Doba trvání přistávání	40

Seznam tabulek

6.1	Simulátory podporované kontrolérem letounu	18
6.2	Vlastnosti vybraných simulátorů	19
9.1	Třídy větrných podmínek	35
9.2	Střední absolutní chyba přistávání	39
9.3	Úspěšnost přistávání	39
9.4	Průměrná střední doba jednoho kroku algoritmu	41

Seznam zkratek

API Application Programming Interface - rozhraní pro programování aplikací. 21, 23, 27

ESC Electronic Speed Controller - elektronický kontrolér rychlosti. 2

GPS Global Positioning System - globální polohový systém. 2–4, 12, 39

GUI Graphical User Interface - grafické uživatelské rozhraní. iv, 3, 23–27, 29, 31, 33

HIL Hardware in the loop. 21

IMU Inertial Measurement Unit - inerciální měřicí jednotka. 2, 4, 11, 12, 14

LSTM Long Short-Term Memory - druh rekurentní neuronové sítě. 11

MAE Mean Absolute Error - střední absolutní chyba. 26, 33, 38

MSE Mean Squared Error - střední kvadratická chyba. 9

RTK Real-time Kinematic. 3

SIH Simulation in Hardware - simulace na hardwaru. 22

SW Software. 4, 24, 27, 42

UAV Unmanned Aerial Vehicle - bezpilotní letadlo. iv, v, 1–6, 10, 11, 13, 14, 16, 19–29, 31, 35, 38, 40, 42, 44

UDP User Datagram Protocol. 20

1 Úvod

Bezpilotní letadla se používají v mnoha oblastech lidské činnosti mimo jiné díky nízkým nákladům na pořízení i provoz a vysoké manévrovatelnosti [9]. Široké uplatnění nachází v oblastech jako je průzkum při mimořádných událostech a s nimi související práce, monitorování prostředí [35], dopravní a bezpečnostní dohled, služba pátrání a záchrany, doprava zboží, zemědělství nebo při výstavbě a kontrole staveb [25]. Součástí každé mise takového letadla je kromě samotné operace ve vzduchu také start a přistání. Řídicí systémy letadel obsahují funkce, které zajišťují jejich autonomii během startu i letu, ale kvůli složitosti manévrů přistání a nedostatečné přesnosti obvyklých senzorů nedosahují potřebné spolehlivosti [7]. Tato diplomová práce se zabývá využitím kamery a plošiny označené fiduciárním markerem pro automatizaci přistávání bezpilotního letadla se svislým přistáním v simulovaném prostředí, vnějšími vlivy, které mohou přistávání ovlivnit a návrhem systému, který umožňuje efektivní testování různých metod přistávání. Jejím cílem je zhodnotit použitelnost různých simulátorů pro řešení úlohy přistávání a simulace vnějších vlivů, navrhnout simulační systém, s jeho využitím vyzkoušet různé dostupné metody přistávání a vyhodnotit jejich vlastnosti z různých hledisek ve vztahu k simulovaným vnějším vlivům.

1.1 Bezpilotní letadlo

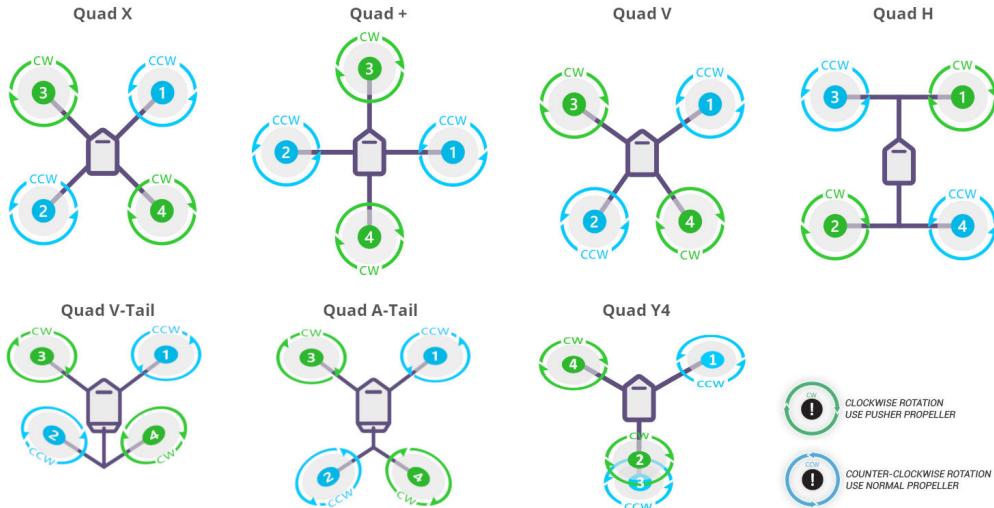
Bezpilotním letadlem se rozumí zařízení, které je schopné vyvozovat síly nesoucí ho v atmosféře, které nejsou reakcemi vůči zemskému povrchu a je způsobilé létat bez pilota, tzn. je za letu řízené automatickým zařízením nebo dálkově ze země a označuje se také jako dron nebo zkratkou UAV z anglického unmanned aerial vehicle. Obecně se může jednat o letadla lehká než vzduch (např. balony, vzducholodě) i těžší než vzduch (např. letadla s nepohyblivými nosnými plochami, rotorová letadla). [3]

Praktické využití přistávání na plošině se týká pouze letadel se svislou dráhou přistání a pro účely této práce budou uvažována pouze malá vicerotorová letadla nebo také multi-koptéry. Experimenty (kapitola 9) byly provedeny s letadlem o 4 rotorech.

1.1.1 Komponenty malého vicerotorového letadla

Vicerotorová letadla mají konstrukční, pohonné a řídicí součásti, které dohromady určují jeho letové vlastnosti jako jsou nejvyšší přípustná hmotnost nákladu, dynamické vlastnosti, dolet atp., podle nichž mohou být daná UAV vhodná jen pro určitou oblast aplikace, a dále palubní příslušenství, které se volí podle plánovaného využití. Základní schéma pohonného a řídicího součástí je na obrázku 1.2.

Hlavní konstrukční součástí je rám, ke kterému jsou připevněny ostatní komponenty. Obvykle má paprscité uspořádání v jehož středu se nachází převážně řídicí komponenty a jehož ramena na koncích nesou motory s vrtulemi. Na různých místech rámu mohou být pomocí držáků a nosičů dále připevněny přídavné senzory závislé na aplikaci (barevné kamery, IR kamery a gimbaly, senzory vzdálenosti atd.), nožičky pro přistávání, antény



OBRÁZEK 1.1: Příklady uspořádání rotorů a rámů u čtyřrotorových letadel. Zeleně značené rotory se točí v záporném směru, modré v kladném.

Převzato z [12] a upraveno.

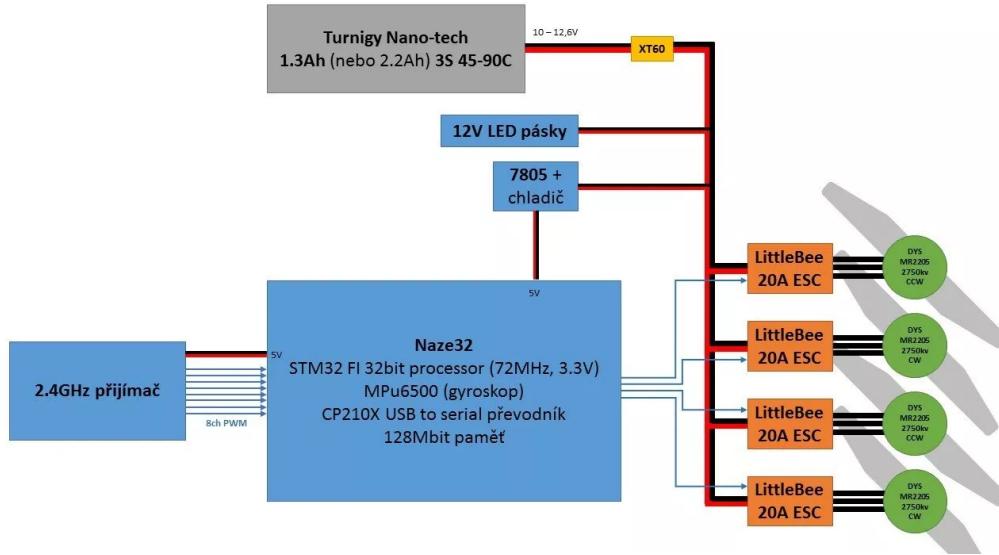
pro rádiovou komunikaci s pozemní stanicí nebo jinými letadly (např. při použití ve skupině více letadel) nebo ochranné rámy pro použití v blízkosti překážek nebo v interiéru. V závislosti na počtu rotorů a jejich uspořádání se rozlišují různé konfigurace rámu, jejich příklady jsou na obrázku 1.1.

Pohonné součásti jsou ty, které slouží k zajištění vztlaku potřebného pro let a jsou ovládány řídicími součástmi. Letadla tohoto druhu jsou téměř výhradně elektrická napájená akumulátory. Každý z rotorů je poháněn obvykle elektronicky komutovaným stejnosměrným motorem zapojeným do elektronického kontroléru rychlosti (ESC), jenž je ovládán řídicí jednotkou dronu a dodává motoru energii z akumulátoru. Směr otáčení jednotlivých rotorů a jejich počet je často volen tak, aby bylo možné kompenzovat moment působící na letadlo otáčením vrtulí, na obrázku 1.1 je toto vyobrazeno modrou a zelenou barvou rotorů. V případě uspořádání s lichým počtem rotorů je nutné použít jiný způsob kompenzace, např. pomocí umístění jednoho z rotorů na pohyblivý kloub a tím řízení směru vytvářeného vztlaku.

Řídicí součásti zahrnují senzory, které poskytují informace o poloze UAV v prostředí a letovou řídicí jednotce, jež tyto vstupy zpracovává pomocí letového řídicího softwaru a na základě vstupů od pilota nebo požadavků autonomního řízení generuje řídicí zásahy pro motory nebo další aktuátory. Mezi senzory patří zejména Inertial Measurement Unit - inerciální měřicí jednotka (IMU), přijímač GPS a nějaký senzor výšky (např. barometr), které dohromady slouží k odhadu absolutní polohy v prostoru a následnému řízení letadla dle dalších požadavků (pilota či programu).

1.2 Přistávání

Konečnou fází letu je u letadel včetně UAV přistávání, které je pro jejich plně autonomní využití potřeba automatizovat. Na základě odhadu polohy je možné pomocí běžných řídicích systémů UAV (např. těch uvedených v kapitole 6) automaticky přistát na zvoleném místě. Problém však tvoří nedostatečná přesnost lokalizace a tím snížená bezpečnost manévrů.



OBRÁZEK 1.2: Blokové schéma zapojení základních řídicích a pohonné součástí kvadrokoptéry. [10]

Řešením nepřesnosti může být nějaké přídavné lokalizační zařízení (např. radar nebo systém real-time kinematic - RTK [37], který umožnuje výrazně zpřesnit odhad polohy pomocí GPS). Taková zařízení ale vyžadují další nastavení na místě použití UAV a jejich hardware je obvykle nákladný.

Vhodnou alternativou může být plošina s fiduciárním markerem, který bude detekovat kamera drunu a na základě jeho umístění a tvaru v obrazu určí polohu plošiny, na kterou tak vhodně řízené letadlo bude moci dosednout. definice, možnosti, navádění na cíl (opticky, proč ne jiné možnosti)

1.3 Struktura práce

Tato diplomová práce je rozdělena celkem do 11 kapitol (včetně této), přičemž kapitoly 1 až 6 jsou spíše teoretického rázu, uvádí čtenáře do problematiky, vymezují klíčové pojmy a popisují jejich příklady v kontextu práce. Jsou představeny různé fiduciární markery, simulátory letu UAV a řídicí programy včetně zhodnocení jejich použitelnosti s ohledem na cíle práce. Také jsou uvedeny různé metody přistávání, které se objevují v literatuře, a pro další zkoumání v praktické části jsou z nich odvozeny 4 vlastní metody.

Praktická část (kapitoly 7 až 10) se zabývá návrhem systému pro simulaci přistávání UAV (kapitola 7), pomocí kterého je možné implementované metody uplatnit v umělém světě a sledovat jejich vlastnosti, včetně grafického uživatelského rozhraní (GUI, kapitola 8), díky kterému může uživatel snadno spouštět simulaci za různých podmínek a zaznamenávat její výsledky. Zjištění jsou diskutována v kapitola 10. kapitole a celou práci shrnuje závěr (kapitola 11), po němž jsou uvedeny veškeré informační zdroje, ze kterých bylo čerpáno při vypracování. Implementace navrženého systému a metod přistávání je dostupná na webu na adrese: <https://github.com/pernik36/dp-uav-landing>.

2 Definice úlohy

Úloha přistávání na plošině, kterou se bude dále zabývat tato práce, je komplexní a algoritmy, které ji mohou efektivně řešit, v závislosti na implementaci vyžadují celou řadu vstupů. Je také potřeba vymezit specifika letadla, pro nějž bude úloha řešena, jako jsou na něm nesené senzory a způsob jeho ovládání, s čímž také souvisí volba řídicího softwaru (SW) letadla a volba simulátoru.

Letadlo použité pro simulaci má 4 rotory uspořádané do tvaru písmene X, vzdálenost mezi dvěma sousedními je asi 37 cm. Mezi jeho senzory patří

- inerciální měřicí jednotka (IMU), která zahrnuje akcelerometry a gyroskopy pro odhad polohy a rychlosti v 6 stupních volnosti,
- přijímač GPS pro odhad absolutní polohy,
- barometr, který je přesnější pro určování výšky než GPS a
- kamera namířená kolmo dolů pro sledování prostoru pod dronem a zachycení fiduciálního markeru umístěného na plošině.

Vnější vlivy jsou takové jevy, které mohou působit na letadlo nebo jeho senzory a tím ovlivnit jeho chování. Tato práce se zabývá působením větru a zastínění plošiny. Naopak se nezabývá vlivem teploty, přestože s klesající teplotou roste vnitřní elektrický odpor akumulátoru UAV ([15]), čímž klesá jeho využitelná energie i výkon a snížený maximální výkon může vést ke změně dynamice letu. Tento vliv nebyl zkoumán, protože ho běžné simulátory přímo nepodporují.

Plošina je tenká čtvercová deska o straně 70 cm, jejíž povrch tvoří fiduciální marker. Podrobněji o ní a fiduciálních markerech pojednává kapitola 3.

Simulovaný svět je tvořen podkladovou plochou s leteckým snímkem travnaté plochy Borského parku v Plzni, na které se v místě specifikovaném uživatelem nachází plošina a na níž je před začátkem simulace umístěn model UAV.

Simulátor používá simulovaný svět a napodobuje jeho fyzikální podstatu. Implementuje umělé senzory letadla, jejichž naměřené hodnoty předává jeho řídicímu softwaru a přijímá od něj řízení aktuátorů, které simuluje včetně jejich interakce s ostatními prvky světa. Napodobuje vnější podmínky jako vítr a osvětlení, které mohou působit na letadlo a přistávací algoritmus. O výběru použitého simulátoru a dalších podrobnostech pojednává kapitola 6.

Přistávací metoda je způsob řízení letadla v simulovaném světě na základě dat z různých jeho senzorů. Jedná se zejména o řízení polohy letadla v průběhu přistávání na

základě odhadu polohy vzájemného postavení UAV a plošiny v prostoru podle obrazových dat z kamery, ale také o další podpůrné činnosti, kterými může být například odhad rychlosti a směru větru.

Úlohou v této práci je potom využití konkrétní přistávací metody k řízení letadla vznášejícího se ve světě simulovaném simulátorem za působení vnějších vlivů v dohledu plošiny tak, aby na ni dosedlo. Při tom se sleduje odchylka od požadované trajektorie, doba trvání, výpočetní náročnost, poloha po dosednutí a další veličiny, podrobnosti o nich jsou v kapitole 9.

3 Plošina pro přistávání

Rovné místo určené pro přistávání UAV s vertikální dráhou přistávání se nazývá plošina pro přistávání, případně přistávací plošina nebo plocha. Může být pevná, přenosná či pohyblivá (s vlastní lokomocí). Pevné plošiny jsou přímo spjaty s místem jejich konstrukce, zatímco u přenosných a pohyblivých lze měnit místo přistání. Obvykle se takto označují místa pro přistání malých letadel, plošiny pro větší stroje by se pak označily jako helipad nebo heliport.

Všechna zmíněná místa většinou nesou nějaké vizuální označení, neboli fiduciární marker (sekce 3.1), které slouží k jednodušší orientaci pilota, případně může podporovat autonomní přistání. Základní a nejpoužívanější variantou označení je velké písmeno H umístěné v kružnici. V oblasti autonomních UAV se plošiny často označují strojově čitelnou jedinečnou značkou, která nese i informaci, pomocí které se dá identifikovat.

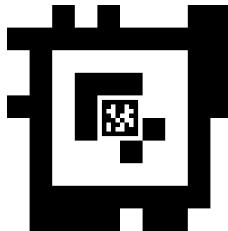
V této práci byla použita plošina zkonztruovaná jako deska tloušťky 1,5 cm tvaru čtverce o straně 70 cm. Celou plošinu vždy pokrýval rekurzivní fiduciární marker Apriltag o 2 vrstvách. Vnější vrstva o rozměrech 10x10 buněk měla uprostřed díru o rozměrech 2x2 buňky (rodina TagCustom48h12), do které byl umístěn tag vnitřní vrstvy o rozměrech 8x8 buněk (rodina Tag36h11) obklopený bílou oddělovací čárou o šířce 1 buňky vnitřního tagu. Rozměry značek jsou zde uvedeny včetně okraje, přičemž vnější značka měla jednu datovou vrstvu za okrajem.

Plošina byla vždy umístěna vodorovně do terénu tak, aby ji ve svislém směru nic nezakrývalo a v její blízkosti nebyly překážky, které by mohly přímo ovlivnit přistávání. Ve větší vzdálenosti mohla být umístěna překážka, která zastiňovala část plochy a tím i značky, címž mohla být negativně ovlivněna přesnost a spolehlivost její detekce, což bylo prozkoumáno v jednom z experimentů (podkapitola 9.6).

3.1 Fiduciární markery

Fiduciární markery (nebo také značky) mohou mít mnoho podob (tvary, vzory, uspořádání klíčových prvků) a oblastí uplatnění (snímkování v medicíně, mapování zemského povrchu, lokalizace zájmových bodů v obrazu, v robotice, identifikace jedinců atp.). Obecně se jako fiduciární marker označuje nějaký předmět, který slouží v zorném poli zobrazovacího přístroje k určení polohy daného zájmového bodu, jeho měřítka nebo jeho identifikaci. V robotice se podle jejich obrazu zachyceného kamerou může určovat vzájemná poloha kamery a předmětu, na němž je značka připevněna. [29]

Možnosti určování vzájemné polohy se využívá v této práci při odhadu umístění přistávací plošiny v prostoru s využitím kamery dronu. Dále budou uvedeny příklady některých používaných typů značek a popsány jejich základní vlastnosti. Pro hodnocení fiduciárních systémů, čili markerů a jejich detektorů, se používají metriky jako četnost falešných pozitiv, četnost záměny značky, četnost falešných negativ, minimální velikost značky [4], doba trvání detekce [6]; [11] a další specifické vlastnosti. O postupech používaných při detekci pojednává kapitola 4.



(A) Souosý dvouvrstvý AprilTag

OBRÁZEK 3.1: Příklady různých fiduciárních markerů

3.1.1 ARTag

ARTag je dvouodstínový fiduciární systém sestávající z 2002 značek, z nichž polovina má černé okraje a polovina bílé. Prostor mezi okraji je vyplněn mřížkou o rozměrech 6x6, přičemž každá z buněk může být bílá nebo černá. Kromě kódovaných bitů obsahuje navíc i kontrolní součet, s jehož využitím lze korigovat chyby (maximálně 2 bity z 36). Má nízkou četnost falešných pozitiv i četnost záměny značky za jinou. Ve srovnání se starším ARToolkit má vylepšenou identifikaci a verifikaci vzorů a snižuje tak četnost záměn značek. Přesto má větší knihovnu možných vzorů. [4]

3.1.2 Aruco

Fiduciární systém Aruco s bity kódovanými pomocí čtverců nevyužívá předdefinovanou knihovnu vzorů, ale v závislosti na požadavcích uživatele na velikost značky a slovníku generuje takové značky, které mají mezi sebou co nejvyšší vzdálenost, čímž se minimalizuje četnost záměn značek, a co největší počet přechodů mezi bity, čímž se minimalizuje četnost falešných pozitiv. Navíc je možné překrývat malé části značky a nenarušit tím detekci, nebo v aplikaci využívat masku předmětu, kterým je tag zakryt, což je výhodné při využití pro rozšířenou realitu. [6]

3.1.3 Apriltag

Dalším podobným systémem je Apriltag, který využívá podobné metody jako výše zmíněné systémy. Definuje různé typy, tzv. rodiny, značek v závislosti na velikosti slova ve slovníku a minimální požadované Hammingovy vzdálenosti mezi tagy. I v tomto případě je tak možné zjišťovat a opravovat chyby při detekci značky. [11]

Ve verzi 3 je navíc dvakrát rychlejší detektor než ve verzi 2, umožňuje použití na míru navržených tvarů (např. kruhových nebo dokonce s dírou) a data mohou být až za okrajem tagu, což zvyšuje datovou hustotu. Do prázdného místa uprostřed tagu je možné umístit další tag a vytvořit tak rekurzivní značku, kterou bude možné detektovat ve větším intervalu vzdáleností [21], toho bylo využito v této práci jako značky na přistávací plošině (obrázek 3.1a).

4 Detekce fiduciárních markerů

Během detekce fiduciárních značek se využívá zejména algoritmů počítačového vidění za účelem zvýraznění a extrakce informace související s tagem a naopak potlačení pozadí a také případně nalezení a lokalizace klíčových bodů, které mohou sloužit pro odhad polohy značky v prostoru v případě známého rozměru tagu a kalibrované kamery. Jako detekce se označuje proces nalezení značky v obrazu, algoritmus, který toto provádí se pak nazývá detektor.

Mezi algoritmy používané při detekci fiduciárních markerů patří prahování, vyhledávání vzorů (template matching), hranové detektory [11] nebo Houghova transformace [24]. Dále je popsáno, jak se některé z těchto metod uplatňují v detektoru použitém v praktické části (AprilTag).

AprilTag. Detektor AprilTagů se skládá z pěti dílčích kroků:

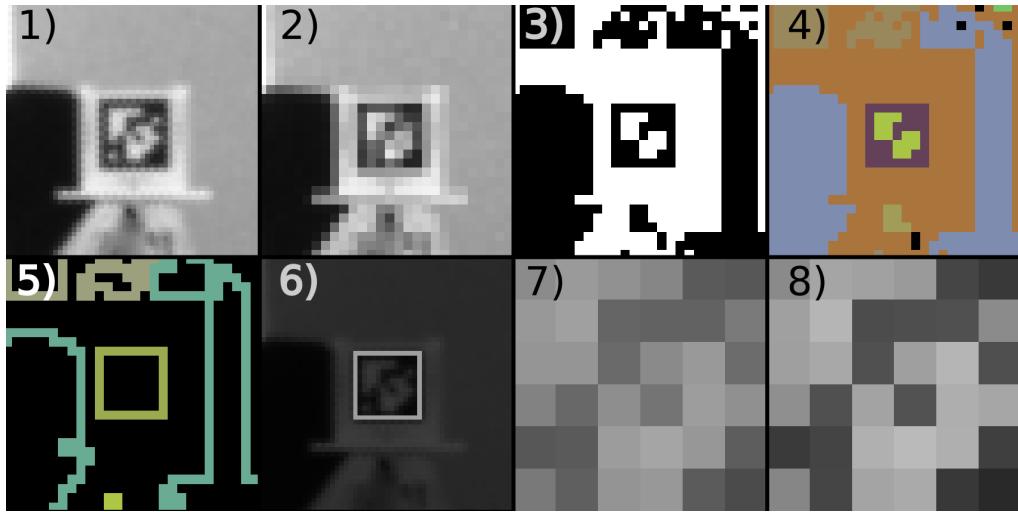
1. Adaptivního prahování,
2. segmentace souvislých hranic,
3. approximace čtyřúhelníků,
4. rychlého dekódování a
5. volitelného zpřesnění hran.

Tyto a další kroky na reálném vstupním obrázku ilustruje obrázek 4.1.

Adaptivní prahování volí práh vždy lokálně v závislosti na okolí prahovaného bodu jako aritmetický průměr minimálního a maximálního jasu. Pro snížení výpočetní náročnosti je v tomto kroku vstupní obraz navíc decimován tak, že se extrémy hledají vždy v buňkách po 4x4 pixelech, ve verzi 2 [11] se autoři chtějí vyhnout nespojitostem na hranicích buněk a tak používají extrémy z osmiokolí (po buňkách). Ve verzi 3 [21] potom zjistili, že je vhodnější decimaci oddělit od prahování a používat bodové převzorkování, které lépe zachová hrany v obrazu, ale může vést na aliasing. Zachování hran je vhodné, protože se v dalším kroku detekují. Málo kontrastní body, které vzejdou z prahování se v dalším zpracování neuvažují, čímž jsou další kroky zrychleny.

Segmentace souvislých hranic probíhá tím způsobem, že se nejprve najdou hrany v prahovaném obrazu, čili takové pixely, které sousedí s pixelem opačné barvy, jež se následně slučují do hranic pomocí algoritmu union-find. Problém, kdy jsou dva velmi blízké tagy odděleny pouze tenkou linií bílých pixelů, který by při sloučení hranic znemožňoval detekovat oba tagy zároveň, je vyřešen tak, že bílé pixely mohou být součástí 2 hranic. Každé takto segmentované části hranice je přiřazeno nějaké číslo unikátní v rámci obrázku (tzv. obarvení). [11]

Zrychlení segmentace ve verzi 3 je docíleno tím, že se v implementaci algoritmu zbytečně nesjednocují již sjednocené části (ušetří se volání sjednocení), navíc se včasně zamítají příliš malé oblasti, které by nemohly vést na dekódovatelný tag. [21]



OBRÁZEK 4.1: Postup při detekci AprilTagů. 1) je vstup, 2) je vstup po decimaci, 3) prahování, 4) union-find, 5) segmentace souvislých hranic, 6) approximace čtyřúhelníků z hranic, 7) vyrovnání, 8) zvýšení ostrosti a dekódování. Převzato z [21].

Aproximace čtyřúhelníku. Pro každé sjednocení z předchozího kroku je potřeba najít vhodný čtyřúhelník, tzn. rozdělit množinu hraničních bodů do 4 skupin, které odpovídají jednotlivým stranám čtyřúhelníku. Hledání optimálního řešení je příliš výpočetně náročné, proto se postupuje tak, že se naleznou kandidáti na rohy a poté se projdou všechny jejich kombinace. Rohy se hledají tak, že se body nejprve seřadí podle úhlu průvodiče vedeného z centroidu množiny a jednotlivými body, poté se postupně approximují body v posuvném okně přímou a hledají se maxima střední kvadratické chyby (MSE), odpovídající body jsou prohlášeny za rohy. Pro každou podmnožinu 4 rohů se zbylé body rozdělí na strany a každá z nich se approximuje přímou. Vybere se taková kombinace rohů, pro kterou je MSE nejnižší. [11]

Ve verzi 3 se řazení bodů provádí podle kvadrantu, ve kterém se bod nachází a sklonu průvodiče. Není tak nutné počítat explicitně úhel. [21]

Rychlé dekódování tagů v nalezených čtyřúhelnících probíhá tak, že se nejprve vyrovnaní (pomocí homografie vypočtené ze souřadnic polohy jejich rohů) a hledá se nejbližší kód z dané rodiny značek pro každou ze 4 možných rotací. Omezí-li se počet odlišných bitů na 2, je možné předpočítat všechny tagy maximálně 2 bity vzdálené od validního tagu a zaznamenat je do hashovací tabulky. Při detekci se pak z tabulky jen vybere příslušná hodnota, nebo se detekce zamítne. [11]

Verze 3 zavádí bilineární interpolaci buněk tagu a zvyšuje ostrost za účelem zlepšení detekce malých značek. [21]

5 Metody přistávání

Součástí této práce je návrh systému pro simulaci přistání bezpilotního letounu na plošině, který umožnuje implementovat libovolnou metodu navádění letounu na plošinu a jeho dosednutí tak, aby bylo možné různé metody vzájemně porovnat z různých hledisek a za různých podmínek a případně tak stanovit, za jakých okolností je vhodné použít daný přístup. Tato kapitola shrnuje metody, které se objevují v další literatuře, a popisuje metody, které byly implementovány, simulovány a následně porovnány mezi sebou (o průběhu simulace, porovnání a výsledcích dále pojednává kapitola 9). Všechny 4 implementované metody mají společný základ, na kterém postupně staví s využitím dalších dílčích stavebních bloků a to samostatně nebo v jejich kombinaci.

Literatura se zabývá 2 druhy metod podle toho, jestli je cíl přistávání kooperativní (umělý, specificky navržený, aby podporoval danou metodu přistávání) nebo nekooperativní (přirozený, místo v prostředí, které je vhodné pro přistání a UAV ho může samostatně detekovat). Metody určené pro použití s nekooperativními cíli jsou obvykle komplexnější a kladou větší nároky na autonomii UAV, ale mají tu výhodu, že není vyžadována manuální pokládka cíle na zem jako v případě metod pro kooperativní cíle. To umožňuje jejich použití i v náročnějších podmínkách např. při záchranných akcích po přírodních katastrofách. [35], [33]

Součástí těchto metod je často algoritmus pro plánování trajektorie, který vede trasu dronu tak, aby se vyhnul detekovaným překážkám nebo oblastem, ve kterých by nebylo vhodné letět (např. v blízkosti osob, aby byla zajistěna jejich bezpečnost). [33]

Metody s kooperativními cíli přistávání lze dále rozlišovat podle způsobu detekce a rozpoznávání daného markeru, což je považováno za nejdůležitější krok celého přitsávání. Existují klasické metody založené na příznacích v obrazu, jež používají uměle vytvořené přesně definované značky s geometrickým vzorem nebo takové, které uplatňují nějakou geometrickou závislost, přičemž způsob návrhu může výrazně ovlivnit schopnosti autonomního přistání UAV (příklady značek jsou uvedeny v kapitole 3). [35]

Klasické metody používají například algoritmy pro extrakci klíčových bodů jako jsou SIFT, SURF nebo ORB [32] nebo složitější algoritmy navržené přímo na míru pro dané markery, sestávající z obecnějších a jednodušších stavebních bloků (např. [21], kapitola 4).

Další skupinou metod detekce kooperativních cílů jsou ty, které jsou založené na hlubokém učení. Navrhují se takovým způsobem, aby výsledný detekční algoritmus optimalizoval rychlosť detekce, přesnost a jednoduchost modelu a aby se uměl vypořádat s komplexními scénami, ve kterých může být snížená viditelnost vlivem rozptýlených částic ve vzduchu (mlha, opar, prach) nebo může být značka rozmazaná (např. pohybem kamery). [35]

Mezi detekční metody s hlubokým učením, které by mohly být použity při přistávání na kooperativní cíle patří například architektury Faster R-CNN, YOLO nebo LightDense-YOLO, které dosahují rychlosti detekce v řádu desítek milisekund a nepřesnosti do 1,5 cm z 5 m výšky. Samotná detekce hranic může být rychlejší než tradiční metody s využitím Houghovy transformace. Použití takovýchto metod může zvýšit robustnost, ale výpočetní náročnost je vyšší. [35]

Samotné metody přistávání se svou koncepcí mohou vzájemně zcela lišit. V [23] je navržen systém pro zpětnovazební učení přistávací strategie pomocí opakování simulace v simulátoru Gazebo. Tento systém je rozšířitelný o další algoritmy zpětnovazebního učení, jiné roboty i prostředí a umožňuje v jednotném rozhraní agentům interagovat s jejich okolím tak, aby byla maximalizována jejich cílová funkce. Oproti jiným přístupům museli autoři používat dočasné zastavení simulačního času, protože je to vyžadováno při jejich použití algoritmů zpětnovazebního učení. Autoři [33] navrhují celý postup pro bezpečné přistávání bez využití kooperativního cíle, který sestává mimo jiné z těchto dílčích částí:

1. Detekce potenciálního místa pro přistávání.
2. Vizuální detekce místa pro přistání s využitím sémantické segmentace obrazu.
3. Detekce davů lidí.
4. Detekce osob.
5. Jednoduché plánování trasy.

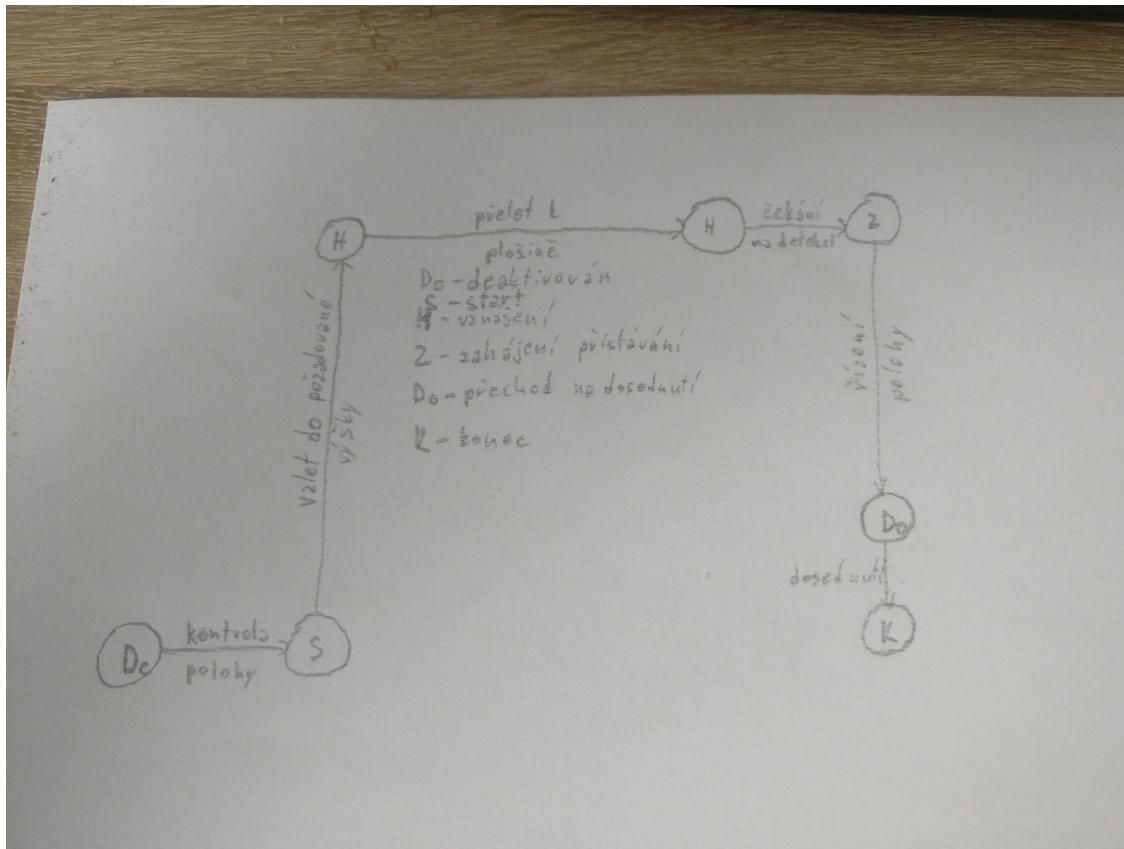
Tento postup se soustřeďuje na přistávání ve venkovním prostoru s důrazem na porozuření prostředí, kdy je během letu vytvářena mapa, která je poté použita při volbě místa přistání a samotném přistávání.

Článek [32] představuje simulační systém používající simulátor Gazebo a vizuální data pro přistávání UAV. Pro tracking plošiny využívá Kálmánův filtr na obrazových datech v simulovaném i skutečném dronu a navrhuje řídící systém vyvinutý pro kontrolér PX4, který umožňuje strategii snadno přenést na fyzické letadlo. Pro detekci plošiny jsou používány příznakové metody a není odhadována vzájemná poloha dronu a plošiny, takže není nutné zavádět data z IMU do trackovacího modulu, čímž je snížena výpočetní náročnost a umožněn výpočet přímo na palubním počítači. Pomocí integrovaných kaskádních PID regulátorů se na základě trackovacích dat řídí poloha a rychlosť UAV tak, aby bylo zajištěno úspěšné přistání. Systém je robustní k náhlým změnám v obrazu a na základě odhadů z Kálmánova filtru je schopen dron řídit i při krátkém přerušení viditelnosti markeru.

V jiném kontextu používají Kálmánův filtr a hluboké učení autoři [34], kteří navrhovali systém pro přistávání na pojízdné plošině a modelovali její pohyb. Porovnávali predikce pouze rekurentní neuronové sítě LSTM, pouze Kálmánova filtru a jejich kombinace, přičemž poslední zmíněná možnost měla nejlepší přesnost a zejména při zahnutých trajektoriích plošiny umožňovala lepsí plánování přistávací trajektorie.

Podobná úloha může být řešena také za předpokladu, že přistávací plošina opustí zorné pole kamery. V takovém případě je také vhodné modelovat její pohyb na základě několika po sobě jdoucích snímků a díky predikci je možné dosáhnout úspěšného přistání po naplnění trajektorie. Výhodou tohoto přístupu je, že není nutné používat gimbal v konečné fázi přistávání. [19]

V metodách popsaných dále v této kapitole, implementovaných a simulovaných v praktické části se využívá podobného přístupu jako v [32], ale Kálmánův filtr je použit přímo na polohu dronu, protože je přístup přímočařejší a výpočty jsou prováděny na osobním počítači, takže požadavek na výpočetní náročnost není tak kritický. Podobného efektu jako dosáhli autoři [14], kteří v průběhu přistávání měnili velikost a typ fiduciárního markeru zobrazeného na displeji, bylo dosaženo zakomponováním menšího markeru do prázdné části většího, jak již bylo popsáno v kapitole 3.



OBRÁZEK 5.1: Činnosti obecného přistávacího algoritmu a jejich posloupnost.

5.1 Obecné vlastnosti implementovaných metod

Bez ohledu na metodu jsou některé činnosti, které jsou prováděny před samotným přistáváním a v jeho průběhu, podmínky a vlivy stejné. Tato podkapitola zachycuje takové společné rysy metod implementovaných pro simulaci v této práci. Činnosti a jejich sled ukazuje diagram v obrázku 5.1.

Na začátku simulace je dron umístěn na zemi a není aktivní. Aby bylo možné simuloval přistání, je s ním nejprve nutné vzletět, čehož se docílí tak, že se aktivuje a řídící jednotce se zadá příkaz pro vznesení a přelet do dané výšky. Následně je možné začít přistávání za následujících podmínek:

1. Letoun se vznáší ve vzduchu a je skoro¹ v klidu.
2. Je znám odhad polohy letounu pomocí GPS a IMU je správně kalibrovaná.
3. Letoun má dostatek energie pro přelet nad plošinu a následné přistání za daných přírodních podmínek.

¹Až na drobné nuance způsobené chybami měření palubních senzorů a náhodnými vlivy prostředí.

4. Okamžitá rychlosť v_o a směr větru ϕ_o jsou náhodné s normálním rozdělením

$$\begin{bmatrix} v_o \\ \phi_o \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} v \\ \phi \end{bmatrix}; \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \right)$$

filtrované dolní propustí 1. řádu s časovou konstantou $\tau_v = 1,59$ pro rychlosť a $\tau_\phi = 4,77$ pro směr. Parametry normálního rozdělení v, ϕ, σ_v a σ_ϕ jsou voleny uživatelem systému pro každé přistání.

5. Přistávací plošina je částečně zastíněná, podíl zastíněné části je p_s a úhel mezi hranou stínu a severo-jižním směrem je θ .

6. Je známa přibližná geografická poloha plošiny.

Z těchto podmínek platí 2., 4., 5. a 6. po celou dobu přistávání. Pro přistání je nejprve nutné přeletět do blízkosti přistávací plošiny tak, aby mohla být zachycena kamerou umístěnou na palubě UAV. Řídící jednotce se tedy zadá příkaz pro vodorovný přelet na známé přibližné umístění plošiny. Je-li po přeletu detekovatelný fiduciální marker plošiny, může se k ní UAV začít přibližovat. Způsob přibližování je závislý na konkrétní metodě a právě jím se jednotlivé metody liší. Pokud se naopak nepodaří značku detektovat, přistání selhává. Může to být způsobeno příliš velkou nepřesností v poloze plošiny, příliš velkou svislou vzdáleností od plošiny, nebo příliš velkým větrem, který letoun vychyluje od vodorovné polohy natolik, že tag je mimo zorné pole kamery.

I přes rozdíly, které jsou popsány dále, se využívá stejného způsobu ovládání dronu. Konkrétně se jedná PID regulátor rychlosti v jednotlivých osách podle vzdálenosti od požadované trajektorie a rotace ψ okolo svislé osy (osa z) podle odchylky od základního směru (SJ směr, osa y). Jeho výstup je prostřednictvím režimu řízení Offboard zaveden do vnitřní kaskády regulátorů řídící jednotky UAV.

V následujících podkapitolách (5.2 až 5.5) jsou popsány způsoby přiblížení použité v implementovaných metodách.

5.2 Přistávání po svislé přímce

Přistávání po svislé přímce je nejjednodušší z implementovaných metod přistávání. Spojívá v tom, že se letoun řídí tak, aby jeho vodorovná vzdálenost od středu plošiny byla během přistávání nulová. Dle velikosti odchylky se řídí rychlosť klesání, dokud dron nedosedne na plošinu. Poté je přistávání ukončeno. Požadovaná rychlosť klesání se stanoví podle vzorce (5.1), kde d je vodorovná vzdálenost UAV od středu plošiny (případně požadované trajektorie v dané výšce u některých jiných metod) a h je výška nad povrchem, ve které se letoun nachází v daném okamžiku.

$$v_z = \frac{1}{2 \cdot \left(1 + \frac{12 \cdot d^2}{h} \right)} \quad (5.1)$$

Vzájemná poloha letadla a středu značky je odhadována detektorem zvolených fiduciálních markerů, tedy AprilTagů, na základě obrazových dat z kamery. Způsob odhadování polohy je podrobněji popsán v kapitole 4. Aby bylo možné relativní vzdálenost určit správně a ve správném měřítku, musí být známá kalibrace kamery a rozměr značky.

Vztah (5.1) byl tímto způsobem stanoven, aby funkce rychlosť byla hladká a v případě odchylky ji bylo možné korigovat před tím, než fiduciální značka opustí zorné pole kamery

a nebude tak možné dále odhadovat polohu plošiny. Pokud by k tomu i tak došlo (např. vlivem silného poryvu větru), dron bude po omezenou dobu (max. 50 snímků bez tagu) stoupat konstantní rychlostí danou v konfiguračním souboru (výchozí hodnota je $v_z = 0,3 \text{ m/s}^2$) a čekat na snímek z kamery, který by obsahoval tag. Nedoje-de-li k tomu včas, pokus o přistání se opakuje od přeletu nad plošinu.

5.3 Přistávání po svislé přímce s využitím Kálmánova filtru

Tento algoritmus je přímo odvozený od předchozího, stejně jako u něj je požadována nulová vodorovná vzdálenost dronu a plošiny během přistávání a výpočet svislé rychlosti se také neliší. Rozdíl spočívá v tom, že je sestaven jednoduchý dynamický model letounu, který se následně využívá pro predikci stavu v Kálmánově filtru [1], [2] a měření polohy získaná z detektoru, která jsou zatížená šumem, se pak filtruji, čímž je dosaženo vyšší přesnosti odhadu polohy. Filtrované odhady se dále používají stejným způsobem jako v předchozí metodě a je podle nich řízena vodorovná rychlosť UAV.

Pro tyto účely byl letoun modelován lineárním stochastickým systémem diskrétním v čase jako hmotný bod se směrem rotace, jehož stav je dán polohou v 3D prostoru a rychlostí v příslušných souřadnicích, uvažovala se také rotace ψ kolem osy z úhlová rychlosť $\dot{\psi}$ v téže ose. Vstupem systému je požadované zrychlení (nebo úhlové zrychlení) v jednotlivých proměnných, které je dané řízením dronu. Kromě vstupu na systém dále působí šum s normálním rozdělením $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ s kovarianční maticí \mathbf{Q} , která byla určena řádově podobně jako v [8] a dále podle obvyklých kovariancí chyb IMU. Vývoj stavu potom určuje rovnice (5.2).

Měření z uvedeného modelu jsou přímo 3 polohové souřadnice (x, y, z) a rotace ψ . I měření je navíc zatíženo náhodnou chybou modelovanou náhodnou veličinou s normálním rozdělením $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. Jeho kovarianční matice \mathbf{R} byla odhadnuta jako výběrová kovarianční matice měření odchylek relativní polohy měřené detektorem od skutečné relativní polohy v simulátoru. Měření \mathbf{z} potom popisuje rovnice (5.3).

Pro fungování Kálmánova filtru je důležitý i vývoj kovarianční matice stavu \mathbf{P} (protože stav je náhodná veličina), kterou je nutné inicializovat pro čas $k = 0$. Tato kovarianční matice byla také odhadnuta jako výběrová pro odchylky požadované polohy pro přelet od skutečné polohy po přeletu. Tento způsob odhadu kovarianční matice odpovídá skutečnému průběhu simulace přistávání, jak byla popsána v podkapitole 5.1. Použitý model je

tedy dán následujícími rovnicemi:

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \quad \text{kde} \quad (5.2)$$

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x}_k = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ z_k \\ \dot{z}_k \\ \psi_k \\ \dot{\psi}_k \end{bmatrix},$$

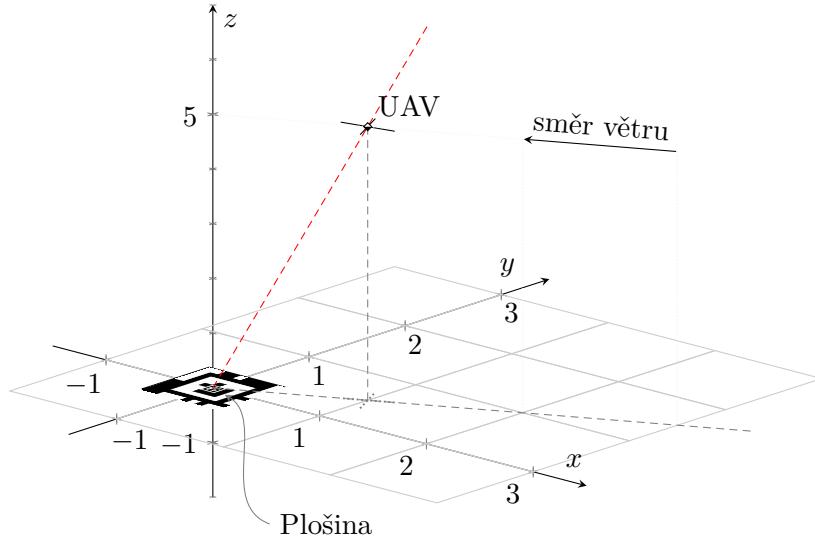
$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix}, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \text{kde}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,002 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,002 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0,004 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,02 \end{bmatrix}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \quad \text{kde} \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad (5.3)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Mezi aktualizacemi senzorů, jejichž měření jsou do řídicího programu zasílána prostřednictvím protokolu MAVLink, jsou hodnoty veličin považovány za konstantní. Frekvence aktualizací je $\frac{1}{\Delta t} = 30$ Hz. Tato frekvence je shodná se snímkovou frekvencí použité kamery a je od ní odvozena i délka kroku Kálmánova filtru, která tedy činí $\Delta t = \frac{1}{30}$ s.



OBRÁZEK 5.2: Přistávání po skloněné přímce (červeně). Letadlo je na kloněné proti větru, čímž ho překonává a zároveň kamerou míří na střed plošiny, která je tak v zorném poli i za přítomnosti poruch.

5.4 Přistávání po skloněné přímce

Tento způsob přistávání také vychází z prvního zmíněného (sekce 5.2), ale narozdíl od předchozího s Kálmánovým filtrem (sekce 5.3) neupravuje měření relativní polohy, ale požadovanou polohu v průběhu přistávání.

V předchozích případech je požadovaná relativní horizontální poloha nulová a požadovaná trajektorie je tedy svislá přímka. V tomto případě je proměnná a je závislá na výšce, ve které se letoun v daný okamžik nachází, a na odhadu středních hodnot rotací okolo os y a x během vznášení v klidu, které jsou dány tím, jak dron překonává síly působené větrným prouděním. Požadovanou trajektorií je přímka s takovým sklonem, aby pro dané rotace (a tím i pro danou rychlosť a daný směr větru) kamera UAV mířila na střed přistávací plošiny. Rychlosť se v závislosti na odchylce od požadované trajektorie vypočte stejným způsobem jako v předchozích případech, tedy dle vztahu (5.1).

Tato metoda má za cíl zlepšit podmínky pro detekci značky na plošině při působení silného větru. V případě nutné korekce nějaké poruchy (způsobené například poryvem větru) zbývá okolo tagu více místa v obrazu, kam se může posunout, aniž by opustil zorné pole a tím znemožnil odhadovat svoji polohu vůči letounu.

5.5 Přistávání po skloněné přímce s využitím Kálmánova filtru

Poslední implementovaný přístup využívá obou změn navržených v předchozích dvou upravených metodách (sekce 5.3 a 5.4). Pro odhad vzájemné polohy dronu a značky na přistávací plošině je aplikován Kálmánův filtr s měřeními z detektoru tagů a lineárním stochastickým systémem jako modelem pohybu a rotace dronu v prostoru. Stejně jako ve všech implementovaných metodách pro přistávání je tento odhad využit PID regulátorem, který řídí požadovanou rychlosť UAV ve dvou vodorovných osách tak, aby byla následována požadovaná trajektorie, a jeho výstup je zapojený do interní kaskády regulátorů

řídicí jednotky letounu. V tomto případě je základní svislá trajektorie nahrazena skloňenou trajektorií proti směru větru, aby i při jeho překonávání letounem zůstala značka na plošině uprostřed zorného pole kamery, jak již bylo popsáno v předchozí podkapitole. Všechny tyto popsané metody byly podrobeny experimentům za různých podmínek a byla vyhodnocena jejich úspěšnost, přesnost a další vlastnosti. Popis těchto experimentů je v kapitole 9.

6 Simulátory bezpilotních letadel

Simulace umožňuje zjednodušit vývoj nových algoritmů tím, že zvyšuje bezpečnost, sniže režii a umožňuje rychlejší iteraci úprav a testování. V případě zkoumání vnějších vlivů na systém, jehož je algoritmus součástí, má umělé prostředí výhodu v tom, že jeho parametry jsou pod kontrolou vývojáře, který tak může systém testovat v širší škále podmínek a ověřit a zhodnotit jeho funkčnost i za podmínek (nebo jejich kombinace), které jsou v realitě v daném místě málo obvyklé. Rizikem je možná nepřesnost simulace, která způsobí odlišné chování systému při reálném nasazení, z toho důvodu je obvyklé simulační výsledky na konci vývoje validovat, jestli odpovídají realitě.

Právě z důvodu minimalizace potřebných úprav a co nejsnazšího přenosu systému do reality je vhodné pro vývoj zvolit takový simulátor, který umožňuje propojení se skutečným letovým řídicím softwarem v řídicí jednotce letounu, jenž může být také simulovaný. V dokumentaci některých letových softwarů se simulaci přímo věnují vybrané kapitoly a nabízí různé podporované simulátory, které je možné spolu s nimi spustit [28], [13], [38].

Mezi rozšířené řídicí softwary patří ArduPilot, Betaflight a PX4. Všechny podporují běh v simulovaném prostředí a spolupracují s vybranými simulátory. ArduPilot a PX4 se mimo jiné svou funkcionalitou zaměřují i na provádění autonomních misí, zatímco Betaflight je určen spíše pro lety s pohledem z první osoby například během závodů, kde je vyžadována velmi rychlá a přesná odezva na vstupy z rádiového dálkového ovládání od pilota a poruchy způsobené například pohybem vzduchu v atmosféře. S různými zaměřeními souvisí i podporované způsoby ovládání, kdy ArduPilot a PX4 je možné řídit vzdáleně pomocí protokolu MAVLink i rádiově modelářským ovladačem, zatímco Betaflight přes MAVLink umí pouze vysílat telemetrii a s jiným než rádiovým ovládáním se příliš nepočítá. Podporované simulátory se mezi řídicími softwary částečně překrývají, jejich přehled je uveden v tabulce 6.1, zdrojem dat je oficiální dokumentace příslušného kontroléru [28], [13], [38].

Funkcemi se v tabulce 6.1 uvedené simulátory mohou výrazně lišit a pro další použití v této práci je třeba vybrat takový, ve kterém bude možné provozovat navádění na plošinu a to nejlépe na základě obrazu a simulovat různé vlivy prostředí. V následujících podkapitolách (sekce 6.1 až 6.6) jsou zevrubně popsány hlavní funkce jednotlivých simulátorů a jejich

Simulátor \ Kontrolér	ArduPilot	Betaflight	PX4
Gazebo	ANO	ANO	ANO
RealFlight	ANO	ANO	NE
jMAVSIM	NE	NE	ANO
FlightGear	NE	NE	ANO
JSBSim	ANO	NE	ANO
AirSim	ANO	NE	ANO

TABULKA 6.1: Přehled podpory často používaných simulátorů vybranými letovými řídicími souftwary.

	Gazebo	RealFlight	jMAVSim	FlightGear	JSBSim	AirSim
def. prostředí	ANO	ANO	?	ANO	-	ANO
vložení plošiny	ANO	ANO	ČÁST.	ANO	-	ANO
světelné podm.	ANO	ANO	NE	ANO	-	ANO
stíny	ANO	ANO	NE	ANO	-	ANO
kamera	ANO	NE	ANO	NE	-	ANO
simulace větru	ANO	ANO	ANO	ANO	ANO	ANO
viditelnost	ANO	ANO	NE	ANO	-	ANO
teplota	ČÁST.	NE	NE	ANO	ANO	NE
dyn. změny	ČÁST.	?	ČÁST.	ANO	ANO	ANO

TABULKA 6.2: Podpora některých funkcí a vlastností, které jsou důležité pro návrh simulačního systému pro přistávání UAV, vybranými simulátory. „ANO“ znamená, že je daná funkce simulátorem zcela podporována; „ČÁST.“, neboli částečně, je uvedeno u funkcí s omezenou podporou, jež nelze zcela použít, např. funkce implementovaná, která nemá žádný vliv na simulovaný model; „NE“ se uvádí u chybějící funkce simulátoru; „-“ vyznačuje funkci nepodporovanou z důvodu, že simulátor má jiné zaměření a nesplňuje podmínky pro implementaci takové funkce; „?“ znamená to, že ze zdrojů dostupných autorovi nebylo možné spolehlivě určit, zda má simulátor danou funkcionalitu. Zkratky v tabulce: Def. prostředí znamená uživatelská definice prostředí, podm. jsou podmínky a dyn. znamená dynamické změny ostatních simulačních podmínek.

obecný popis. Porovnání jejich vlastností důležitých pro tuto práci je uvedené v tabulce 6.2.

6.1 Gazebo

Gazebo je simulátor vyvíjený nadací Open Source Robotics Foundation. Je výchozím simulátorem v Robot Operating System, díky čemuž se těší velké oblibě při 3D simulačních dynamiky robotů. Má velmi aktivní komunitu. Umožňuje využití různých fyzikálních enginů, modelů senzorů a podporuje snadné vytváření 3D světů. Díky tomu lze snadno testovat návrhy robotů a algoritmů nebo i trénovat systémy strojového učení pomocí realistických scénářů. Gazebo používá modulární architekturu s oddělenými knihovnami pro simulaci fyziky, vykreslování, uživatelské rozhraní, komunikaci a generování dat senzorů. Podporované UAV zahrnují čtyřrotorové letouny (Iris a Solo), šestirotorové letouny (Typhoon H480), různé konvertoplány (např. letadla s plochou dráhou letu a svislým startem a přistáním), běžná letadla nebo pozemní vozítka. I když je Gazebo bohatou platformou, vykreslovací techniky nejsou tak pokročilé jako například v Unreal Engine nebo Unity, které využívají jiné simulátory. [20]

Co se týče simulace vnějších vlivů, Gazebo nativně umožňuje měnit podmínky osvětlení, simulaci větru je možné provést pomocí doplňku distribuovaného společně s Gazeboem a v definici modelu je možné upravit, jak bude vítr působit na model. Teplotu i její senzor je možné také simulovat pluginem, ale působení na model není implementované. Ve starších verzích Gazebo označovně jako Gazebo classic je možné simulovat chování baterie pomocí pluginu [36], ale změna dynamiky způsobená změnou vnitřního odporu baterie není ani s tímto pluginem přímočará.

6.2 RealFlight

RealFlight je komerční simulátor s 3D zobrazením, který umožňuje návrh a testování vlastních letounů. Podporuje simulaci různého počasí, včetně zhoršené viditelnosti, větru a změn podmínek osvětlení. Objekty v simulátoru také mohou vrhat stíny. Je zaměřen spíše na výcvik pilotů rádiem ovládaných modelů letadel a není možné ho spustit bez připojeného ovladače [16], [41]. Kromě toho oficiální dokumentace nezmiňuje možnosti spouštění přímo s různými nastaveními vnějších vlivů [41], což by ztěžovalo vývoj, ani možnosti využití obrazových dat. Není tedy příliš vhodnou variantou pro použití v této práci. Narozdíl od Gazeba není podporován kontrolérem PX4.

6.3 jMAVSIM

Java Micro Air Vehicle Simulator (jMAVSIM) je jednoduchý a lehký simulátor multirotorových letounů vyvinutý týmem PIXHAWK engineering. Podporuje protokol MAVLink, používá knihovnu Java3D pro vizualizaci a připojuje se přímo k HIL pomocí sériového spojení nebo k Software-in-the-Loop (SITL) pomocí komunikace přes User Datagram Protocol (UDP) k řídicímu softwaru PX4. Má jednoduchou monolitickou architekturu. [20]

Podporuje ho jediný z vybraných kontrolérů a to PX4. Zaměřuje se na simulaci dynamiky dronu, ale vyobrazení světa není příliš realistické, dokumentace nezmiňuje, jak měnit světelné podmínky a vytvářet stíny. Umožňuje umístění kamery na model a přenos obrazu přímo přes MAVLink, tímto způsobem může být přenos řešen i v reálném UAV. Také v něm lze nastavovat vlastnosti větru. [5], [27]

Nemožnost simulace různých světelných podmínek, obtížná rozšířitelnost a málo rozsáhlá dokumentace nejsou pro tuto práci vhodné.

6.4 FlightGear

FlightGear je pokročilý open-source simulátor, který je snadno rozšířitelný a má mnoho funkcí, které ho přibližují realitě. Simuluje polohu hlavních vesmírných těles (Slunce, Měsíc, planety sluneční soustavy a některé viditelné hvězdy) v závislosti na čase a poloze, je dodáván s modelem povrchu Země, který je rozšířitelný o vlastní modely. Lze v něm simulovat počasí (vítr, srážky, změna osvětlení) a zobrazuje i stíny objektů. Podpora pro virtuální kameru umístěnou na modelu v dokumentaci není zmiňována. Má podporu pro různé modely dynamiky letounů včetně JSBSim (o tom dále v podkapitole 6.5). [39]

Jako jediný z vybraných kontrolérů ho podporuje PX4. Kvůli obtížnému použití s kamerou a zbytečně mnoha funkcím, které by se musely nastavit, je jeho vhodnost pro použití v této práci omezená.

6.5 JSBSim

JSBSim je open source model letové dynamiky, který definuje pohyb letadla, rakety apod. pod vlivem sil a momentů, které na něj působí pomocí různých řídících mechanismů nebo přirozeně. JSBSim nemá vlastní grafiku. Může být spuštěn sám o sobě jako samostatný program, který bere vstup ze skriptovacího souboru a různých souborů konfigurace vozidel. Lze jej také začlenit do větší implementace leteckého simulátoru, který zahrnuje vizuální systém. Nejznámějšími příklady použití JSBSim jsou v současnosti simulátory FlightGear (open source), Outerra, BoozSimulator (open source) a OpenEagles (open source). [40]

Tento model lze použít pro simulaci s kontroléry ArduPilot a PX4, které ho oba podporují. Přestože je možné pomocí JSBSim dobře modelovat dynamiku letounů, jeho samostatné využití pro simulaci přistání UAV by nebylo vhodné, kvůli absenci grafického prostředí a tím i kamery a dále kvůli obtížné přenositelnosti na reálný systém, protože simulace probíhá dávkově.

6.6 Airsim

Aerial Informatics and Robotics Platform (AirSim) byla vyvinuta společností Microsoft a klade si za cíl podporovat vývoj a testování algoritmů pro aplikace autonomních vozidel, jako jsou algoritmy hlubokého učení, počítáčového vidění a algoritmy zpětnovazebního učení. Fyzikální engine AirSim je založen na Unreal Engine 4 a může pracovat s vysokou frekvencí pro simulace HIL v reálném čase s podporou populárních protokolů (např. MavLink). Simulátor odpovídá modulární architektuře s důrazem na rozšiřitelnost. [20]

Mezi její funkce, které by byly užitečné při návrhu systému pro simulaci přistávání, patří realistická grafika včetně zobrazování stínů, kterou je možno pozorovat pomocí virtuálních kamer implicitně umístěných na modelu, simulace atmosférických jevů, které ovlivňují viditelnost jako srážky, ve vzduchu rozptýlené prachové částice, mlha a také různé světelné podmínky. Kromě toho má bohaté API a všechny aspekty simulovaného světa je možné dynamicky upravit narozdíl například od Gazebo, kde je nutné tato nastavení předem určit v definičním souboru světa. [18]

Mezi podporované letové řídicí softwary patří u této simulační platformy stejně jako u JSBSim ArduPilot a PX4. AirSim je ve srovnání s ostatními zmíněnými simulátory mladším projektem a navíc měly být práce na ní v roce 2022 zastaveny kvůli plánovanému příchodu nového simulátoru od společnosti Microsoft, orientovaného na podnikatelskou sféru [31]. Tyto skutečnosti by mohly vést k tomu, že by při vývoji systému pro přistávání navrženého v této práci nebylo možné řešit některé problémy, jež by vyvstaly, kvůli chybějící komunitě, nedostatku informačních zdrojů a přerušené oficiální podpoře.

7 Návrh systému pro simulaci přistávání UAV na plošině

Jedním z cílů této práce bylo navrhnut systém, pomocí kterého by bylo možné simulovat přistání bezpilotního letounu na plošině s využitím některé metody z kapitoly 5. Jeho návrh je obsahem této kapitoly. Byla zvolena částečně modulární architektura, díky čemuž je možné testovat různé metody přistávání nebo i jiné úlohy. Podkapitoly 7.1 až 7.4 nahlíží na tento systém z různých hledisek. Nejprve jsou popsány jednotlivé součásti a jejich funkce, poté vnější pohled na systém a jeho rozhraní a nakonec jsou charakterizovány vazby uvnitř systému.

7.1 Komponenty systému

Systém využívá vlastní komponenty i komponenty od jiných autorů, které jsou dále popsány v této podkapitole včetně důvodů, proč byly zvoleny.

7.1.1 Letový řídicí software

Letový řídicí software běží při jeho použití v realitě přímo na počítači, který je umístěný na palubě dronu a k němuž jsou obvykle připojeny všechny senzory a aktuátory, jimž je letoun vybaven. Využití přímo tohoto softwaru při simulaci zmenšuje rozdíly mezi simulačním prostředím a realitou, což umožňuje snadnější přenos systému do skutečnosti při jeho případném reálném nasazení.

Senzory a aktuátory jsou při simulaci pouze napodobené simulátorem a software běží buď stejně jako simulátor v použitém osobním počítači, nebo, je-li to podporováno, může být spuštěn na skutečném palubním počítači UAV, se zavedenými umělými vstupy ze simulace (tzv. Simulation in Hardware - simulace na hardwaru (SIH)). Software zpracovává data ze senzorů a vstupy od uživatele, generuje akční zásahy, které předává aktuátorům a může poskytovat telemetrické služby nebo služby vysokoúrovňového dálkového ovládání např. prostřednictvím protokolu MAVLink.

Právě takové služby se využívají v navrhovaném systému a z toho důvodu byl z výběru uvedeného v kapitole 6 vyřazen řídicí software BetaFlight, který je možné řídit jen modelářským rádiovým ovladačem. Ze zbývajících dvou variant byl v návrhu dále uvažován kontrolér PX4, který je zaměřený více na profesionální a akademické využití a také nabízí širší možnosti podporovaných simulátorů než ArduPilot.

7.1.2 Simulátor

Simulátor je v navrhovaném systému počítačový program, který napodobuje realitu, letovému řídicímu softwaru poskytuje umělá měření ze senzorů v simulovaném prostředí, přijímá od něj výstupy pro aktuátory a modeluje jejich vliv na simulované vozidlo. Dále

simuluje dynamické chování celého letadla i ostatních předmětů umístěných v napodobeném světě a, podporuje-li to, také jejich vzájemné interakce. Také zprostředkovává vizuální pohled do této umělé reality prostřednictvím simulované kamery.

Z možností podporovaných kontrolérem PX4 bylo vybráno Gazebo, které alespoň částečně podporuje všechny aspekty, které byly při návrhu považovány za důležité, je rozšířitelné, má bohatou komunitu a je pro podobné úlohy běžně používáno. Ostatním simulátorům některé vlastnosti chyběly, měly jiné zaměření, skončila jim podpora nebo nebyly příliš rozšířené. I přes to by dalším dobrým kandidátem byl AirSim, který má bohatší API než Gazebo s dobrou dokumentací, takže by absence velké komunity nemusela působit problémy. Má také realističtější renderování.

7.1.3 Grafické uživatelské rozhraní (GUI)

Grafické uživatelské rozhraní (GUI) je vlastní komponentou navrhovaného systému a vytváří jednotné prostředí pro ovládání a nahlížení stavu ostatních komponent uživatelem. Tím tvoří rozhraní mezi uživatelem a systémem. Pro jeho konstrukci byl zvolen framework Qt s bindingy do jazyka Python PyQt [42], který kromě grafického rozhraní propojuje i některé komponenty v systému. O funkcích GUI podrobně pojednává kapitola 8.

7.1.4 Hodiny ze simulátoru

Některé kroky přistávacích metod je nutné časově opozdit. Simulační čas se může lišit od reálného, proto je jednou z komponent systému také přijímač hodinového signálu ze simulátoru, který ostatním komponentám může poskytovat časovač. Ten se aktivuje po uplynutí stanovené doby v simulačním čase a může danou komponentu asynchronně informovat o této události. Čas ze simulátoru Gazebo se přenáší pomocí zpráv prostřednictvím knihovny gz-transport a Python bindingů gz-python [30].

7.1.5 Kamera ze simulátoru

Pro odhad polohy plošiny vzhledem k letounu se používá obraz prostředí s fiduciárním markerem, jejž je nutné získat ze simulátoru. To se provádí stejným způsobem jako u hodin, tedy pomocí gz-python, jen s využitím zprávy jiného typu. Kamera je v simulátoru Gazebo reprezentována pluginem, který obrazová data zprostředkovává jako zprávy, a její definice je součástí modelu letounu. Před použitím je nutné kameru kalibrovat, což bylo provedeno zachycením několika desítek obrázků čtvercové plochy rozdělené na černé a bílé čtverce o známých rozměrech uspořádaných do šachovnice z různých úhlů. Později se ukázalo, že informace o kalibraci je možné získat přímo z pluginu tím, že se definuje téma zpráv, do kterého se vysílá tato i další metadata.

Modul navrhovaného systému, který přijímá obrazová data ze simulátoru zároveň provádí i jejich zpracování pomocí detektoru fiduciárních markerů, který je také komponentou tohoto systému a je popsán dále. Metodám přistávání tento modul zprostředkovává detektorem vypočtenou relativní vzdálenost v jednotlivých osách, rotaci ve svislé ose a obraz s označenými detekovanými značkami.

7.1.6 Detektor fiduciárních markerů

V obrazu ze simulátoru se detekují fiduciární značky a odhaduje se podle nich vzájemná poloha plošiny a UAV, jehož součástí je kamera. Tyto dílčí úlohy zajišťuje právě detektor fiduciárních markerů (kapitola 4). Pro použití v navrhovaném systému byly vybrány značky AprilTag 3, protože umožňují konstrukci rekurzivních značek a tím i detekci ve

větším rozsahu vzdáleností. K jejich rozpoznávání se využívá knihovna AprilTag 3 [21] s Python bindingy dt-apriltag [22].

7.1.7 Metoda přistávání

Metodou přistávání se rozumí ta komponenta navrhovaného systému, která kombinuje informace z různých jiných součástí (např. senzory z řídicího SW, polohová data z kamery atp.) za účelem ovládání dronu dle postupu popsaného v kapitole 5. Jedná se tedy o implementaci metody přistávání. UAV je ovládáno pomocí protokolu MAVLink, konkrétně prostřednictvím jeho implemetace v mavSDK pro Python [17].

7.1.8 Mise a jejich seznamy

Uživatel může pomocí GUI nebo v YAML souboru definovat vybrané podmínky prostředí a další vstupy systému (podkapitola 7.2) pro jeden průběh zvoleného přistávacího algoritmu. Takové definici se říká mise a je možné jich určit více. Všechny definice se ukládají v souboru, takže mezi běhy systému mohou být uchovány. Mise je navíc možné zahrnovat do jejich seznamů, jež se nazývají experimenty, a spouštět je postupně automaticky s vybraným počtem opakování pro každou položku zvlášť. Uchování definovaných experimentů je řešeno stejným způsobem jako u misí, tedy uložením do souboru. Každá mise má po svém ukončení určitý výsledek, který se v případě spuštění mise jako součásti experimentu uchovává ve výstupním souboru společně s výsledky již proběhlých misí.

7.1.9 Model světa

Umělé prostředí je definováno modelem světa ve formátu SDF [26], který je využíván při simulaci. Obsahuje plochu s leteckým snímkem Borského parku v Plzni, definici světelních podmínek, vlastnosti větru a model přistávací plošiny a její stínění. Pro každou misi se podle vstupů dodaných uživatelem generuje odpovídající soubor, který je pak předán simulátoru.

7.2 Vstupy systému

Tato podkapitola popisuje, jaké vstupy systém očekává a jakým způsobem jsou využity v komponentách ve smyslu informací, které uživatel systému poskytuje v průběhu jeho používání nebo předem.

7.2.1 Počáteční poloha UAV a plošiny

Do simulovaného světa se před začátkem simulace umisťuje model letounu a plošiny, jejichž poloha v rovině země vzhledem k počátku simulovaného světa (jeho geografické souřadnice jsou dány v konfiguračním souboru a odpovídají středu leteckého snímku použitého jako podkladu) je dána dvěma souřadnicemi v metrech (v osách x a y) a úhlem rotace ve stupních kolem svislé osy z . Těchto 6 vstupů (3 pro UAV a 3 pro plošinu) se zadává při definici mise v GUI a jsou také součástí uložené mise v souboru (sekce 7.2.5). Kvůli omezení simulátoru a způsobu jeho spuštění neexistuje přímočarý způsob, jak nastavit rotaci modelu dronu před spuštěním simulace, proto se rotoce nastavuje až za letu stejným způsobem, jakým se zadávají příkazy ke změně polohy prostřednictvím řídicího SW UAV.

7.2.2 Nastavení zastínění plošiny

Jedním ze simulovaných vnějších vlivů je zastínění plošiny, které je dáno 2 vstupy - podílem zastíněné a nezastíněné délky úsečky vedené kolmo na stranu plošiny, která míří ke zdroji světla, jejím středem a sklonem stínu ve stupních. Oba parametry jsou součástí definice mise (soubor nebo GUI). Stínění je napodobováno umístěním tenkého kvádru o rozměrech 30 x 5 m (šířka x výška) vysoko nad povrch na takové místo, aby vznikl požadovaný stín. Gazebo nesimuluje rozptyl světla postupně procházejícího atmosférou, takže ostrost stínu je závislá na přednastaveném konstantním rozptylu světla, ale ne na vzdálenosti vrhajícího objektu od vrženého stínu.

7.2.3 Nastavení větru

Vítr je simulován pomocí pluginu WindEffects, jehož parametry jsou nastaveny na základě 4 vstupních hodnot zadávaných v GUI nebo definici mise v souboru. Nastavuje se vodorovná rychlosť a její směrodatná odchylka v metrech za sekundu a úhel směru větru vzhledem k ose x a jeho směrodatná odchylka ve stupních.

7.2.4 Výběr přistávací metody

Metody přistávání jsou předdefinované včetně jejich parametrů (např. zisky složek PID regulátoru) a jsou identifikovány názvem. Při definici mise v GUI si uživatel vybere jednu z metod z rozbalovací nabídky, v souboru s definicemi misí je uložen její název a při spuštění mise se pak používá implementace této vybrané metody pro řízení letadla.

7.2.5 Uložené mise a jejich seznamy

Po spuštění systému se dříve definované a uložené mise a jejich seznamy (tzv. experimenty) načtou ze souborů `mise.yaml` a `experimenty.yaml`, které obsahují seznam definic misí resp. experimentů. Definice experimentu je seznam názvů misí a počtu jejich opakování v rámci něj.

7.2.6 Konfigurační soubor

Konfigurační soubor obsahuje mnoho dalších vstupů, které jsou potřebné pro správný běh systému. Jedná se mimo jiné o nastavení některých cest k souborům a složkám (např. složka pro uchovávání definic umělých světů simulátoru), výchozí kalibrace kamery, která se může použít dokud není doručena kalibrace přímo ze simulátoru, model UAV, který se má použít pro simulaci, kovarianční matice a matice modelu letounu pro Kálmánův filtr, směr dopadajícího slunečního záření nebo geografické souřadnice počátku kartézského souřadného systému umělého světa.

7.3 Výstupy systému

Předmětem této podkapitoly je, jaké informace tvoří výstup systému směrem k jeho uživateli a jejich původ uvnitř systému. Výstupy zahrnují různé informace grafického rázu (3D vizualizace simulátoru a pohled kamery), textové informace (stavy a výsledky) i numerická data (setpointy, chyby atp.)

7.3.1 3D vizualizace simulovaného prostředí

Simulátor Gazebo má volitelně grafický režim, jehož součástí je 3D vizualizace simulace, ve které je možné se volně pohybovat a sledovat, co se ve světě odehrává.

7.3.2 Pohled kamery s vyznačenými detekovanými markery

Ve vlastním GUI se v záložce „Živě“ zobrazuje pohled z kamery, který je využívaný řídicím algoritmem přistávání. V obrazu jsou červeným čtyřúhelníkem vyznačeny všechny AprilTagy, které detektor zachytíl, a jejich souřadnice a rotace v souřadné soustavě kamery.

7.3.3 Stav mise a experimentu

V záložce „Živě“ uživatelského rozhraní se také vyskytuje textový souhrn stavu probíhajícího experimentu a mise. Zobrazuje se aktuální opakování dané mise a její pořadí v probíhajícím experimentu. Přistávací metody jsou členěny do několika stavů (obrázek 5.1), z nichž má každý název, který se na téže záložce také zobrazuje. Kromě toho se během přistávání zobrazují grafy skutečné chyby polohy a rotace UAV vzhledem k požadované trajektorii.

7.3.4 Výsledek mise a experimentu

Po ukončení mise se její výsledek, který zahrnuje různé sledované metriky, vypisuje do konzole a v případě, že mise byla spuštěna jako součást experimentu se navíc výsledek přidá do souboru s výsledky všech doposud proběhlých misí experimentu. Tento soubor je pak možné po ukončení analyzovat a vyvodit souhrnné výsledky. Sledované metriky zahrnují počet kroků algoritmu, počet kroků, ve kterých nebyl detekován tag, skutečné místo přistání, definice mise, skutečné polohy a časy přechodů mezi stavy metody, MAE na konci přistávání a průměrná doba jednoho kroku algoritmu.

7.3.5 Uložené mise a jejich seznamy

Při definici mise nebo experimentu v GUI se na disk uloží soubor, který zahrnuje všechny definice misí, respektive experimentů, aby mohly být při dalším spuštění systému načteny.

7.4 Struktura systému

Tato podkapitola se zabývá uspořádáním komponent v navrženém systému a charakteristikou jejich vzájemných informačních vazeb a vazeb s okolím systému. Součásti budou kategorizovány podle různých hledisek, stejně tak i vazby, a bude podán pohled na strukturu systému s ohledem na jeho použití. Obrázek 7.1 obsahuje diagram struktury systému s komponentami, vstupy a výstupy a některými vazbami mezi nimi. Pro udržení rozumné míry přehlednosti byla část na této úrovni abstrakce méně významných vazeb vypuštěna i za cenu snížené informační hodnoty.

Systém je možné rozdělit do dvou pomyslných částí z nichž každá zastává převážně určitou funkci. Jednak se jedná o část přípravnou a dozorovací, jednak o část simulační. Některé v systému použité komponenty byly vytvořeny autorem této práce, ty v diagramu reprezentuje kruh s vepsaným názvem nebo zkratkou komponenty, další jsou od jiných autorů a jsou vyznačené obdélníkem. Vazby se značí orientovanými šipkami, které míří ve směru toku informace, od výstupu ke vstupu, jsou-li zúčastněny komponenty ve vztahu vstup-výstup i výstup-vstup v závislosti na situaci pro jeden druh informace, je šipka oboustranná. Šedé šipky označují vazby se soubory, černé jiné vazby, které jsou obvykle popsané podle druhu informace, jejíž tok reprezentují.

Přípravná část systému zahrnuje GUI, experimenty (v obrázku 7.1 zkráceno na Exp., u dalších komponent budou zkratky uvedeny v závorce samostatně), mise a definiční

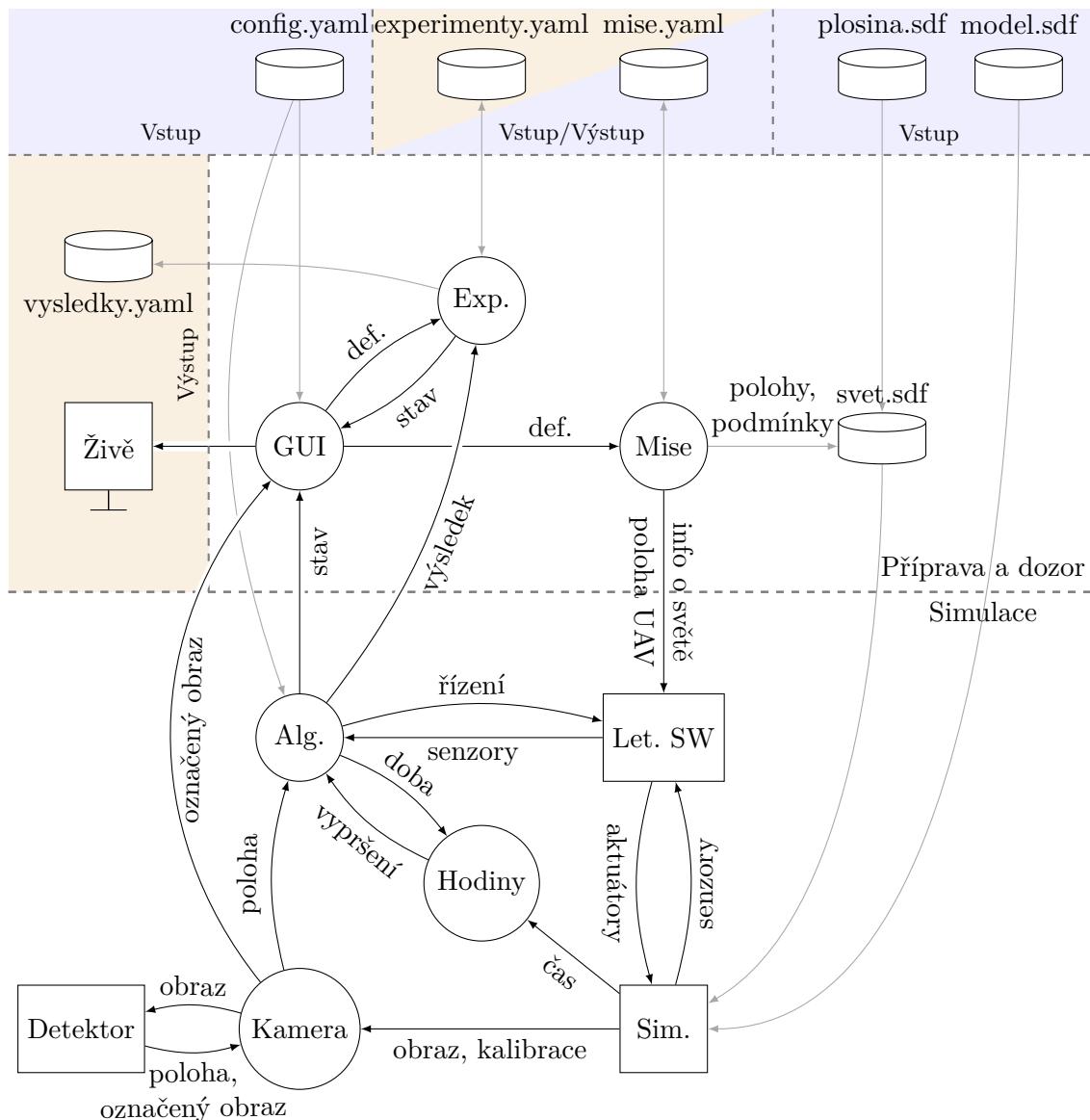
soubor prostředí (svet.sdf). S vnějším prostředím systému se vážou všechny tyto součásti. Experimenty a mise se načítají z příslušných souborů a ukládají do nich (experimenty.yaml a mise.yaml, jedná se o vstupy i výstupy zároveň), experimenty navíc po dokončení každé z dílčích misí uloží svůj výsledek do souboru vysledky.yaml. GUI přijímá jako vnější vstup od uživatele definice misí a experimentů a požadavky na ovládání systému (spouštění a zastavení simulace, ukládání misí a experimentů), dalším vstupem je pak konfigurační soubor (config.yaml), pomocí kterého se nastavují některé aspekty chování a zobrazení GUI. Jeho výstupem je naopak poskytnutí informací uživateli o již definovaných a právě probíhajících experimentech a misích. Soubor svet.sdf obsahuje direktivu pro zahrnutí modelu přistávací plošiny, která se tak stává jeho vnějším vstupem.

Budeme-li uvažovat vazby uvnitř systému v jeho přípravné a dozorovací části, u GUI se jedná o předávání definice experimentu a mise odpovídajícím komponentám po jejich zadání uživatelem a získávání stavu o běžícím experimentu. Mise upravuje definiční soubor světa (svet.sdf) o podmínky simulace, polohu plošiny a polohu stínění, ten si pak při svém startu načítá simulátor.

Komponentami simulační části systému jsou přistávací algoritmus (Alg.), letový řídicí software (Let. SW), hodiny ze simulátoru, detektor fiduciárních markerů, kamera ze simulátoru a simulátor. Mezi vazby s vnějším prostředím patří vstup přistávacího algoritmu, kterým je konfigurační soubor, stejný vstup má i kamera (tato vazba je v obrázku 7.1 skrytá). Dalším vstupem je soubor model.sdf, který obsahuje definici simulovaného letounu včetně použitých modelů senzorů a aktuátorů, jež jsou implementovány jako pluggy simulátoru.

V rámci simulační části systému se uplatňuje několik dále uvedených vazeb. Přistávací algoritmus využívá služby letového řídicího SW a tím mu zadává příkazy k řízení a zpět naopak dostává aktualizace hodnot naměřených senzory UAV. Během simulace algoritmus může vyžadovat odměření určité doby, v takovém případě předá hodinám ze simulátoru tuto dobu a po jejím uplynutí je zpět informován o tomto jevu. Dále z kamery pravidelně dostává odhadovanou polohu plošiny vůči kameře. Řídicí SW letounu komunikuje se simulátorem přímo prostřednictvím jeho API, dodává mu řízení simulovaných aktuátorů a zpět dostává umělá měření senzorů. Vstupem hodin je čas poskytovaný simulátorem, podobně kamera z něj získává obraz a kalibrační data. Detektor tento obraz dostává od kamery a jako výstup zpět vrací obraz s označenými detekovanými markery a polohu plošiny vůči kameře.

Mezi přípravnou a dozorovací částí a částí simulační se uplatňují další vazby. Přistávací algoritmus o svém aktuálním stavu informuje GUI a po dokončení manévru předá výsledek mise experimentu. Letový řídicí SW přijímá informace o simulačním prostředí (geografické souřadnice počátku) a požadovanou polohu UAV v něm. Simulátor načítá definiční soubor umělého světa, který je vytvořen v přípravné části. Tyto vazby tvoří uzavřenou smyčku, kdy uživatel se systémem komunikuje prostřednictvím GUI, jež komunikuje s ostatními částmi systému a připraví simulaci, simulační část ji provede a vrátí výsledky, které jsou prostřednictvím výstupů prezentovány uživateli a ten může na jejich základě provést další akce.



OBRÁZEK 7.1: Struktura navrhovaného systému pro simulaci přistávání bezpilotního letounu (UAV), jeho komponenty a jejich vzájemné informační vazby, vstupy a výstupy. Kruhové prvky jsou vlastní, obdélníkové od jiných autorů a nízké válce značí soubory. V oranžových polích jsou výstupy, v modrých vstupy. GUI znamená grafické uživatelské rozhraní, Exp. je experiment, Alg. je přistávací algoritmus, Let. SW je letový řídicí software UAV a Sim. označuje simulátor.

8 Grafické uživatelské rozhraní

Systém pro simulaci přistávání UAV navržený v kapitole 7 obsahuje kromě komponent důležitých pro samotnou simulaci také grafické uživatelské rozhraní (GUI), které slouží k oboustranné komunikaci s uživatelem před simulací a během ní. Je rozčleněno podle 3 hlavních způsobů užití do 3 obrazovek:

1. Mise (podkapitola 8.1), sloužící k přípravě misí, správě uložených misí a jejich spouštění.
2. Živě (podkapitola 8.2), která zobrazuje aktuální data v průběhu simulace.
3. Experimenty (podkapitola 8.3), jež slouží ke sdružování misí do množin, správě těchto množin a hromadnému spouštění misí z nich.

Mezi obrazovkami se přepíná v záhlaví nebo pomocí kombinace kláves + . GUI je implementováno v programovacím jazyce Python pomocí bindingů frameworku Qt do Pythonu zvaného PyQt [42].

8.1 Obrazovka Mise

Mise je první ze 3 obrazovek GUI navrhovaného systému (kapitola 7), pomocí níž je možné definovat mise pro simulaci, zobrazovat je na mapě, ukládat a spouštět. K jednotlivým úkonům slouží různé její části (v obrázku 8.1 označené ① - ⑨), které budou dále popsány. Mezi ovládacími prvky se lze pohybovat pomocí klávesy tabulátoru.

① **Výběr obrazovky.** Jak již bylo zmíněno, celé GUI je členěno do 3 obrazovek. V levém horním rohu každé z nich jsou po celou dobu běhu programu umístěny 3 záložky, které reprezentují jednotlivé obrazovky (Mise, Živě, Experimenty). Záložka právě nahlížené obrazovky je zvýrazněna světlejším provedením a je graficky spojena se spodní částí rozhraní. V podkapitolách o dalších obrazovkách (sekce 8.2 a 8.3) již tento výběr není zmiňován, protože je společný.

② **Mapa** vizualizuje polohu plošiny nastavenou ve střední části obrazovky ③ pomocí jejího obrázku s červenou hranou a polohu letounu (zadanou v části ④) pomocí bílého kříže s kruhy na všech ramenou, který tak připomíná kvadrokoptéru, na leteckém snímku použitém v simulátoru jako podklad.

③ **Umístění plošiny** umožňuje nastavit souřadnice X a Y středu plošiny vzhledem k počátku simulovaného světa, který se nachází uprostřed mapy, a úhel natočení ϕ vůči ose x . Změna polohy aktualizuje vizualizaci v mapě ②.

④ **Umístění letounu** poskytuje stejná nastavení (X, Y, ϕ) pro letoun jako část ③ pro plošinu. Nastavené hodnoty reflektuje ikona dronu v mapě ②).



OBRÁZEK 8.1: Obrazovka „Mise“ grafického uživatelského rozhraní navrhovaného systému pro simulaci přistávání bezpilotního letadla. Slouží k definici vlastností simulace, ukládání definic a spouštění simulace s danými podmínkami.



OBRÁZEK 8.2: Příklad simulace přistávání bezpilotního letounu na plošinu s fiduciálním markerem, která je částečně zastíněná.

⑤ **Nastavení větru** zahrnuje 4 proměnné, které ovlivňují působení větru na model v simulaci. Jedná se o vodorovnou rychlosť a její směrodatnou odchylku (v a σ_v) a vodorovný úhel vzhledem k ose x a jeho směrodatnou odchylku (ϕ a σ_ϕ). Tyto hodnoty se pak využívají při generování definičního souboru světa pro nastavení vlastností pluginu, který generuje okamžité vlastnosti větru v každém kroku simulace.

⑥ **Vlastnosti plošiny** dává možnosti nastavení dalších aspektů simulované plošiny, tedy její velikosti (možnost strana), podílu zastíněné části úsečky vedené středem plošiny ve směru osy y a sklonu stínu. Obrázek 8.2 ukazuje příklad simulace přistávání letounu na částečně zastíněnou plošinu.

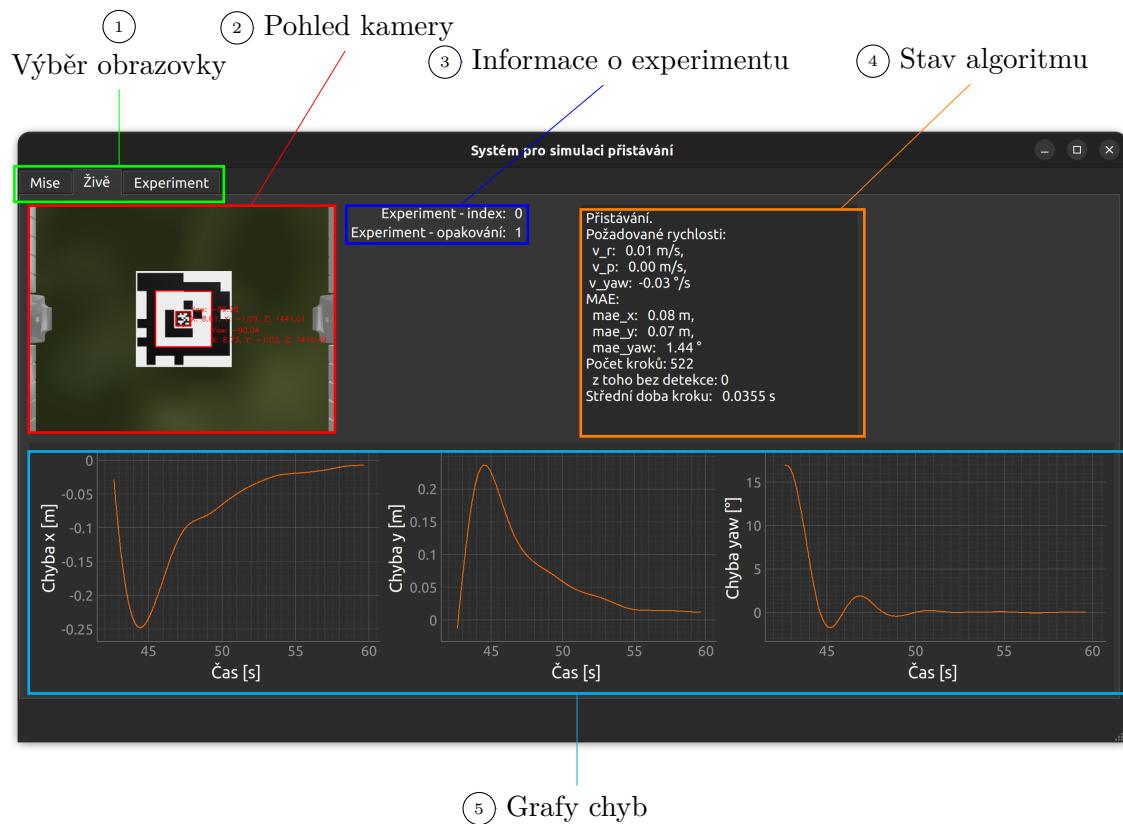
⑦ **Výběr metody přistávání** umožňuje uživateli z nabídky vybrat algoritmus, který se použije k řízení UAV během simulace. Metody v nabídce jsou předdefinované a uživatelsky je nelze měnit. Vlastnosti některých metod je možné upravit změnou konfiguračního souboru.

⑧ **Práce s misí** je část obrazovky, pomocí níž se nastavuje název mise, definice jím identifikovaná se dá uložit do seznamu, načíst z něj (což přepíše aktuální hodnoty v částech ② - ⑦) nebo odstranit. Aktuálně nastavené hodnoty v předchozích částech pak mohou být využity pro spuštění simulace, kdy se název mise zároveň použije jako název světa a nedodá-li ho uživatel, je použita výchozí hodnota „BezNazvu“.

⑨ **Uložené mise** zobrazuje seznam názvů dříve uložených definic misí. Ty se uchovávají v souboru načítaném při spuštění programu. Spravovat je lze pomocí ovládacích prvků v části ⑧.

8.2 Obrazovka Živě

Druhou ze 3 obrazovek GUI navrhovaného systému (kapitola 7) je obrazovka Živě, která uživateli zprostředkovává informace o právě probíhající simulaci. Jedná se o pohled z kamery, informace o experimentu (je-li nějaký spuštěn), výpis stavu poskytovaný algoritmem a grafy chyb v poloze a natočení vůči plošině. Snímek této obrazovky je na obrázku 8.3.



OBRÁZEK 8.3: Obrazovka „Živě“ grafického uživatelského rozhraní navrhovaného systému pro simulaci přistávání bezpilotního letadla. Slouží k dohledu uživatele na simulaci.

② **Pohled kamery** se nachází v levém horním rohu obrazovky a jedná se o obraz získávaný z kamery, která je svázaná se simulovaným modelem. Pokud je v jejím zorném poli detekován marker, je označen červeným rámem a jsou k němu připojeny informace o jeho poloze vůči dronu.

③ **Informace o experimentu** zahrnují index probíhající mise a počet již proběhlých opakování této mise. Index odpovídá pořadí v tabulce na obrazovce Experimenty. Pokud žádný experiment neprobíhá, zobrazují se místo těchto informací pomlčky v příslušných polích.

④ **Stav algoritmu** je text v přirozeném jazyce, který poskytuje algoritmus jako shrnutí svého stavu v daném okamžiku. Pro implementované metody se během přistávání jedná o požadované rychlosti vodorovně vpravo, vpřed a kolem svislé osy dronu, MAE v souřadnicích x , y a rotaci kolem svislé osy, počtu již proběhlých kroků algoritmu a střední doby kroku.

⑤ **Grafy chyb.** Spodní polovinu obrazovky zaujmají 3 grafy, ve kterých je vykreslen průběh chyby v souřadnicích x a y a rotaci kolem svislé osy v závislosti na čase. Chybou se rozumí rozdíl skutečné a požadovanou trajektorií. Grafy jsou interaktivní, je možné přiblížení a oddálení pomocí kolečka myši a pohyb tahem kurzoru, kdy osa x všech grafů je svázána. Vrátit se do původního zobrazení je možné pomocí tlačítka, které se zobrazí v levém dolním rohu grafu při změně.

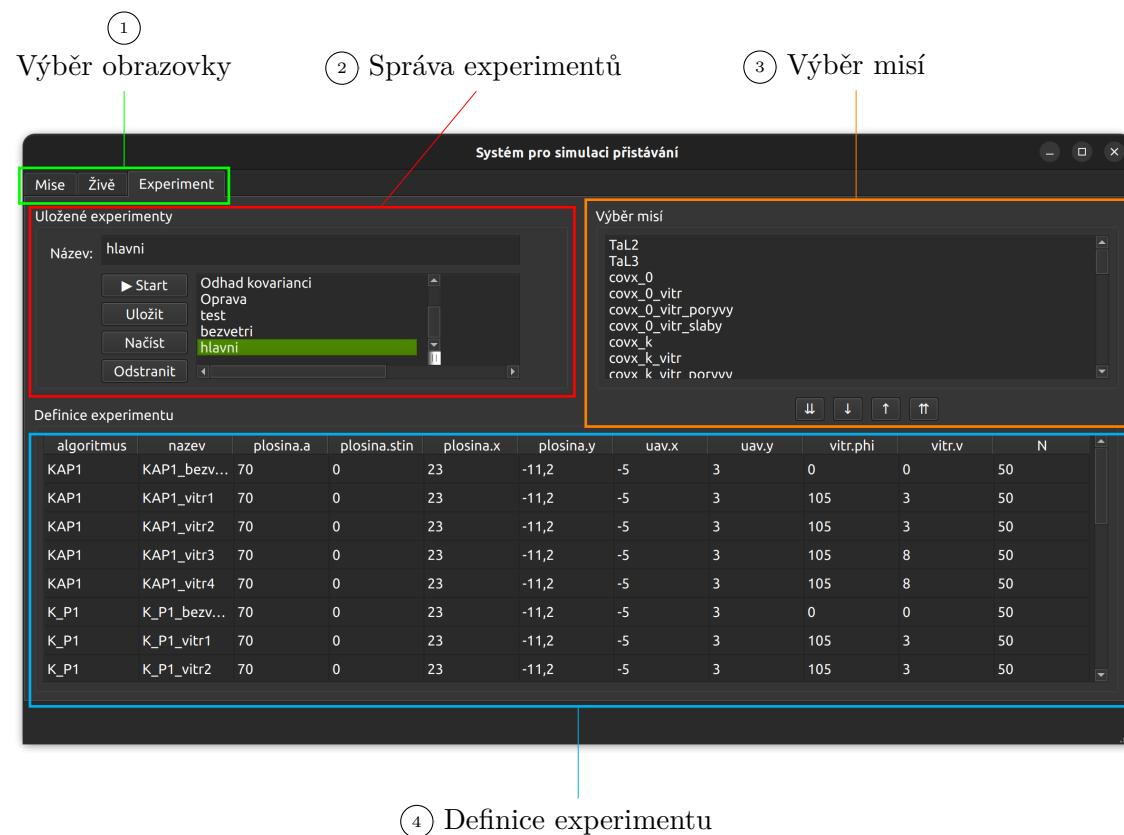
8.3 Obrazovka Experimenty

Poslední obrazovka GUI se nazývá Experimenty a funkčně se podobá obrazovce Mise. Rozdílem je, že definice experimentu se neskládá z parametrů simulace jako u mise, ale jedná se o seznam misí. S těmito seznamy už se pak pracuje podobně jako s misemi, tedy se ukládají, načítají, mažou a spouští. Dále je uveden popis částí této obrazovky.

② **Správa experimentů** má 3 součásti, jednak lze nastavit název aktuálního experimentu, dále je možné aktuální experiment uložit s daným jménem, načíst ho, nebo odstranit, je-li již uložen, a spustit. Poslední součástí je seznam již uložených experimentů. Jednoduché klepnutí v seznamu změní aktuální název, dvojitým klepnutím se vybraný experiment rovnou načte.

③ **Výběr misí** se využívá při definování nového experimentu tak, že se označí mise, kterou nebo které si přejeme přidat do experimentu a použijeme tlačítko s jednou šipkou dolů (\downarrow) umístěné pod nabídkou. Tím se vybrané mise přenesou do tabulky s definicí experimentu (④) a nastaví se jim výchozí počet opakování daný v konfiguračním souboru. Všechny mise z nabídky se do experimentu přidají tlačítkem s dvojitou šipkou ($\downarrow\downarrow$). Odstranění misí z aktuálního experimentu se provádí obdobně pomocí zbylých dvou tlačítek se šipkami vzhůru (\uparrow a $\uparrow\uparrow$), jen se mise označují v tabulce s definicí experimentu (④).

④ **Definice experimentu** je zobrazená v tabulce v dolní polovině obrazovky, každý její řádek reprezentuje jednu misi, jejíž některé vlastnosti jsou vypsány ve sloupcích. Poslední sloupec je počet opakování dané mise v experimentu, je-li spuštěn, simulují se postupně všechny mise v pořadí, v jakém jsou uvedeny v tabulce, a to vždy s uvedeným počtem opakování.



OBRÁZEK 8.4: Obrazovka „Experimenty“ grafického uživatelského rozhraní navrhovaného systému pro simulaci přistávání bezpilotního letadla určená k vytváření, správě a spouštění experimentů

9 Experimenty a vyhodnocení

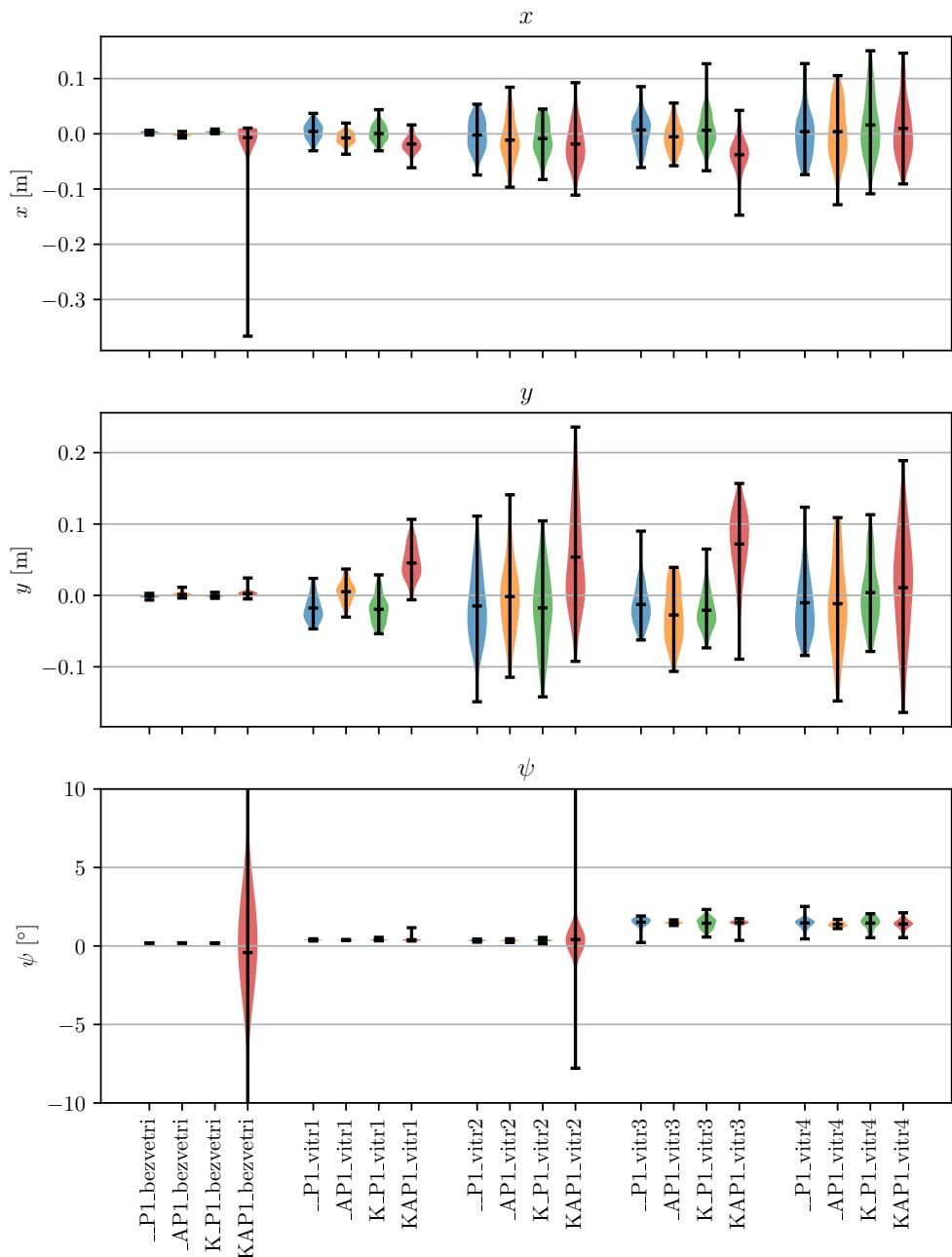
V navrženém systému byly implementovány a odsimulovány 4 vybrané metody přistávání (kapitola 5), na nichž byly provedeny experimenty popsané v této kapitole, které sloužily primárně ke zjištění rozdílů mezi implementovanými algoritmy podle různých hledisek. Během několikrát opakované simulace byly sledovány určité ukazatele, které byly zaznamenávány a statisticky zpracovány, konkrétní provedení je vždy uvedeno v příslušné podkapitole daného experimentu. Pro provedení experimentů sekce 9.1 až 9.3 bylo zvoleno 5 různých tříd větrných podmínek (bezvětrí, slabý vítr, slabý vítr s poryvy, čerstvý vítr, čerstvý vítr s poryvy). Nastavení simulačních parametrů v jednotlivých třídách shrnuje tabulka 9.1.

třída	v [m/s]	σ_v [m/s]	ς [°]	σ_ς [°]
bezvětrí	0	0	0	0
slabý vítr	3	1	105	10
slabý vítr s poryvy	3	3	105	20
čerstvý vítr	8	2	105	10
čerstvý vítr s poryvy	8	4	105	20

TABULKA 9.1: Třídy větrných podmínek a jejich parametry použité při simulaci v některých experimentech.

9.1 Přesnost přistání

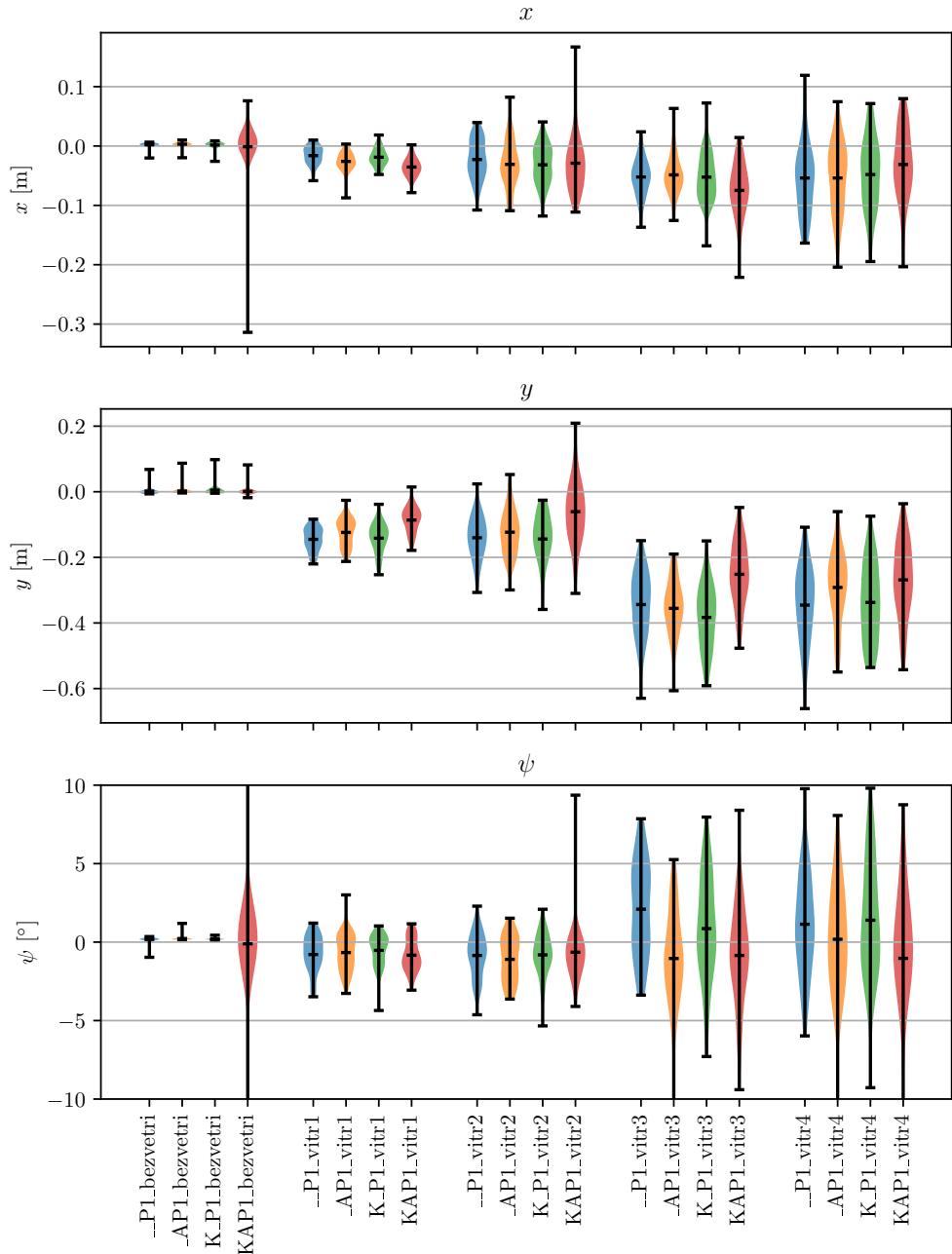
Podstatou tohoto experimentu bylo zjistit, jak přesně UAV dosedne na plošinu, použijí-li se jednotlivé algoritmy. Simulace byla provedena 50x pro každou z 20 dvojic (třída větru, metoda přistávání) a zaznamenávala se skutečná poloha letadla v simulačním prostředí v okamžiku před požadavkem na dosednutí a po uvedení do klidu. Během vyhodnocení se porovnávala tato zaznamenaná poloha s polohou plošiny dle definice dané mise. Z rozdílů v souřadnicích x , y a rotaci ψ (kolem osy z) byly pro každý z 20 případů sestrojeny grafy zobrazující průměr chyby, oba extrémy a odhad rozdělení (barevná plocha kolem linie) pro okamžik před dosednutím (obrázek 9.1) a po uvedení letadla do klidu (obrázek 9.2). Vizualizaci střední hodnoty, kovarianční elipsy a každého z přistání pro 2 příklady přímo na obrázku plošiny ukazuje obrázek 9.3.



OBRÁZEK 9.1: Chyba polohy a rotace letadla před požadavkem na dosednutí v závislosti na použité metodě přistávání a třídě větru. Metody jsou označené barevně a třídy větru jsou od sebe odděleny prostorově. Každý graf zobrazuje střední hodnotu, oba extrémy a odhad rozdělení dané chyby.

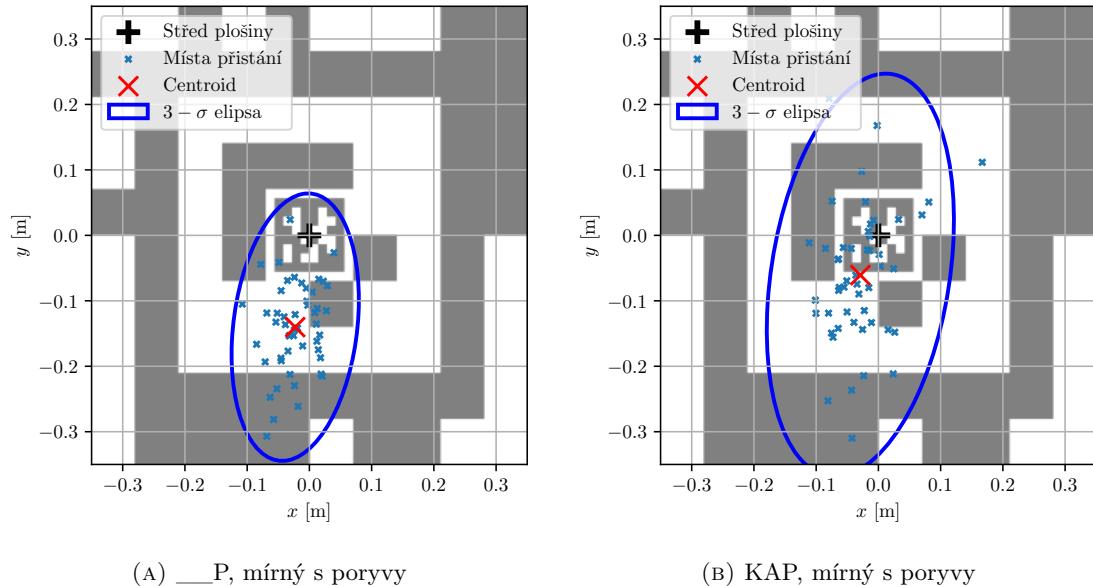
Před dosednutím je odchylka polohy od plošiny v případě všech algoritmů kromě KAP (odhad polohy s Kálmánovým filtrem přistávající po skloněné přímce) podobná a to v řádu nižších jednotek centimetrů v obou prostorových souřadnicích, kdy nejhorší je s průměrnou odchylkou $-2,74 \times 10^{-2}$ m metoda přistávající po skloněné přímce bez Kálmánova filtru v případě čerstvého větru v souřadnici y . Metoda KAP tuto mez překračuje ve 4 případech, přičemž nejhorší je za čerstvého větru s výchylkou $7,19 \times 10^{-2}$ m. Průměrná odchylka v úhlu natočení ψ je až na výjimky u metody KAP za bezvětrí a při mírném větru s poryvy nízká do přibližně 2° . Se sílícím větrem a poryvy tato odchylka roste a

mezi jednotlivými pokusy jsou větší rozdíly. V prostorových souřadnicích je patrný vliv rychlosti větru i poryvů zejména na rozdíly mezi pokusy, vliv na samotnou odchylku nelze odlišit od šumu.



OBRÁZEK 9.2: Chyba polohy a rotace letadla po jeho uvedení do klidu v závislosti na použité metodě přistávání a třídě větru. Metody jsou označené barevně a třídy větru jsou od sebe odděleny prostorově. Každý graf zobrazuje střední hodnotu, oba extrémy a odhad rozdělení dané chyby.

Odchylky po uvedení letadla do klidu jsou výrazně větší, protože tření mezi plošinou a letadlem není po dotyku dostatečné ani při nastavení vysokého koeficientu tření obou ploch v simulátoru a při doběhu motorů před deaktivací je letadlo snášeno ve směru větru po plošině. Výsledkem je zvyšující se polohová odchylka v 1 směru pro rostoucí rychlos



OBRÁZEK 9.3: Porovnání míst přistání, jejich centroidů a kovariančních elips pro dva příklady algoritmů za mírného větru.

větru patrná u všech metod přistávání. Pro úhlovou odchylku platí zvětšování rozdílů mezi jednotlivými případy přistávání, ale posun v jednom směru není příliš patrný. Výsledky všech metod jsou srovnatelné, ale metody, které využívají skloněnou trajektorii ($_\text{AP}$ a $_\text{KAP}$) mají ve většině případů průměrnou odchylku bližší 0 než ostatní metody, jeden takový případ ukazuje i obrázek 9.3.

9.2 Přesnost přistávání

Další experiment zjišťoval přesnost celého přistávacího manévrů, kdy během každé ze simulací postupně počítal střední absolutní chybu (MAE) v souřadnicích x , y a rotaci ψ v každém kroku algoritmu vůči jím požadované trajektorii (ve 2 případech přímka kolmo procházející středem plošiny, ve 2 případech skloněná přímka pod stejným úhlem jako v daném větru přistávající dron nasměrovaná proti němu). Stejně jako minulý experiment byl tento pokus proveden 50x pro každý prvek kartézského součinu metody přistávání \times třídy větru. Tabulka 9.2 shrnuje MAE na konci přistávání pro dané podmínky simulace.

Na přesnost přistávání má negativní vliv rychlosť větru i jeho nárazy, kvůli zvolenému směru jsou hlavní projevy patrné v ose y . Algoritmy bez Kálmánova filtru ($_\text{P}$ a $_\text{AP}$) si při slabém větru bez poryvů vedou lépe než ty s Kálmánovým filtrem, zatímco při silnějším větru mají menší MAE metody se skloněnou trajektorií ($_\text{AP}$ a $_\text{KAP}$), na jejichž MAE nemá zvyšující se intenzita větru tak velký vliv. V rotaci jsou vždy lepší algoritmy bez Kálmánova filtru a zdá se, že poryvy větru mají navzdory očekávání spíše pozitivní vliv na střední chybu rotace. Kromě zmíněných odlišností jsou výsledky jednotlivých algoritmů obecně srovnatelné.

9.3 Úspěšnost přistávání

Během silných poryvů se stává, že se letoun v blízkosti plošiny vychýlí natolik, že se marker na plošině dostane mimo zorné pole kamery. V takový okamžik ztrácí algoritmus možnost podle obrazu odhadovat vzájemnou polohu UAV a značky, proto čeká 50 snímků

alg.		bezvětrí	slabý	slabý s p.	čerstvý	čerstvý s p.
P	x	0,10	$8,82 \times 10^{-2}$	0,10	0,11	0,12
	y	$4,68 \times 10^{-2}$	$8,83 \times 10^{-2}$	0,11	0,17	0,17
	ψ	2,48	2,60	2,53	3,37	3,12
AP	x	0,10	$8,00 \times 10^{-2}$	$8,95 \times 10^{-2}$	$7,66 \times 10^{-2}$	0,09
	y	$4,46 \times 10^{-2}$	$4,88 \times 10^{-2}$	$7,58 \times 10^{-2}$	$7,57 \times 10^{-2}$	0,11
	ψ	2,48	2,59	2,53	3,33	3,23
K_P	x	0,10	$8,80 \times 10^{-2}$	0,10	0,11	0,12
	y	$5,23 \times 10^{-2}$	0,09	0,11	0,18	0,17
	ψ	3,72	3,75	3,65	3,72	3,66
KAP	x	0,10	$7,87 \times 10^{-2}$	0,10	0,11	0,11
	y	$5,70 \times 10^{-2}$	$7,75 \times 10^{-2}$	0,11	0,12	0,13
	ψ	4,19	3,82	4,75	4,07	4,10

TABULKA 9.2: Střední absolutní chyba sledování požadované trajektorie v závislosti na větrných podmínkách a použitém algoritmu. K v názvu algoritmu znamená, že byl použit Kálmánův filtr, A znamená přistání po skloněné přímce a P proporcionální regulátor rychlosti.

s požadavkem na fixní polohu na případné odregulování poruchy způsobené poryvem, po kterém by mohl být marker opět viditelný, a mezitím stoupá rychlosť 0,3 m/s. Neobjeví-li se během této doby, považuje se pokus o dosednutí za neúspěšný a je opakován tak, že dron podle GPS přelétává na přibližnou polohu plošiny do výšky 10 m a sleduje, jestli se objeví na snímcích z kamery marker. Po úspěšné detekci se opět pokouší přistát. Při tomto experimentu se zaznamenával počet pokusů o dosednutí a vypočetla se jeho střední hodnota přes 50 provedených pokusů u všech dvojic metod přistávání a tříd větru. Výsledky jsou v tabulce 9.3.

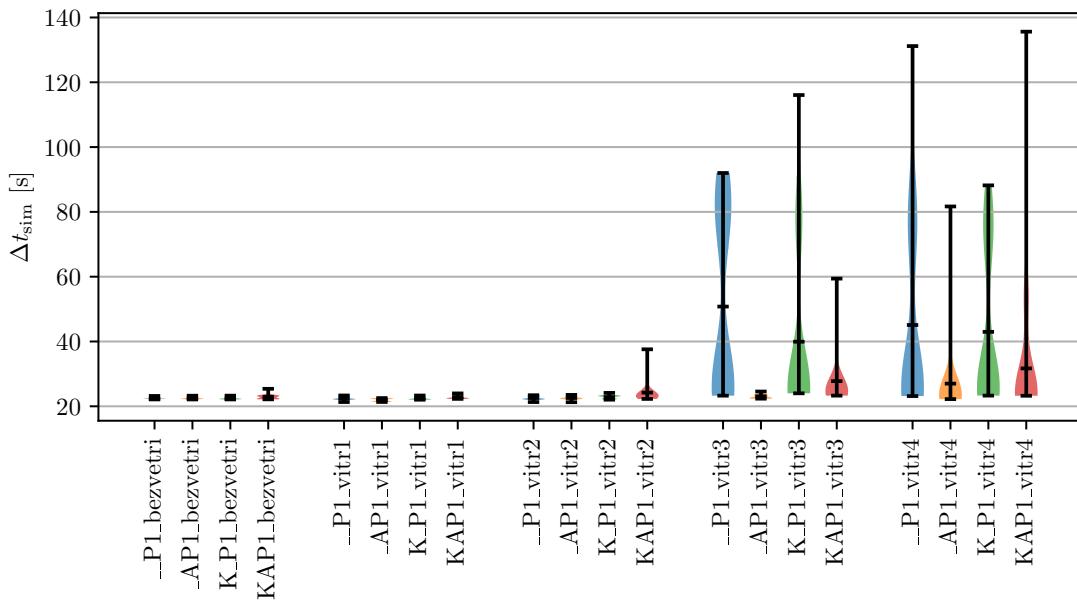
alg.	bezvětrí	slabý	slabý s p.	čerstvý	čerstvý s p.
P	1,00	1,00	1,00	3,18	2,70
AP	1,00	1,00	1,00	1,00	1,12
K_P	1,00	1,00	1,00	2,16	2,42
KAP	1,00	1,00	1,00	1,10	1,26

TABULKA 9.3: Úspěšnost přistávání vyhodnocená jako průměrný počet pokusů potřebných k úspěšnému dosednutí v závislosti na větrných podmínkách a použitém algoritmu.

Úspěšnost přistávání se liší až při čerstvém větru, kde ji vylepšuje sklonění trajektorie i Kálmánův filtr tím, že snižuje průměrný potřebný počet pokusů o dosednutí. Při použití obou strategií je průměrný počet pokusů mírně vyšší než kdyby se Kálmánův filtr nepoužil.

9.4 Doba trvání přistávání

Doba, za jak dlouho UAV přistane souvisí s úspěšností přistávání, protože při opakováném hledání plošiny se ztrácí čas, a také s přesností, jelikož rychlosť klesání roste s klesající odchylkou od požadované dráhy. Vyhodnocovaná doba se počítala od okamžiku nalezení fiduciárního markeru v obrazu do okamžiku uvedení letadla do klidu a i v tomto případě bylo provedeno stejných 50 opakování pro každou z 20 dvojic podmínek. Střední hodnota, extrémy a odhad rozdělení pro všechny možnosti přes všechna jejich opakování je v grafu na obrázku 9.4.



OBRÁZEK 9.4: Doba trvání přistávání od jeho zahájení po uvedení letadla do klidu. Pro každou z kombinací algoritmu (každý má přiřazenou barvu) a třídy větru (skupiny na vodorovné ose) je vyznačena střední hodnota doby trvání, extrémy a odhad jejího rozdělení.

Ve výsledcích je patrné snížení průměru doby přistávání u metody se skloněnou trajektorií bez Kálmánova filtru a za většího větru i u obou metod s Kálmánovým filtrem, jak bylo očekáváno dle průměrného počtu pokusů o přistání (podkapitola 9.3), pokusy jsou také patrné v rozdělení časů v grafu (obrázek 9.4). Metody s Kálmánovým filtrem za mírného větru přistávají průměrně pomaleji, za čerstvého větru a bezvětří jsou lepší než základní metoda a mají za čerstvého větru mají lepší rozdělení časů napříč pokusy, jež je koncentrováno více v nižších hodnotách.

9.5 Střední doba výpočtu v jednom kroku algoritmu

Pro posouzení výpočetní náročnosti jednotlivých metod byla v průběhu přistávání zaznamenávána také průměrná doba potřebná k výpočtu jednoho kroku algoritmu od obdržení snímku z kamery po předání řídicího požadavku letové řídicí jednotce. Krok algoritmu zahrnuje detekci fiduciárních markerů, odhad jejich polohy, přepočet polohy do vodorovné souřadné soustavy dronu, volitelně filtraci Kálmánovým filtrem, přepočet na odchylku od vybrané trajektorie a výpočet požadavku na rychlosť v jednotlivých osách letadla pomocí PID regulátorů. Průměrná (přes pokusy) střední (přes jednotlivé kroky během přistávání)

alg.	průměr [s]
P	$1,44 \times 10^{-2}$
AP	$1,48 \times 10^{-2}$
K_P	$2,47 \times 10^{-2}$
KAP	$2,95 \times 10^{-2}$

TABULKA 9.4: Průměrná střední doba jednoho kroku přistávacího algoritmu vážená počtem kroků během pokusu.

doba výpočtu jednoho kroku přistávacího algoritmu je pro dané metody uvedena v tabulce 9.4.

Se složitostí algoritmu roste i čas potřebný k provedení jednoho jeho kroku, při sklonění trajektorie se krok oproti základnímu algoritmu zpomalí o 0,4 ms, při použití Kálmánova filtru je nárůst asi 2/3 a využijí-li se obě možnosti, čas potřebný k výpočtu naroste o 15,1 ms na více než dvojnásobek.

9.6 Vliv stínu na podíl nedetekovaných markerů

10 Diskuze

Jedním z cílů této práce bylo navrhnut systém pro simulaci přistání UAV na plošině, která byla navržena pro navádění letadla pomocí vidění tak, že ji pokrýval fiduciární marker. Tento způsob navádění byl zvolen kvůli jednoduchosti, kdy nevyžaduje žádný speciální hardware, výhodou je i možnost použití v uzavřeném prostoru. Z několika možností fiduciárních markerů byl zvolen AprilTag, který svým rekurzivním usporádáním 2 tagů umožnil větší rozsah vzdáleností, ve kterých mohl být detekován.

Ústřední součástí navrženého systému je simulátor. Před jeho výběrem byla provedena rešerše, která se zabývala podporovanými funkcemi simulátorů, aby mohly být naplněny cíle práce, a letovými řídícími softwary, aby byl zjednodušen případný přenos na reálné letadlo, z níž jako kandidáti vzešly Gazebo a AirSim. První volba zvítězila kvůli jejímu tradičnímu použití v podobných úlohách a rozsáhlé komunitě, ale při vývoji systému se ukázalo, že nemá zcela vyspělou dokumentaci, a oživení některých součástí vyžadovalo mnoho neúspěšných pokusů. Spolupráce s dalšími programovými prostředky je také do jisté míry omezená, v čemž se při zpětném pohledu zdá být AirSim lepší. Kromě toho má také realističtější renderování, ale nevýhodou je, že jeho vývoj byl přerušen. Na poli letových řídicích SW připadaly v úvahu PX4 a ArduPilot, které byly oba podporovány oběma kandidátními simulátory, a byla zvolena PX4 jako profesionálnější možnost, jejíž autoři se přímo podílejí na vývoji používaných komunikačních protokolů a byla tak očekávána jejich dobrá podpora.

Byly simulovány 2 vnější vlivy, konkrétně vliv větru v 5 různých třídách rychlosti a proměnlivosti rychlosti i směru a vliv zastínění plošiny. Teplota vzduchu nebyla simulována i přes to, že může mít výrazný vliv na výkon akumulátoru dronu, protože Gazebo tento vliv neumí promítat a vlastní implementace modelu V navrženém systému by bylo možné po úpravě simulovat i další vlivy, například kouř nebo mlhu v prostředí. V praktické části byly provedeny experimenty, které zjišťovaly, jak tyto vnější vlivy působí na navržené přistávací algoritmy.

Systém pro simulaci, který byl v této práci představen a implementován, umožňuje kromě simulace přistávání řešit i další obecnější úlohy, protože je pro algoritmy definované jednotné rozhraní a je možné dodat jiný vlastní algoritmus. S uživatelem systém komunikuje pomocí grafického rozhraní, umožňujícího připravovat mise s různými parametry, spoustět je samostatně, nebo dávkově spolu s dalšími, přičemž se jejich výsledky uchovávají ve strukturovaných textových souborech. Průběh mise včetně pohledu z kamery a stavu probíhajícího algoritmu zahrnujícího například aktuální chybu vůči požadované trajektorii je možné sledovat na zvláštní obrazovce v tomto rozhraní.

Práce se dále zabývala metodami přistávání uvedenými v související literatuře včetně přistávání na pohyblivých plošinách nebo plošinách s dynamickými značkami. Pro praktickou část byly navrženy 4 příbuzné metody, které byly podrobeny experimentům za různých podmínek a bylo zaznamenáno jakou mají přesnost přistání, přesnost sledování zvolené trajektorie, úspěšnost, jak dlouho přistání trvá, jaká je jejich výpočetní náročnost a jaký vliv na detekci značek má stín, který překrývá část plošiny.

Z hlediska přesnosti přistávání mezi implementovanými metodami nebyly velké rozdíly a kvůli vlastnostem simulátoru, ve kterém se uplatňovalo příliš malé tření, byl dron i po přistání snášen dále od středu plošiny. Větší vliv na přesnost přistávání by mohlo mít ladění PID regulátoru s využitím dynamického modelu letadla a případně i modelu neurčitosti, které kvůli rozsahu nebylo provedeno. Mohlo by se jednat o důležité rozšíření pro budoucí práci.

Přestože vylepšení základního algoritmu v podobě skloněné přistávací trajektorie a Kálmánova filtru samostatně vylepšují úspěšnost přistávání i jeho rychlost a v některých případech vylepšují přesnost přistávání, jejich kombinace v jedné metodě nepřináší další zlepšení a zejména v přesnosti je často i horší. Může to být způsobeno tím, že kombinace do smyčky programu přidává další zátěž, čímž se do systému zavede významné dopravní zpoždění a požadavky pro dron kvůli němu opozdí natolik, že to má vliv na dynamiku řízení letadla, protože poruchy nejsou dostatečně rychle odregulovány.

Výhodou pokročilejších metod (zejména těch s přistáváním po skloněné přímce) bylo to, že jejich řídící strategie byly schopné i v čerstvém větru s poryvy přistát průměrně na 1,12., (bez Kálmánova filtru) resp. 1,26. (s Kálmánovým filtrem) pokus, což je méně než polovina pokusů, jež průměrně potřebovala metoda, která používala pouze proporcionalní regulátor.

11 Závěr

V této diplomové práci byl na základě rešerše simulátorů a metod pro přistávání bezpilotního vícerotorového letadla na plošině navržen a implementován systém pro simulaci, pomocí kterého se simulovaly 4 metody přistávání využívající 3 přístupy v různých vzájemných kombinacích, jež naváděly UAV podle obrazu fiduciárního markeru o 2 různě velkých souosých vrstvách umístěného na plošině za působení rozličných větrných podmínek nebo stínu zakrývajícího část plošiny.

Příliš dlouhá doba trvání jednoho kroku algoritmu může mít negativní vliv na jeho dynamické vlastnosti a tím i na přesnost přistání. Ukázalo se, že volba metody nemá příliš velký vliv na přesnost, větší vliv by mohlo mít ladění použitých regulátorů, čímž by bylo vhodné se dále zabývat. Pokročilejší metody významě zvyšují robustnost, protože umožňují přistát na menší počet pokusů i v silnějším větru, a tím je zvýšena i rychlosť přistávání, což zmírňuje rizika spojená s reálným přistáváním v mnoha oblastech. Stín má vliv pouze na úspěšnost detekce markeru, která je ale i při zastínění dostatečná k tomu, aby neovlivňovala chod přistávacího algoritmu.

Navržený systém podporuje rychlou iteraci při vývoji nových algoritmů a umožňuje efektivně testovat odlišné metody za různých vnějších podmínek a zachytávat jejich výsledky.

Bibliografie

- [1] R. E. Kalman, „A New Approach to Linear Filtering and Prediction Problems,“ *Journal of Basic Engineering*, roč. 82, č. 1, 35–45, břez. 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: <http://dx.doi.org/10.1115/1.3662552>.
- [2] R. E. Kalman a R. S. Bucy, „New Results in Linear Filtering and Prediction Theory,“ *Journal of Basic Engineering*, roč. 83, č. 1, 95–108, břez. 1961, ISSN: 0021-9223. DOI: 10.1115/1.3658902. URL: <http://dx.doi.org/10.1115/1.3658902>.
- [3] ^ CSN 31 0001, „Letectví a kosmonautika - Terminologie,“ Český normalizační institut, tech. zpr., 2005.
- [4] M. Fiala, „ARTag, a Fiducial Marker System Using Digital Techniques,“ in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, 2005. DOI: 10.1109/cvpr.2005.74. URL: <http://dx.doi.org/10.1109/CVPR.2005.74>.
- [5] B. Anton. „jMAVSim.“ (2013), URL: <https://github.com/DrTon/jMAVSim> (cit. 30.04.2024).
- [6] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas a M. Marín-Jiménez, „Automatic generation and detection of highly reliable fiducial markers under occlusion,“ *Pattern Recognition*, roč. 47, č. 6, 2280–2292, čvn. 2014, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.01.005. URL: <http://dx.doi.org/10.1016/j.patcog.2014.01.005>.
- [7] A. Gautam, P. Sujit a S. Saripalli, „A survey of autonomous landing techniques for UAVs,“ in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, s. 1210–1218. DOI: 10.1109/ICUAS.2014.6842377.
- [8] T. Kojima a T. Namerikawa, „Image-based position estimation of UAV using Kalman Filter,“ in *2015 IEEE Conference on Control Applications (CCA)*, IEEE, zář. 2015. DOI: 10.1109/cca.2015.7320663. URL: <http://dx.doi.org/10.1109/CCA.2015.7320663>.
- [9] S. Hayat, E. Yanmaz a R. Muzaffar, *Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint*, říj. 2016. DOI: 10.1109/COMST.2016.2560343.
- [10] J. Maňák. „Jak postavit RC kvadrokoptéru (drona).“ (2016), URL: <https://bastlirna.hwkitchen.cz/jak-postavit-rc-kvadrokopteru-drona/> (cit. 13.05.2024).
- [11] J. Wang a E. Olson, „AprilTag 2: Efficient and robust fiducial detection,“ in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, říj. 2016. DOI: 10.1109/iros.2016.7759617. URL: <http://dx.doi.org/10.1109/IROS.2016.7759617>.
- [12] H. Willee. „Ardupilot Documentation,“ ArduPilot. (2016), URL: <https://ardupilot.org/ardupilot/index.html> (cit. 13.05.2024).

- [13] H. Willee. „Simulation,“ ArduPilot. (2016), URL: <https://ardupilot.org/dev/docs/simulation-2.html> (cit. 29. 04. 2024).
- [14] R. Acuna a V. Willert, „Dynamic Markers: UAV landing proof of concept,“ 2017. DOI: 10.48550/ARXIV.1709.04981. URL: <https://arxiv.org/abs/1709.04981>.
- [15] A. Łebkowski, „Temperature, Overcharge and Short-Circuit Studies of Batteries used in Electric Vehicles,“ *Przeglad Elektrotechniczny*, roč. 1, květ. 2017. DOI: 10.15199/48.2017.05.13.
- [16] R. Mackay. „Using SITL with RealFlight,“ ArduPilot. (2017), URL: <https://ardupilot.org/dev/docs/sitl-with-realflight.html> (cit. 30. 04. 2024).
- [17] J. Oes, J. Vautherin, I. Sadykov a M. Riegler. „MAVSDK client for Python.“ (2017), URL: <https://github.com/mavlink/MAVSDK-Python/> (cit. 03. 05. 2024).
- [18] S. Shah, D. Dey, C. Lovett a A. Kapoor, „AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,“ in *Field and Service Robotics*, 2017. eprint: arXiv:1705.05065. URL: <https://arxiv.org/abs/1705.05065>.
- [19] R. Acuna, D. Zhang a V. Willert, „Vision-Based UAV Landing on a Moving Platform in GPS Denied Environments Using Motion Prediction,“ in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, IEEE, 2018. DOI: 10.1109/lars/sbr/wre.2018.00096. URL: <http://dx.doi.org/10.1109/LARS/SBR/WRE.2018.00096>.
- [20] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen a U. P. Schultz, „A survey of Open-Source UAV flight controllers and flight simulators,“ *Microprocessors and Microsystems*, roč. 61, 11–20, zář. 2018, ISSN: 0141-9331. DOI: 10.1016/j.micpro.2018.05.002. URL: <http://dx.doi.org/10.1016/j.micpro.2018.05.002>.
- [21] M. Krogius, A. Haggenmiller a E. Olson, „Flexible Layouts for Fiducial Tags,“ in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [22] A. Petrov. „Python bindings to the Apriltags library.“ (2019), URL: <https://github.com/duckietown/lib-dt-apriltags/> (cit. 04. 05. 2024).
- [23] A. Rodriguez-Ramos, C. Sampedro, H. Bayle, P. de la Puente a P. Campoy, „A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform,“ *Journal of Intelligent and Robotic Systems: Theory and Applications*, roč. 93, s. 351–366, 1-2 ún. 2019, ISSN: 15730409. DOI: 10.1007/s10846-018-0891-8.
- [24] K. Shabalina, A. Sagitov, L. Sabirova, H. Li a E. Magid, „ARTag, AprilTag and CALTag Fiducial Systems Comparison in a Presence of Partial Rotation: Manual and Automated Approaches,“ in *Lecture Notes in Electrical Engineering*. Springer International Publishing, dub. 2019, 536–558, ISBN: 9783030112929. DOI: 10.1007/978-3-030-11292-9_27.
- [25] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha et al., *Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges*, 2019. DOI: 10.1109/ACCESS.2019.2909530.

- [26] Open Source Robotics Foundation. „SDFormat.“ (2020), URL: <http://sdformat.org/> (cit. 04.05.2024).
- [27] H. Willee. „jMAVSIM with SITL,“ PX4. (2020), URL: https://docs.px4.io/main/en/sim_jmavsim/ (cit. 29.04.2024).
- [28] H. Willee. „Simulation,“ PX4. (2020), URL: <https://docs.px4.io/main/en/simulation/> (cit. 29.04.2024).
- [29] M. Košťák a A. Slabý, „Designing a Simple Fiducial Marker for Localization in Spatial Scenes Using Neural Networks,“ *Sensors*, roč. 21, č. 16, s. 5407, srp. 2021, ISSN: 1424-8220. DOI: 10.3390/s21165407.
- [30] R. Mainwaring. „Python bindings for gz-msgs and gz-transport.“ (2021), URL: <https://github.com/srmainwaring/gz-python/> (cit. 03.05.2024).
- [31] Microsoft Research. „Home - Airsim.“ (2021), URL: <https://microsoft.github.io/AirSim/> (cit. 01.05.2024).
- [32] M. Saavedra-Ruiz, A. M. Pinto-Vargas a V. Romero-Cano, „Monocular visual autonomous landing system for quadcopter drones using software in the loop,“ srp. 2021. URL: <http://arxiv.org/abs/2108.06616>.
- [33] E. Kakaletsis, C. Symeonidis, M. Tzelepi et al., „Computer Vision for Autonomous UAV Flight Safety: An Overview and a Vision-based Safe Landing Pipeline Example,“ *ACM Computing Surveys*, roč. 54, 9 pros. 2022, Týká se hlavně bezpečnosti, žádné konkrétní algoritmy nepředstavuje, ISSN: 15577341. DOI: 10.1145/3472288.
- [34] W. Luo, H. Ebel a P. Eberhard, „An LSTM-based approach to precise landing of a UAV on a moving platform,“ *International Journal of Mechanical System Dynamics*, roč. 2, s. 99–107, 1 břez. 2022, ISSN: 27671402. DOI: 10.1002/msd2.12036.
- [35] L. Xin, Z. Tang, W. Gai a H. Liu, *Vision-Based Autonomous Landing for the UAV: A Review*, lis. 2022. DOI: 10.3390/aerospace9110634.
- [36] M. Pecka. „Gazebo ROS Battery plugin,“ České vysoké učení technické. (2023), URL: https://github.com/ctu-vras/gazebo_ros_battery (cit. 30.04.2024).
- [37] J. V. Sickl a J. A. Dutton. „Lesson 8: Real-Time Global Positioning System Surveying.“ (2023), URL: <https://www.e-education.psu.edu/geog862/node/1845> (cit. 13.05.2024).
- [38] Betaflight. „SITL,“ Betaflight. (2024), URL: <https://betaflight.com/docs/development/sitl> (cit. 29.04.2024).
- [39] FlightGear. „Features – FlightGear Flight Simulator.“ (2024), URL: <https://www.flightgear.org/about/features/> (cit. 30.04.2024).
- [40] B. Jon. „JSBSim Open Source Flight Dynamics Model.“ (2024), URL: <https://jsbsim.sourceforge.net/> (cit. 30.04.2024).
- [41] RealFlight. „RealFlight RC Flight Simulator Software and Accessories.“ (2024), URL: <https://www.realflight.com/> (cit. 30.04.2024).
- [42] Riverbank Computing Ltd. „What is PyQt?“ (2024), URL: <https://www.riverbankcomputing.com/software/pyqt/> (cit. 03.05.2024).