



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.

12. 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

<b>Abstrakt</b>	<b>ii</b>
<b>1 Úvod</b>	<b>1</b>
1.1 Bezpilotní letadlo . . . . .	1
1.2 Přistávání . . . . .	1
1.3 Struktura práce . . . . .	1
<b>2 Definice úlohy</b>	<b>2</b>
<b>3 Plošina pro přistávání</b>	<b>3</b>
3.1 Fiduciární markery . . . . .	3
3.1.1 ARTag . . . . .	4
3.1.2 Aruco . . . . .	4
3.1.3 Apriltag . . . . .	4
<b>4 Detekce fiduciárních markerů</b>	<b>5</b>
<b>5 Metody přistávání</b>	<b>7</b>
5.1 Obecné vlastnosti metod . . . . .	7
5.2 Přistávání po svislé přímce . . . . .	8
5.3 Přistávání po svislé přímce s využitím Kálmánova filtru . . . . .	8
5.4 Přistávání po skloněné přímce . . . . .	10
5.5 Přistávání po sklonené přímce s využitím Kálmánova filtru . . . . .	10
<b>6 Simulátory bezpilotních letadel</b>	<b>12</b>
6.1 Gazebo . . . . .	13
6.2 RealFlight . . . . .	14
6.3 jMAVSim . . . . .	14
6.4 FlightGear . . . . .	14
6.5 JSBSim . . . . .	14
6.6 Airsim . . . . .	15
<b>7 Návrh systému pro simulaci přistávání UAV na plošině</b>	<b>16</b>
7.1 Komponenty systému . . . . .	16
7.1.1 Letový řídicí software . . . . .	16
7.1.2 Simulátor . . . . .	16
7.1.3 Grafické uživatelské rozhraní (GUI) . . . . .	17
7.1.4 Hodiny ze simulátoru . . . . .	17
7.1.5 Kamera ze simulátoru . . . . .	17
7.1.6 Detektor fiduciárních markerů . . . . .	17
7.1.7 Metoda přistávání . . . . .	18
7.1.8 Mise a jejich seznamy . . . . .	18

7.1.9	Model světa . . . . .	18
7.2	Vstupy systému . . . . .	18
7.2.1	Počáteční poloha UAV a plošiny . . . . .	18
7.2.2	Nastavení zastínění plošiny . . . . .	18
7.2.3	Nastavení větru . . . . .	19
7.2.4	Výběr přistávací metody . . . . .	19
7.2.5	Uložené mise a jejich seznamy . . . . .	19
7.2.6	Konfigurační soubor . . . . .	19
7.3	Výstupy systému . . . . .	19
7.3.1	3D vizualizace simulovaného prostředí . . . . .	19
7.3.2	Pohled kamery s vyznačenými detekovanými markery . . . . .	19
7.3.3	Stav mise a experimentu . . . . .	19
7.3.4	Výsledek mise a experimentu . . . . .	20
7.3.5	Uložené mise a jejich seznamy . . . . .	20
7.4	Struktura systému . . . . .	20
<b>8</b>	<b>Grafické uživatelské rozhraní</b>	<b>23</b>
8.1	Obrazovka Mise . . . . .	23
8.2	Obrazovka Živě . . . . .	25
8.3	Obrazovka Experimenty . . . . .	27
<b>9</b>	<b>Experimenty a vyhodnocení</b>	<b>29</b>
9.1	Přesnost přistání . . . . .	29
9.2	Přesnost přistávání . . . . .	29
9.3	Úspěšnost přistávání . . . . .	30
9.4	Střední doba výpočtu v jednom kroku algoritmu . . . . .	30
9.5	Vliv stínu na podíl nedetekovaných markerů . . . . .	30
<b>10</b>	<b>Diskuze</b>	<b>31</b>
10.1	Možnosti reálného nasazení systému . . . . .	31
<b>11</b>	<b>Závěr</b>	<b>32</b>
	<b>Bibliografie</b>	<b>33</b>

# Seznam obrázků

5.1	Přistávání po skloněné přímce . . . . .	11
7.1	Struktura navrhovaného systému . . . . .	22
8.1	GUI: Obrazovka „Mise“ . . . . .	24
8.2	Přistávání na zastíněnou plošinu . . . . .	25
8.3	GUI: Obrazovka „Živě“ . . . . .	26
8.4	GUI: Obrazovka „Experimenty“ . . . . .	27

# Seznam tabulek

6.1	Simulátory podporované kontrolérem letounu . . . . .	12
6.2	Vlastnosti vybraných simulátorů . . . . .	13
9.1	Třídy větrných podmínek . . . . .	29

# Seznam zkratek

**API** Application Programming Interface - rozhraní pro programování aplikací. 15, 17, 21

**GPS** Global Positioning System - globální polohový systém. 7, 30

**GUI** Graphical User Interface (grafické uživatelské rozhraní). iv, 17–21, 23, 25, 27

**HIL** Hardware in the loop. 15

**IMU** Inertial Measurement Unit - inerciální měřicí jednotka. 7, 9

**MSE** Mean Absolute Error - střední absolutní chyba. 27, 29, 30

**MSE** Mean Squared Error - střední kvadratická chyba. 5

**SIH** Simulation in Hardware - simulace na hardwaru. 16

**SW** Software. 18, 21

**UAV** Unmanned Aerial Vehicle - bezpilotní letadlo. iv, v, 1, 3, 8, 10, 11, 13–23, 25, 29, 30

**UDP** User Datagram Protocol. 14

# 1 Úvod

## 1.1 Bezpilotní letadlo

UAV - využití, aplikace, součástí mise je přistávání - potřeba jeho automatizace; druhy, historie, specifika; zúžení na čtyřrotorový pro účely práce S klesající teplotou roste vnitřní elektrický odpor akumulátoru UAV ([12]), čímž klesá jeho využitelná energie i výkon. Snížený maximální výkon může vést ke změně dynamice letu.

## 1.2 Přistávání

definice, možnosti, navádění na cíl (opticky, proč ne jiné možnosti)

## 1.3 Struktura práce

## 2 Definice úlohy

# 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 letadel s největším rozměrem nejvíše okolo 2 m, plošiny pro větší stroje by se pak označily jako helipad nebo heliport.

Všechna zmíněná místa většinou nesou nějaké vizuální označení, neboli fiduciální 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 [doplňit tloušťku] tvaru čtverce o straně 70 cm. Celou plošinu vždy pokrýval rekurzivní fiduciální marker Apriltag o 2 vrstvách. Vnější vrstva o rozměrech 10x10 měla uprostřed díru o rozměrech 2x2 (rodina TagCustom48h12), do které byl umístěn tag vnitřní vrstvy o rozměrech 8x8 (rodina Tag36h11) obklopený oddělovací čárou o šířce 1. 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, čímž mohla být negativně ovlivněna přesnost a spolehlivost její detekce, což bylo prozkoumáno v jednom z experimentů (podkapitola ??).

## 3.1 Fiduciální markery

Fiduciální 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 fiduciální marker označuje nějaký předmět, který slouží v zorném poli zobrazovacího přístroje k určení polohy nějakého zájmového bodu, jeho měřítka nebo jeho identifikaci. V robotice se podle jejich obrazu zachyceného kamery může určovat vzájemná poloha kamery a předmětu, na němž je značka připevněna. [20] Toho se využívá v této práci při odhadu polohy přistávací plošiny v prostoru. Dále budou uvedeny příklady některých používaných typů značek a popsány jejich základní vlastnosti. Pro hodnocení fiduciálních systému, čili markerů a jejich detektorů, se používají metriky jako 1. míra falešných pozitiv, 2. míra záměny značky, 3. míra falešných negativ, 4. minimální velikost značky [3], 5. doba trvání detekce [5]; [7] a další specifické vlastnosti. O postupech používaných při detekci pojednává kapitola 4.

### 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 byty z 36). Má nízkou míru falešných pozitiv i míru záměny značky zajinou. Ve srovnání se starším ARToolkit má vylepšenou identifikaci a verifikaci vzorů a snižuje tak míru záměn značek. Přesto má větší knihovnu možných vzorů. [3]

### 3.1.2 Aruco

Fiduciární systém Aruco s byty 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 míra záměn značek, a co největší počet přechodů mezi byty, čímž se minimalizuje míra falešných pozitiv. Navíc je možné překrývat malé části okraje značky a nenarušit tím detekci, nebo v aplikaci využívat

### 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. [7] 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. [14]

Do díry v tagu je možné umístit další tag a vytvořit tak rekurzivní značku, kterou bude možné detektovat ve větším intervalu vzdáleností [14], toho bylo využito v této práci jako značky na přistávací plošině.

## 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 [7] nebo Houghova transformace [16]. 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. aproximace čtyřúhelníků, 4. rychlého dekódování a 5. volitelného zpřesnění hran. Během prahování se 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 [7] se autoři chtějí vyhnout nespojitostem na hranicích buněk a tak používají extrémy z osmiokolí (po buňkách). Ve verzi 3 [14] 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 koroku detekují. Málo kontrastní body, které vzejdou z prahování se v dalším zpracování neuvažují, čímž jsou další kroky zrychleny.

Hranice se segmentují tak, že se nejprve najdou hrany v prahovaném obrazu, čili takové pixely, které sousedí s pixelem opačné barvy, které 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, 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. barvení). [7] Zrychlení ve verzi 3 je docíleno tím, že se zbytečně nesjednocují již sjednocené části (ušetří se volání sjednocení), navíc se včasné zamítají příliš malé oblasti, které by nemohly vést na dekódovatelný tag. [14]

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 přímkou 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 přímkou. Vybere se taková kombinace rohů, pro kterou je MSE nejnižší.

[7] Ve verzi 3 se řazení bodů provádí podle kvadrantu, ve kterém se bod nachází a sklonu průvodiče. Není tak nutné počítat explicitně úhel. [14]

Nalezené čtyříhelníky se vyrovnají (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. [7] Verze 3 zavádí bilineární interpolaci buněk tagu a zvyšuje ostrost za účelem zlepšení detekce malých značek. [14]

# 5 Metody přistávání

Součástí této práce je návrh systému pro simulaci přistání bezpilotního letounu na plošině, který umožňuje 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. V této kapitole jsou popsány metody, které byly implementovány, simulovány a následně porovnány mezi sebou (o průběhu simulace, porovnání a výsledcích dále pojednává kapitola 9). Všechny 4 metody mají společný základ, na kterém postupně staví s využitím dalších dílčích stavebních bloků a to samostatně nebo v jejich kombinaci.

## 5.1 Obecné vlastnosti 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 je vyobrazen na obrázku ??.

Na začátku simulace je dron umístěn na zemi a není aktivní. Aby bylo možné simulovat přistání, je s ním nejprve nutné vzletět. Toho 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 skoro<sup>1</sup> 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.
4. Okamžitá rychlosť  $v_o$  a směr větru  $\phi_o$  jsou náhodné s normálním rozdělením

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

filtrovaným dolní propustí 1. rádu s časovou konstantou  $\tau_v = 1,59$  pro rychlosť a  $\tau_\phi = 4,77$  pro směr. Parametry normální rozdělení  $v, \phi, \sigma_v$  a  $\sigma_\phi$  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  $\theta$ .
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

---

<sup>1</sup>Až na drobné nuance způsobené chybami měření palubních senzorů a náhodnými vlivy prostředí.

umístěnou na palubě UAV. Řídicí 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 přiblížování 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 posány dále, se využívá stejného způsobu ovládání dronu. Konkrétně se jedná PID regulátor rychlosti v jednotlivých osách podle vzdálenosti od požadované trajektorie a rotace  $\psi$  ve svislé ose (osa  $z$ ) podle odchylky od základního směru (SJ směr, osa  $y$ ). Jeho výstup je prostřednictvím režimu řízení Offboard zaveden do vnitřní kaskády regulátorů řídicí 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. Spočívá 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 rychlosti 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 rychlosťí 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. Nedojde-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 rychlosťi 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 zatížena 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  $\psi$  kolem osy  $z$  úhlová rychlosť  $\dot{\psi}$  v též 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 rozdelením  $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$  s kovarianční maticí  $\mathbf{Q}$ , která byla určena řádově podobně jako v [6] a dále podle obvyklých kovarianc chyb IMU. Vývoj stavu potom určuje rovnice (5.2).

Měření z uvedeného modelu jsou přímo 3 polohové souřadnice  $(x, y, z)$  a rotace  $\psi$ . I měření je navíc zatíženo náhodnou chybou modelovanou náhodnou veličinou s normálním rozdelením  $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ . 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:

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

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

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

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

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

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

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.

## 5.4 Přistávání po skloněné přímce

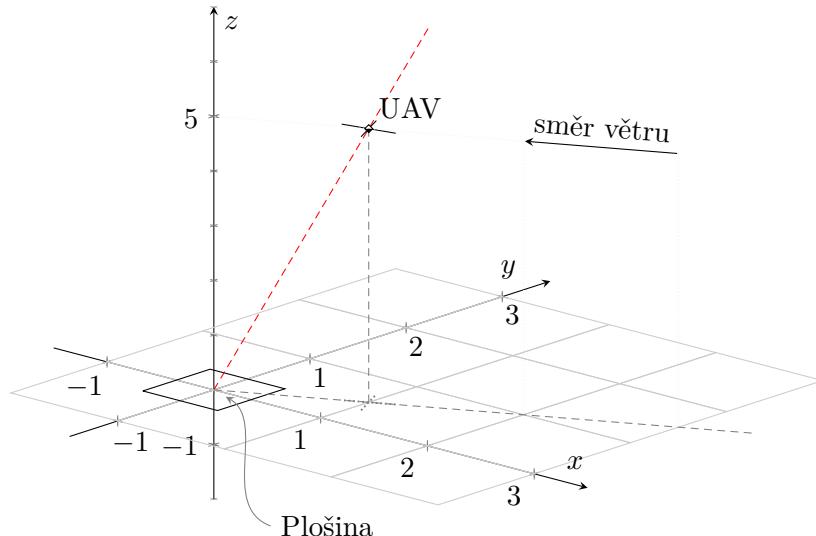
Tento způsob přistávání také vychází z prvního zmíněného (sekce 5.2), ale narozdíl od předchozího s Kálmánovým filtrem (sekce 5.3) neupravuje měření relativní polohy, ale požadovanou polohu v průběhu přistávání.

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 trajektorii 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,



OBRÁZEK 5.1: Přistávání po skloněné přímce (červeně). Letoun je naklopený proti větru, čímž ho překonává a zároveň kamerou míří na střed plošiny, která je tak v zorném poli i za přítomnosti poruch.

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 skloněnou 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, snižuje 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 tak 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 věnuje nějaká kapitola a nabízí různé podporované simulátory, které je možné spolu s nimi spustit [19], [8], [24].

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 závodění, kde je vyždována 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 [19], [8], [24].

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

Kontrolér Simulátor	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. [13]

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 [23], 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 [9], [27]. Kromě toho oficiální dokumentace nezmiňuje možnosti spouštění přímo s různými nastaveními vnějších vlivů [27], 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. [13]

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, a umožňuje nastavení vlastností větru. [4], [18]

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 rozšiřitelný 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 rozšiřitelný 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 sekci 6.5). [25]

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). [26]

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 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. [13]

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. [11]

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 [22]. 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 simuloval 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. Systém využívá vlastní komponenty i komponenty od jiných autorů, které jsou dále popsány v podkapitole 7.1 včetně důvodů, proč byly zvoleny. Podkapitoly 7.2 a 7.3 popisují, jaké vstupy systém očekává a jakým způsobem jsou využity v komponentách, co je jeho výstupem a odkud daná informace pochází. Dále (podkapitola 7.4) je na systém nahlíženo zevnitř a jsou charakterizovány vazby mezi komponenty a jejich informační obsah.

## 7.1 Komponenty systému

### 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, jimiž 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 napodobeném světě a, podporuje-li to, i jejich vzájemné interakce. Také zprostředkovává vizuální pohled do této umělé reality prostřednictvím simulované kamery.

Z možností podporovaných kontorlérem 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 [28], 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 ale 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 [21].

### 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 kamery 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 svíslé 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 [14] s Python bindingy dt-apriltag [15].

### 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 popsáного v kapitole 5. Jedná se tedy o implementaci metody přistávání. UAV je ovládáno pomocí protokolu MAVLink, konkrétně prostřednictvím jeho implemetace v mavSDK pro Python [10].

### 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 [17], 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

### 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 spouštění 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 zadaných 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 pak uložen její název. Při spuštění mise se pak volají metody této vybrané metody.

### 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`, který obsahuje seznam definic misí a `experimenty.yaml`, který obsahuje seznam definic experimentů. Definice experimentu je seznam názvů misí a počtu jejich opakování v rámci experimentu.

### 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), 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

### 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 obraze jsou červeným čtyřúhelníkem vyznačeny všechny AprilTagy, které detektor zachytí, 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 (??), z nichž má každý název, který se na téže záložce také zobrazuje.

### 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.

### 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.

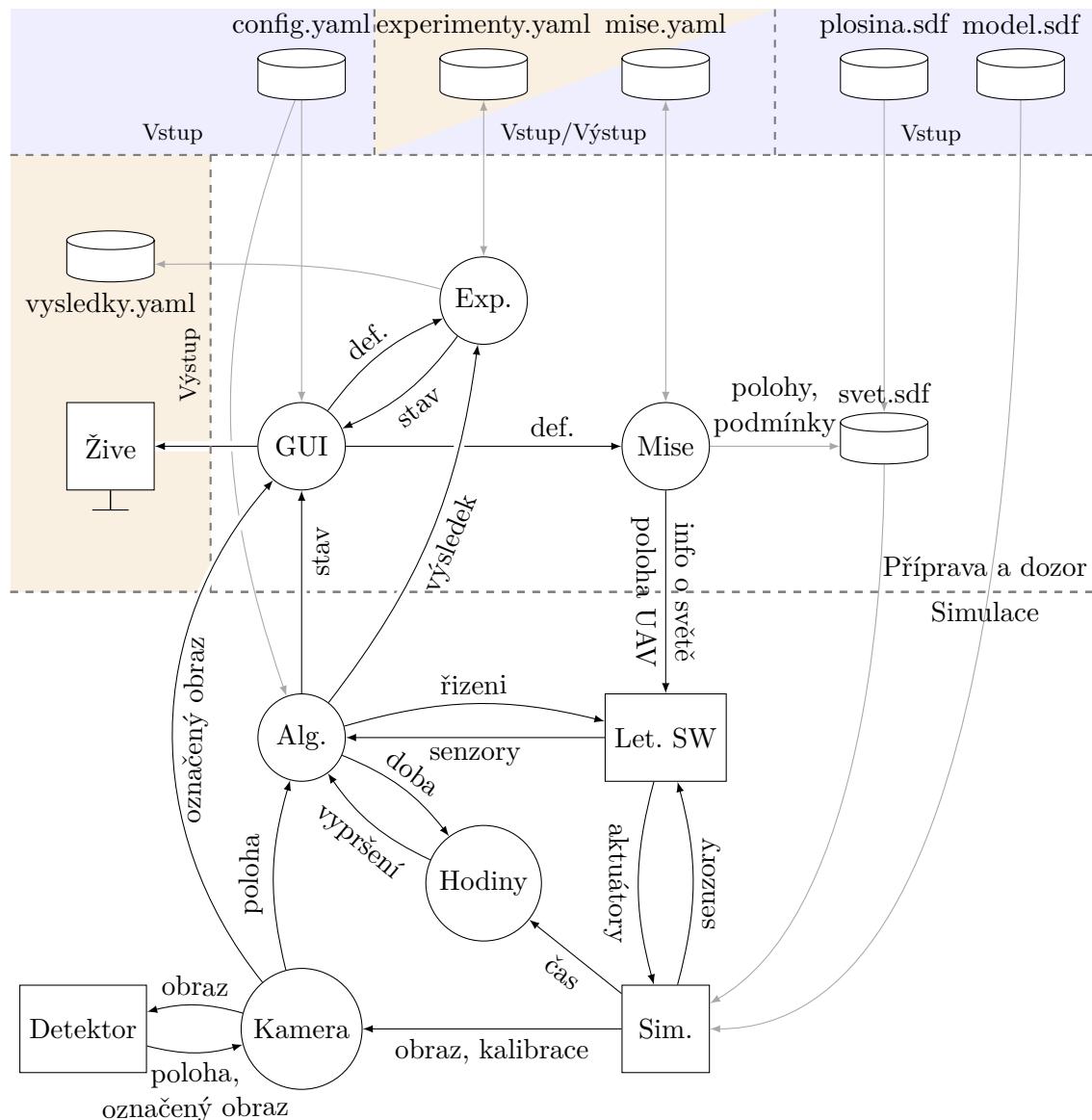
Přípravná část systému zahrnuje GUI, experimenty (v obrázku 7.1 zkráceno na Exp., u dalších komponent budou zkratky uvedeny v závorce samostatně), mise a definiční soubor prostředí (svet.sdf). S vnějším prostředím systému se vážou všechny tyto součásti. Experimenty a mise se načítají z příslušných souborů 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 řídící software (Let. SW), hodiny ze simulátoru, detektor fiduciárních markerů, kamera za simulátoru a simulátor. Mezi vazby s vnějším prostředím patří vstup přistávacího algoritmu, kterým je konfigurační soubor, stejný vstup má i kamera (tato vazba je ale 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 řídícího 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. Řídící 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 jeho 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 řídící 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 [28].

## 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  $\phi$  vůči ose  $x$ . Změna polohy aktualizuje vizualizaci v mapě ②.

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



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

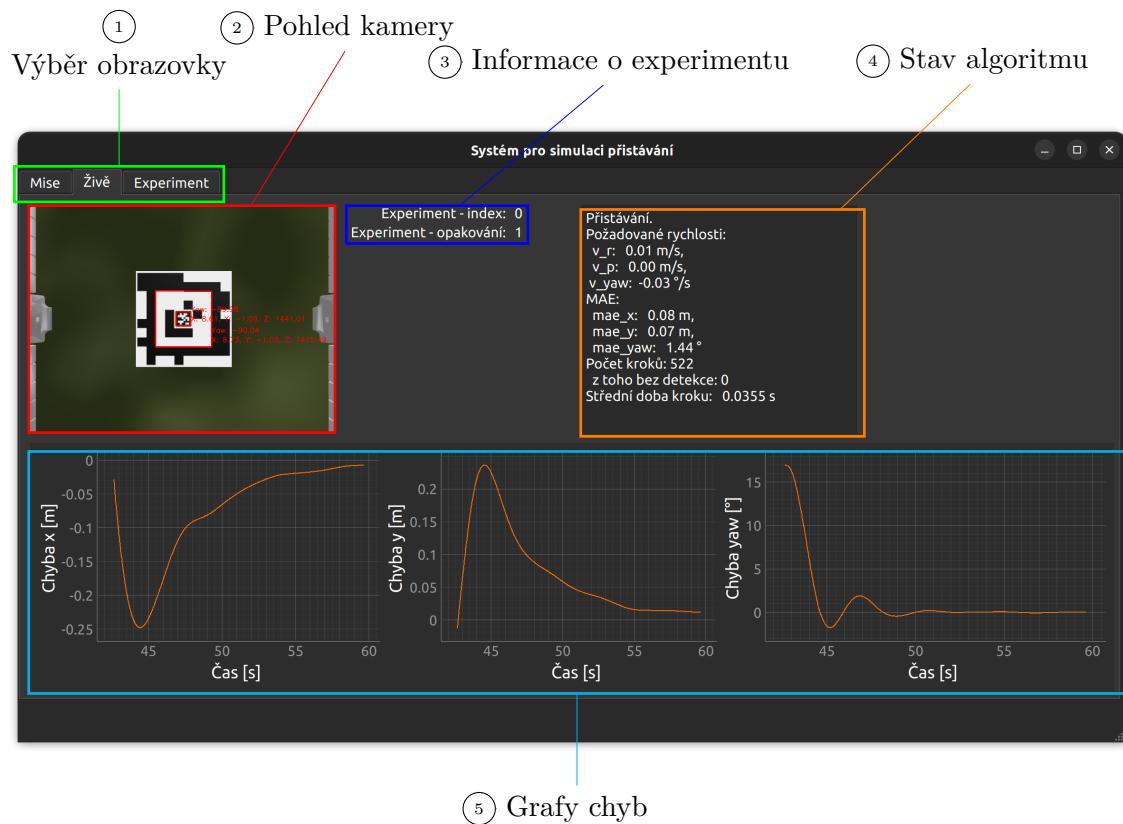
OBRÁZEK 8.2: Příklad simulace přistávání bezpilotního letounu na plošinu s fiduciárním markerem, která je částečně zastíněná.

- 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  $\sigma_v$ ) a vodorovný úhel vzhledem k ose  $x$  a jeho směrodatnou odchylku ( $\phi$  a  $\sigma_\phi$ ). 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 v 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.

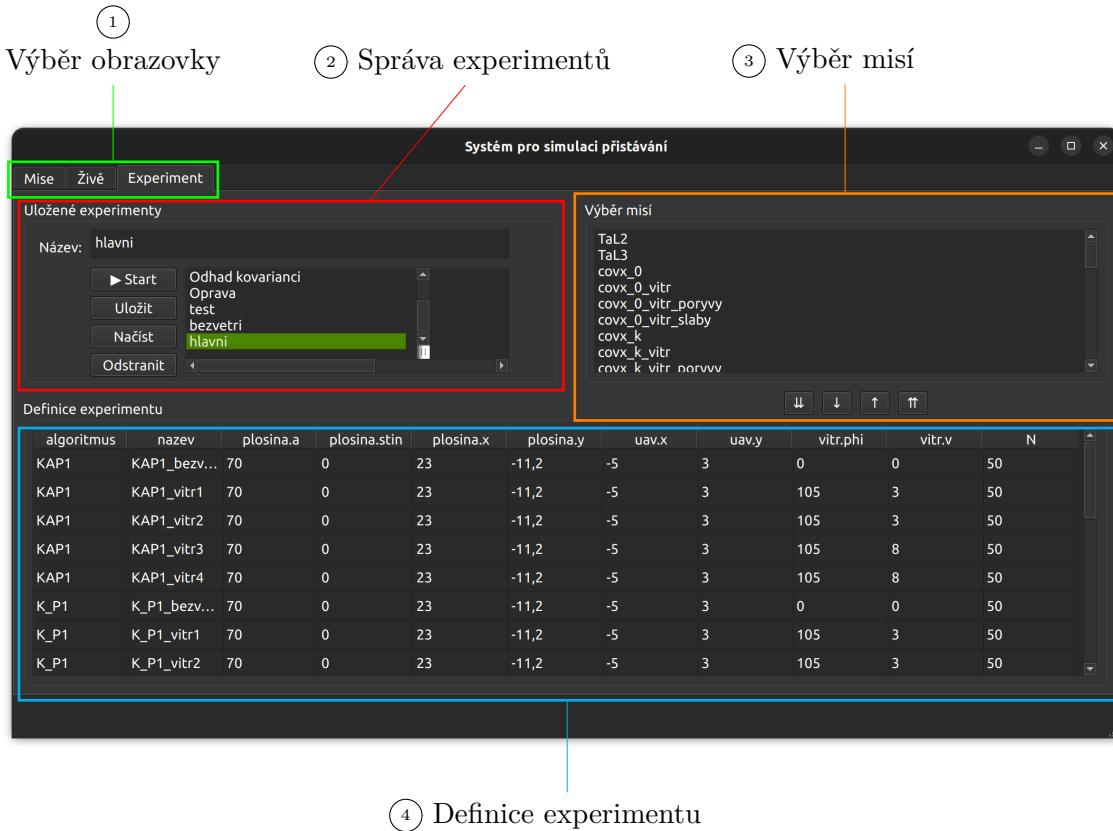
## 8.2 Obrazovka Živě

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.

- 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.



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



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

**(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, MSE v souřadnicích  $x$ ,  $y$  a rotaci kolem svislé osy, počtu již proběhlých kroků algoritmu a střední doby kroku.

**(5) Grafy chyb.** Spodní polovinu obrazovky zaujímají 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 skutečnou 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 (↓) 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 (↓↓). Odstranění misí z aktuálního experimentu se provádí obdobně pomocí zbylých dvou tlačítek se šipkami vzhůru (↑ a ↑↑), jen se mise označují v tabulce s definicí (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.

# 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.

## 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 a zaznamenávala se skutečná poloha letounu v simulačním prostředí v okamžiku jeho dosednutí. Během vyhodnocení se porovnávala tato zaznamenaná poloha s polohou plošiny dle definice dané mise. Z rozdílů v souřadnicích  $x$ ,  $y$  a rotaci  $\psi$  (kolem osy  $z$ ) byly pro každý z případů vypočteny výběrová střední hodnota a výběrový rozptyl. Shrnutí takto zjištěných výsledků obsahuje ?? a v grafické podobě zobrazující vizualizaci střední hodnoty, kovarianční elipsy a každého z přistání přímo na obrázku plošiny ??.

## 9.2 Přesnost přistávání

Další experiment zjišťoval přesnost celého přistávacího manévrů, kdy během každé ze simulací postupně počítal střední absolutní chybu (MSE) v souřadnicích  $x$ ,  $y$  a rotaci  $\psi$  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 v předchozím případě byl pokus proveden 50x pro každý prvek kartézského součinu metody přistávání  $\times$  třídy

třída	$v$ [m/s]	$\sigma_v$ [m/s]	$\varsigma$ [°]	$\sigma_\varsigma$ [°]
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.

větru. ?? shrnuje výběrové střední hodnoty a výběrové rozptyly MSE na konci přistávání pro dané podmínky simulace.

### 9.3 Úspěšnost přistávání

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

### 9.4 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 fiduciálních 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 pomocí PID regulátorů. Střední doba výpočtu jednoho kroku algoritmu je pro jednotlivé metody uvedena v tabulce ??.

### 9.5 Vliv stínu na podíl nedetekovaných markerů

Počítat podíl snímků, ve kterých nebyl detekován tag po zahájení přistávání.

# 10 Diskuze

Poznámky: U experimentu 9.2 by bylo dobré porovnat nejen metody, ale i ladění jejich PID. V podkapitole 8.1 je řečno, že se nevyužívá nastavení strany plošiny. Proč?

## 10.1 Možnosti reálného nasazení systému

# 11 Závěr

# 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] 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>.
- [4] B. Anton. „jMAVSim.“ (2013), URL: <https://github.com/DrTon/jMAVSim> (cit. 30. 04. 2024).
- [5] 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>.
- [6] 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>.
- [7] 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>.
- [8] H. Willee. „Simulation,“ ArduPilot. (2016), URL: <https://ardupilot.org/dev/docs/simulation-2.html> (cit. 29. 04. 2024).
- [9] R. Mackay. „Using SITL with RealFlight,“ ArduPilot. (2017), URL: <https://ardupilot.org/dev/docs/sitl-with-realflight.html> (cit. 30. 04. 2024).
- [10] 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).
- [11] 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>.

- [12] 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.
- [13] 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>.
- [14] 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.
- [15] A. Petrov. „Python bindings to the Apriltags library.“ (2019), URL: <https://github.com/duckietown/lib-dt-apriltags/> (cit. 04.05.2024).
- [16] 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.
- [17] Open Source Robotics Foundation. „SDFormat.“ (2020), URL: <http://sdformat.org/> (cit. 04.05.2024).
- [18] H. Willee. „jMAVSIM with SITL,“ PX4. (2020), URL: [https://docs.px4.io/main/en/sim\\_jmavsim/](https://docs.px4.io/main/en/sim_jmavsim/) (cit. 29.04.2024).
- [19] H. Willee. „Simulation,“ PX4. (2020), URL: <https://docs.px4.io/main/en/simulation/> (cit. 29.04.2024).
- [20] 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.
- [21] R. Mainwaring. „Python bindings for gz-msgs and gz-transport.“ (2021), URL: <https://github.com/srmainwaring/gz-python/> (cit. 03.05.2024).
- [22] Microsoft Research. „Home - Airsim.“ (2021), URL: <https://microsoft.github.io/AirSim/> (cit. 01.05.2024).
- [23] M. Pecka. „Gazebo ROS Battery plugin,“ České vysoké učení technické. (2023), URL: [https://github.com/ctu-vras/gazebo\\_ros\\_battery](https://github.com/ctu-vras/gazebo_ros_battery) (cit. 30.04.2024).
- [24] Betaflight. „SITL,“ Betaflight. (2024), URL: <https://betaflight.com/docs/development/sitl> (cit. 29.04.2024).
- [25] FlightGear. „Features – FlightGear Flight Simulator.“ (2024), URL: <https://www.flightgear.org/about/features/> (cit. 30.04.2024).
- [26] B. Jon. „JSBSim Open Source Flight Dynamics Model.“ (2024), URL: <https://jsbsim.sourceforge.net/> (cit. 30.04.2024).
- [27] RealFlight. „RealFlight RC Flight Simulator Software and Accessories.“ (2024), URL: <https://www.realflight.com/> (cit. 30.04.2024).
- [28] Riverbank Computing Ltd. „What is PyQt?“ (2024), URL: <https://www.riverbankcomputing.com/software/pyqt/> (cit. 03.05.2024).