



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.

1. května 2024



ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2023/2024

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	<b>Bc. Vojtěch BRENÍK</b>
Osobní číslo:	<b>A22N0105P</b>
Studijní program:	<b>N0714A150011 Kybernetika a řídicí technika</b>
Specializace:	<b>Umělá inteligence a automatizace</b>
Téma práce:	<b>Pokročilé algoritmy autonomního přistávání bezpilotního letounu na plošině</b>
Zadávací katedra:	<b>Katedra kybernetiky</b>

## 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 . . . . .	9
5.5 Přistávání po skloněné přímce s využitím Kálmánova filtru . . . . .	10
<b>6 Simulátory bezpilotních letadel</b>	<b>11</b>
6.1 Gazebo . . . . .	12
6.2 RealFlight . . . . .	13
6.3 jMAVSim . . . . .	13
6.4 FlightGear . . . . .	13
6.5 JSBSim . . . . .	13
6.6 Airsim . . . . .	14
<b>7 Návrh systému pro přistávání</b>	<b>15</b>
7.1 Vstupy systému . . . . .	15
7.2 Komponenty systému . . . . .	15
7.3 Výstupy systému . . . . .	15
7.4 Struktura systému . . . . .	15
<b>8 Grafické uživatelské rozhraní</b>	<b>16</b>
<b>9 Experimenty a vyhodnocení</b>	<b>17</b>
<b>10 Diskuze</b>	<b>18</b>



10.1	Možnosti reálného nasazení systému . . . . .	18
<b>11</b>	<b>Závěr</b>	<b>19</b>
	<b>Bibliografie</b>	<b>20</b>

# Seznam obrázků

5.1	Přistávání po skloněné přímce . . . . .	10
-----	---	----

# Seznam tabulek

6.1	Simulátory podporované kontrolérem letounu . . . . .	11
6.2	Vlastnosti vybraných simulátorů . . . . .	12

# Seznam zkratek

- . 25
- API** Application Programming Interface - rozhraní pro programování aplikací). 1, 18, 32, 39
- . 12–14
- . 26, 34
- . 12, 19–21, 26, 27, 29, 30
- . 2, 8
- . 6, 7, 25, 26, 34
- . 13, 14, 16, 39
- MSE** mean squared error - střední kvadratická chyba. 22, 25, 33–35, 37, 39, 40
- . 2, 9, 21–23, 25, 26, 33, 36, 39–43
- . 3, 4
- . 19

# 1 Úvod

## 1.1 Bezpilotní letadlo

UAV - využití, aplikace, součástí mise je přistávání - potřeba jeho automatizace; druhy, historie, specifiky; zúžení na čtyřrotorový pro účely práce S klesající teplotou roste vnitřní elektrický odpor akumulátoru UAV (Łebkowski, 2017), 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, radiomajáky?)

## 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 spjaté s místem jejich konstrukce, zatímco u přenosných a pohyblivých lze měnit místo přistání. Obvykle se takto označují místa pro přistání letadel s největším rozměrem nejvýše 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 nepoužívanější 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 bylo použito několik plošin s různými rozměry. Každá plošina byla zkonstruována jako deska tloušťky [doplnit tloušťku] tvaru čtverce o straně a), b), c) [doplnit strany]. Celou plošinu vždy pokrýval rekursivní fiduciální marker Apriltag o 2 vrstvách. Vnější vrstva o rozměrech 10x10 měla uprostřed díru o rozměrech 2x2 (rodina Custom48h12), do které byl umístěn tag vnitřní vrstvy o rozměrech 8x8 (rodina [doplnit]) obklopený oddělovací čarou o tloušťce 1. Rozměry značek byly uvedeny včetně okraje, přičemž vnější značka měla jednu datovou vrstvu na obvodu okraje.

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.

### 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 nějakého zájmového bodu, jeho měřítko nebo jeho identifikaci. V robotice se podle jejich obrazu zachyceného kamerou může určovat vzájemná poloha kamery a předmětu, na němž je značka připevněna. (Košťák a Slabý, 2021) 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 fiduciárních systémů, 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 (Fiala, 2005), 5. doba trvání detekce (Garrido-Jurado et al., 2014; Wang a Olson, 2016) 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 bity z 36). Má nízkou míru falešných pozitiv i míru záměny značky za jinou. Ve srovnání se starším ARToolkit má vylepšenou identifikaci a verifikaci vzorů a snižuje tak míru záměn značek. Přesto má větší knihovnu možných vzorů. (Fiala, 2005)

### 3.1.2 Aruco

Fiducírní 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 míra záměn značek, a co největší počet přechodů mezi bity, 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. Wang a Olson, 2016 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. Krogus, Haggemiller a Olson, 2019

Do díry v tagu je možné umístit další tag a vytvořit tak rekurzivní značku, kterou bude možné detekovat ve větším intervalu vzdáleností (Krogus, Haggemiller a Olson, 2019), 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 obraze, algoritmus, který toto provádí se pak nazývá detektor.

Mezi algoritmy používané při detekci fiduciárních markerů patří prahování, vyhledávání vzorů (template matching), hranové detektory (Wang a Olson, 2016) nebo Houghova transformace (Shabalina et al., 2019). Dále je pospáno, 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 (Wang a Olson, 2016) se autoři chtějí vyhnout nespojitostem na hranicích buněk a tak používají extrémy z osmiokolí (po buňkách). Ve verzi 3 (Krogius, Haggenmiller a Olson, 2019) potom zjistili, že je vhodnější decimaci oddělit od prahování a používají bodové převzorkování, které lépe zachová hrany v obraze, 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 obraze, 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. obarvení). (Wang a Olson, 2016) 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. (Krogius, Haggenmiller a Olson, 2019)

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 aproximují 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 aproximuje přímkou. Vybere se taková kombinace rohů, pro kterou je MSE nejmenší. (Wang

a Olson, 2016) Ve verzi 3 se řazení bodů provádí podle kvadrantu, ve kterém se bod nachází a sklonu průvodiče. Není tak nutné počítat explicitně úhel. (Krogus, Haggemiller a Olson, 2019)

Nalezené čtyřúhelníky 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. (Wang a Olson, 2016) Verze 3 zavádí bilineární interpolaci buněk tagu a zvyšuje ostrost za účelem zlepšení detekce malých značek. (Krogus, Haggemiller a Olson, 2019)

## 5 Metody přistávání

Součástí této práce je návrh systému pro simulaci přistání bezpilotního letounu na plošinu, 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 řídicí 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 rychlost  $v_o$  a směr větru  $\varsigma_o$  jsou náhodné s normálním rozdělením

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

filtrovaným lineárním systémem 1. řádu s časovou konstantou  $\tau =$ . Parametry normálního rozdělení  $v, \varsigma, \sigma_v$  a  $\varsigma$  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-jihním 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 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

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

k ní UAV začít přibližovat. Způsob přibližování je závislý na konkrétní metodě a právě jím se jednotlivé metody liší. Pokud se naopak nepodaří značku detekovat, 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 rychlost klesání, dokud dron nedosedne na plošinu. Poté je přistávání ukončeno. Požadovaná rychlost klesání se stanoví podle vzorce (5.1):

$$v_z = \quad (5.1)$$

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.

## 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 (princip jeho fungování je popsán v podkapitole ??) a měření polohy získaná z detektoru, která jsou zatížena šumem, se pak filtrují, čímž je dosaženo vyšší přesnosti odhadu polohy. Filtrované odhady se dále používají stejným způsobem jako v předchozí metodě a je podle nich řízena vodorovná rychlost UAV.

Pro tyto účely byl letoun modelován lineárním stochastickým systémem diskretním 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 rychlost  $\dot{\psi}$  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 rozdělením  $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$  s kovarianční maticí  $\mathbf{Q}$ , která byla určena řádově podobně jako Kojima a Namerikawa (2015) 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  $\psi$ . 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  $z$  potom popisuje ??.

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:

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

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ z_k \\ \dot{z}_k \\ \psi_k \\ \dot{\psi}_k \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.3)$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix}, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \text{kde} \quad (5.4)$$

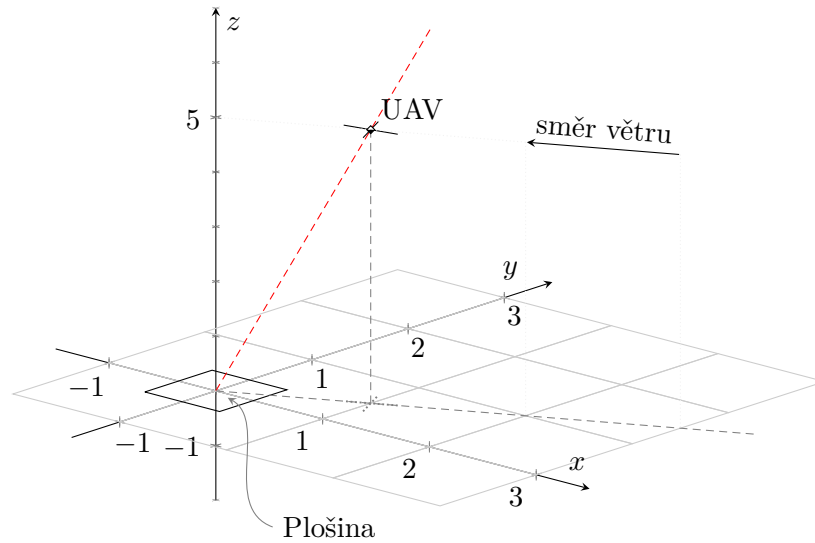
$$\mathbf{Q} = \begin{bmatrix} \dots \end{bmatrix} \quad (5.5)$$

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 rychlost a daný směr větru) kamera UAV mířila na střed přistávací plošiny. Rychlost se v závislosti na odchylce od požadované trajektorie vypočte stejným způsobem jako v předchozích případech, tedy dle vztahu (5.1).



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

Tato metoda má za cíl zlepšit podmínky pro detekci značky na plošině při působení silného větru. V případě nutné korekce nějaké poruchy (způsobené například porывem 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 rychlost 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 (Willee, 2020b; Willee, 2016; Betaflight, 2024).

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 vyžadována velmi rychlá a přesná odezva na vstupy z rádiového dálkového ovládání od pilota a poruchy způsobené například pohyby vzduchu v atmosféře. S různými zaměřenímí 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 (Willee, 2020b; Willee, 2016; Betaflight, 2024).

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

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 softwary.

	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; „-“ vy značuje 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.

(sekce 6.1 až 6.6) jsou zevrubně popsány hlavní funkce jednotlivých simulátorů a jejich 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 simulacích 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. (Ebeid et al., 2018)

Co se týče simulace vnějších vlivů, Gazebo nativně umožňuje měnit podmínky osvětlení, simulaci větru je možné provést pomocí doplňku distribuovaného společně s Gazebem 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 Gazeba označované jako Gazebo classic je možné simulovat chování baterie pomocí pluginu (Pecka, 2023), 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 (Mackay, 2017; RealFlight, 2024). Kromě toho oficiální dokumentace nezmiňuje možnosti spouštění přímo s různými nastaveními vnějších vlivů (RealFlight, 2024), 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 multirotových 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. Ebeid et al., 2018

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. (Anton, 2013; Willee, 2020a)

Nemožnost simulace různých světelných podmínek, obtížná rozšiřitelnost 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). (FlightGear, 2024)

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 řídicích 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). (Jon, 2024)

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, počítačového 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. (Ebeid et al., 2018)

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 Gazeba, kde je nutné tato nastavení předem určit v definičním souboru světa. (Shah et al., 2017)

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 (Research, 2021). 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 přistávání

7.1 Vstupy systému

7.2 Komponenty systému

7.3 Výstupy systému

7.4 Struktura systému

## 8 Grafické uživatelské rozhraní

Budoucí výstup projektu - rozhraní pro spawnování dronu, plošiny a nastavení simulátoru atd.

## 9 Experimenty a vyhodnocení

Porovnání vybraných algoritmů

# 10 Diskuze

## 10.1 Možnosti reálného nasazení systému

## 11 Závěr

# Bibliografie

- [1] 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>.
- [2] Babushkin Anton. *jMAVSim*. 2013. URL: <https://github.com/DrTon/jMAVSim> (cit. 30.04.2024).
- [3] S. Garrido-Jurado et al. „Automatic generation and detection of highly reliable fiducial markers under occlusion“. In: *Pattern Recognition* 47.6 (čvn. 2014), 2280–2292. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.01.005. URL: <http://dx.doi.org/10.1016/j.patcog.2014.01.005>.
- [4] Takaaki Kojima a Toru 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>.
- [5] John Wang a Edwin 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/iro.2016.7759617. URL: <http://dx.doi.org/10.1109/IROS.2016.7759617>.
- [6] Hamish Willee. *Simulation*. ArduPilot. 2016. URL: <https://ardupilot.org/dev/docs/simulation-2.html> (cit. 29.04.2024).
- [7] Randy Mackay. *Using SITL with RealFlight*. ArduPilot. 2017. URL: <https://ardupilot.org/dev/docs/sitl-with-realflight.html> (cit. 30.04.2024).
- [8] Shital Shah et al. „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>.
- [9] Andrzej Łebkowski. „Temperature, Overcharge and Short-Circuit Studies of Batteries used in Electric Vehicles“. In: *Przegląd Elektrotechniczny* 1 (květ. 2017). DOI: 10.15199/48.2017.05.13.
- [10] Emad Ebeid et al. „A survey of Open-Source UAV flight controllers and flight simulators“. In: *Microprocessors and Microsystems* 61 (zář. 2018), 11–20. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2018.05.002. URL: <http://dx.doi.org/10.1016/j.micpro.2018.05.002>.
- [11] Maximilian Krogus, Acshi Haggemiller a Edwin Olson. „Flexible Layouts for Fiducial Tags“. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019.
- [12] Ksenia Shabalina et al. „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. URL: [http://dx.doi.org/10.1007/978-3-030-11292-9\\_27](http://dx.doi.org/10.1007/978-3-030-11292-9_27).
- [13] Hamish 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).



- [14] Hamish Willee. *Simulation*. PX4. 2020. URL: <https://docs.px4.io/main/en/simulation/> (cit. 29.04.2024).
- [15] Milan Košťák a Antonín Slabý. „Designing a Simple Fiducial Marker for Localization in Spatial Scenes Using Neural Networks“. In: *Sensors* 21.16 (srp. 2021), s. 5407. ISSN: 1424-8220. DOI: 10.3390/s21165407. URL: <http://dx.doi.org/10.3390/s21165407>.
- [16] Microsoft Research. *Home - Airsim*. 2021. URL: <https://microsoft.github.io/AirSim/> (cit. 01.05.2024).
- [17] Martin 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).
- [18] Betaflight. *SITL*. Betaflight. 2024. URL: <https://betaflight.com/docs/development/sitl> (cit. 29.04.2024).
- [19] FlightGear. *Features – FlightGear Flight Simulator*. 2024. URL: <https://www.flightgear.org/about/features/> (cit. 30.04.2024).
- [20] Berndt Jon. *JSBSim Open Source Flight Dynamics Model*. 2024. URL: <https://jsbsim.sourceforge.net/> (cit. 30.04.2024).
- [21] RealFlight. *RealFlight RC Flight Simulator Software and Accessories*. 2024. URL: <https://www.realflight.com/> (cit. 30.04.2024).