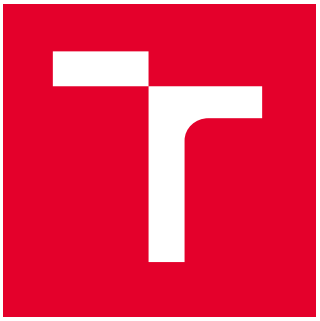


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

PLÁNOVÁNÍ LETOVÝCH TRAJEKTORIÍ PRO ROJE DRONŮ

FLIGHT PATH PLANNING FOR DRONE SWARMS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Martin Procházka**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jiří Janoušek**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Martin Procházka ***ID:*** 230160

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Plánování letových trajektorií pro roje dronů

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou automatického plánování přesné letové trajektorie pro roj dronů dle obsluhou vybraných letových formací a navrhněte v prostředí softwaru Blender sadu skriptů v jazyce Python, které umožní uživateli v grafickém prostředí softwaru pomocí doplňků automaticky vytvářet letové trajektorie pro zvolené formace a seskupení roje dronů, zaměřte se také na trajektorie při přeletu dronů mezi jednotlivými formacemi v 3D prostoru.
2. Vytvořené doplňky rozšiřte o uživatelské funkce volby počtu dronů, výšky a vzdáleností letu, volby formací, změny seskupení v prostoru, změny vlastností parametrů dronů a definice rozměrů letového perimetru.
3. Otestujte funkčnosť navržených doplňků simulací libovolného letového scénáře, přidejte funkci exportování letové trasy a popište formát výstupu.
4. Implementujte skript pro kontrolu chyby lidského faktoru při plánovíní letových formací, který automaticky zkontroluje uživatelem zvolenou velikost letových formací, zejména z pohledu dodržení vzájemných vzdáleností dronů a rychlostí letu, dále také dodržení maximálních rozměrů letového perimetru.
5. Ověřte funkčnost vytvořených doplňků testováním na poskytnutých dronech při reálném letu s využitím všech jejich funkcí pro plánování letových trajektorií.

DOPORUČENÁ LITERATURA:

K. Eliker, H. Bouadi and M. Haddad, "Flight planning and guidance features for an UAV Flight Management Computer," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016, pp. 1-6, doi: 10.1109/ETFA.2016.7733735.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

Vedoucí práce: Ing. Jiří Janoušek

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

ABSTRAKT

Tato práce se zabývá problematikou plánování bezpečných letových trajektorií pro roje bezpilotních prostředků s důrazem na praktickou použitelnost během tvorby samotné letové mise. Společně s návrhem samotného plánovacího algoritmu je také v rámci této práce realizována sada doplňků pro 3D modelovací prostředí Blender, pomocí kterých je možné navrhnout letovou misi, ověřit bezpečnost a dodržení stanovených limitů mise a následně samotnou misi exportovat pro praktickou realizaci.

KLÍČOVÁ SLOVA

Roj dronů, plánování trajektorie, Blender, Python

ABSTRACT

This bachelor's thesis addresses the issue of planning flight paths for drone swarms with specified safety constrains and with the emphasis on the feasibility of real-world usage. Along with the design of the planning algorithm itself, this thesis also implements an add-on for the 3D environment Blender, which can be used to design a flight mission, verify safety constrains and export the mission itself.

KEYWORDS

Drone swarm, path planning, Blender, Python

PROCHÁZKA, Martin. *Plánování letových trajektorií pro roje dronů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 49 s. Bakalářská práce. Vedoucí práce: Ing. Jiří Janoušek

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Martin Procházka
VUT ID autora:	230160
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Plánování letových trajektorií pro roje dronů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

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

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Tato bakalářská práce se zabývá tvorbou řešení pro plánování letových formací pro letky dronů.

Počet oblastí, ve kterých je možno prakticky uplatnit použití dronů zejména v poslední době přibývá. Jednou takovou oblastí je realizace světelných představení. Pro jejich realizaci je zapotřebí návrhového prostředí, které dokáže jednoduše pracovat jak s objekty ve 3D, tak i omezeními, které použití dronů přináší. Toto omezení spočívá ve vytvoření letové trasy, která dodržuje předem stanovený rozestup od okolních jednotek a také při jejimž vykonávání daný dron nepřekročí rychlostní limit.

Hlavním cílem této práce je zvolit plánovací metodu, která umožní naplánovat letové trasy pro přelet mezi jednotlivými 3D formacemi a ten následně integrovat do modelovacího prostředí Blender pro praktické a uživatelsky přívětivé použití.

1 Plánovací algoritmus

Cílem plánovacího algoritmu je propojit dvě shodně velké množiny bodů v 3D prostoru, které reprezentují jednotlivé pozice počáteční a cílové formace. Při tomto úkonu je zejména potřeba zajistit bezpečnou vzdálenost mezi jednotlivými drony během celé doby provádění naplánovaného přeletu.

1.1 Dostupná řešení

Při prostudování dostupných řešení, které se týkají této problematiky, je možné se setkat s přístupy, které se výše popsaný problém snaží vyřešit. Tato řešení jsou velmi výpočetně náročná, což značně limituje jejich použití, jelikož by uživatel byl nucen čekat, než se výpočet dokončí. V rámci této bakalářské práce bych proto rád vyžil dvou konceptů, pomocí kterých je možné realizovat naplánování přeletu v rozumném čase, konkrétněji v několika sekundách pro řádově desítky dronů, a zároveň také zajistit požadavky na bezpečné provedení letu.

Prvním přístupem k plánování je realizace trajektorií formou úsečky. Následně jsou zvoleny vhodné páry pozic ze startovní a cílové množiny tak, aby se minimalizovala celková dráha letu. Díky tomuto se plánovací proces zásadně výpočetně zjednoduší, což jej dovolí provést v nízkém časovém horizontu [1]. Nevýhodou tohoto přístupu je zanedbání fyzických rozměrů dronů a nemožnost zajistit vzájemný bezpečný rozestup během přeletu.

Dalším konceptem, se kterým je možno se setkat, je využití metody umělých potenciálových polí, kde je výsledná trajektorie určena simulací působení přitažlivých a odpudivých sil. Nevýhodou této metody je vysoká výpočetní náročnost a možnost vzniku lokálních minim, díky kterým nemusí veškeré trajektorie končit v příslušných cílových pozicích. [2]

2 Umělá potenciálová pole

Metoda umělých potenciálových polí je jednou z metod, kterou lze použít pro plánování trasy prostředím, zejména pak pro větší množství navigovaných objektů zároveň. Tato metoda je inspirována přírodními zákony. Konkrétně chováním náboje v elektrickém poli, kdy je výsledná trajektorie náboje určena přitažlivými a odpudivými silami, které na daný náboj během jeho cesty působí. V případě plánování trasy je tedy náboj reprezentován naváděným objektem, který je přitahován ke své cílové pozici a zároveň je odpuzován od nežádoucích pozic, které tvoří překážky prostředí. Obdobně, jako v případě Coulombova zákona, je velikost působící síly závislá na vzdálenosti od zdroje daného potenciálu. Výslednou trajektorii lze získat tak, že bude provedena simulace působení sil na naváděný objekt a takto získaná trajektorie naváděného objektu bude samotným výstupem této plánovací metody [2].

2.1 Charakteristika potenciálů

Stejně tak, jako v případě elektrických polí, má náboj tendenci zaujmout pozici s nejnižším potenciálem. Nicméně narozdíl od přírodních zákonů, je možné charakteristiky potenciálů definovat dle libovolných požadavků, a tím i výsledné potenciálové pole. Příkladem tohoto může být charakteristika potenciálu na vzdálenosti od cílové pozice, u kterého by bylo nežádoucí, aby se projevoval pouze v blízkém okolí cíle, ale aby prostupoval napříč celým prostorem, díky čemuž bude vždy působit na naváděný objekt. Matematicky lze toto pole popsat následovně [3]:

$$\varphi_C(O) = \frac{1}{2} K_C \|C - O\|^2 \tag{2.1}$$

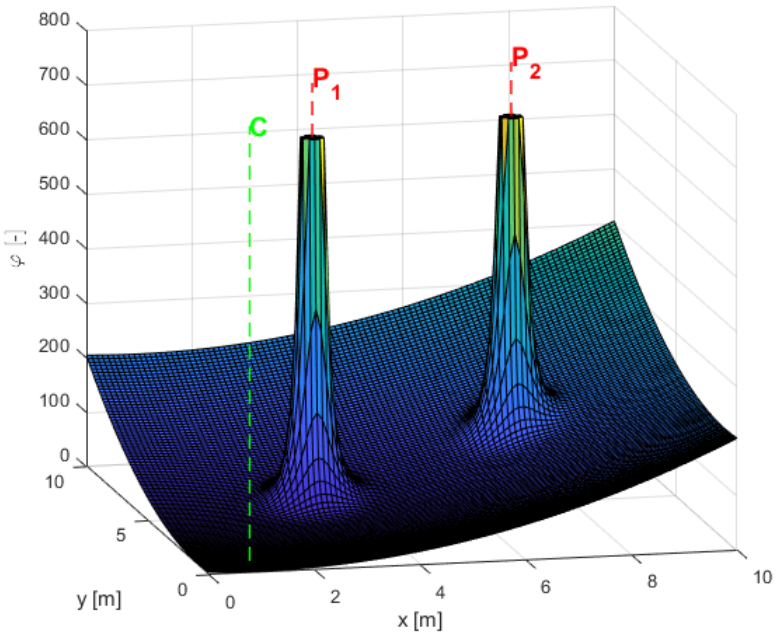
Kde $\mathbf{C}$ značí souřadnici cílové polohy, $\mathbf{O}$ aktuální souřadnici naváděného objektu, $\mathbf{K_C}$ konstantu určující velikost potenciálu a $\varphi_\mathbf{C}$ samotnou velikost potenciálu v místě naváděného objektu.

Mimo cílovou pozici je potenciálové pole také ovlivňováno překážkami. V tomto případě je našim cílem zajistit, aby se naváděný objekt k překážce nepřiblížil na příliš blízkou vzdálenost. Z tohoto důvodu je vhodné, aby velikost potenciálu nepřímo rostla se vzdáleností, avšak zároveň je také žádoucí, aby narozdíl od charakteristiky potenciálu cílové pozice, překážka ovlivňovala celkové potenciálové pole pouze ve své bezprostřední blízkosti. Předpis této charakteristiky je pak následující [3]:

$$\varphi_{P_n}(O) = \begin{cases} \frac{1}{2} K_p (\frac{1}{\|P_n - O\|} - \frac{1}{\rho})^2 & , \|P_n - O\| \leq \rho \\ 0 & , \|P_n - O\| > \rho \end{cases} \tag{2.2}$$

Kde $\mathbf{P_n}$ značí souřadnici $\mathbf{n}$-té překážky, $\mathbf{O}$ aktuální souřadnici naváděného objektu, $\mathbf{K_p}$ konstantu určující velikost potenciálu, ρ představující poloměr oblasti, ve které

se potenciál překážky projevuje, a φ_{P_n} samotnou velikost potenciálu v místě navá-děného objektu.



Obr. 2.1: Příklad potenciálového pole 2D úlohy

Příklad výsledného potenciálového pole je zobrazen na obrázku 2.1, které bylo vytvořeno s následujícími parametry: $\mathrm{K_C} = 5$, $\mathrm{K_p} = 50$ a $\rho = 1{,}5$.

2.2 Působení na naváděný objekt

Po získání potenciálového pole můžeme za pomocí výpočtu gradientu získat v daném místě žádaný směr pohybu, pomocí kterého je možno se přesunout k místu s nižším potenciálem [3].

$$\vec{v}(O) = -\nabla\varphi(O) \tag{2.3}$$

Pro získání celkového vektoru rychlosti je tedy zapotřebí najít velikosti jednotlivých složek, které na naváděný objekt působí. První z nich je přitažlivé působení na objekt k cílové pozici a jeho přepis vypadá následovně [3]:

$$\vec{v}_C(O) = K_C\|C - O\| \cdot \nabla O \tag{2.4}$$

Zbylé působení má odpudivý charakter od překážek a lze jej popsat takto [3]:

$$\vec{v}_{P_n}(O) = \begin{cases} -K_p(\frac{1}{\|P_n - O\|} - \frac{1}{\rho})\frac{1}{\|P_n - O\|^2} \cdot \nabla O & , \|P_n - O\| \leq \rho \\ 0 & , \|P_n - O\| > \rho \end{cases} \tag{2.5}$$

Po zjištění jednotlivých příspěvků působení stačí již pouze tyto jednotlivé složky sečíst, a tím získat výsledné působení:

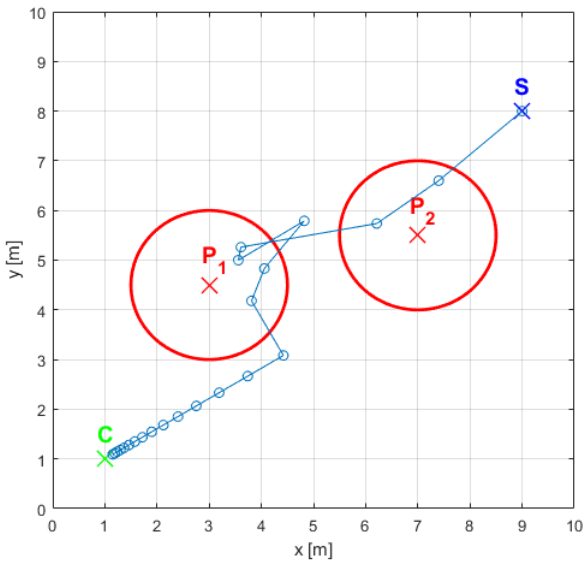
$$\vec{v}(O) = \vec{v}_C(O) + \sum_{n=1}^{m} \vec{v}_{P_n}(O) \tag{2.6}$$

Kde **m** představuje počet překážek.

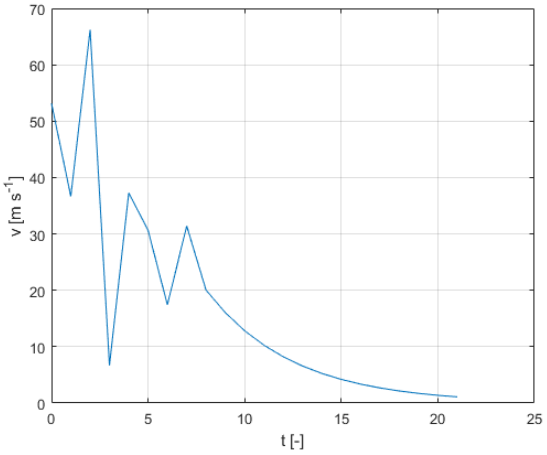
Závěrem je aktualizována pozice o aktuální přírůstek, který plyne z požadované rychlosti a z časového kroku simulace:

$$O_{t+1} = O_t + \frac{\vec{v}(O_t)}{f_{FPS}} \tag{2.7}$$

Kde **t** značí krok simulace a $\mathbf{f_{FPS}}$ snímkovací frekvenci.



Obr. 2.2: Příklad simulace 2D úlohy



Obr. 2.3: Průběh velikosti žádané rychlosti pohybu

Dle výše popsaného postupu byla provedena simulace přeletu z pozice S do pozice C s překážkami P_1 a P_2, jejíž výsledek je zobrazen na obrázku 2.2. Zároveň je zde vyobrazeno pole působnosti jednotlivých překážek. Simulace byla provedena za následujících parametrů: $K_C = 5$, $K_p = 50$ a $\rho = 1{,}5$.

Z výsledné trajektorie je jasně patrné, že ji není možné prakticky realizovat, což je možné si ověřit na grafu žádané rychlosti 2.3, kde se požadovaná rychlost pohybuje násobně mimo realizovatelnou mez, která se typicky pohybuje v jednotkách metrů za sekundu. Z tohoto důvodu je potřeba upravit vztah aktualizace polohy 2.7 tak, aby bylo možné limitovat rychlost dle požadavků. Mimo samotnou maximální rychlost je také vhodné, aby se rychlost pohybu zmenšovala v případě, že se naváděný objekt nachází v blízkosti překážky a cílové či startovní pozice. Díky tomuto je možné předejít velkým skokům v trajektorii, které by mohly vést na nežádoucí chování. Faktor, který bude velikost rychlosti určovat, lze vyjádřit následovně [3]:

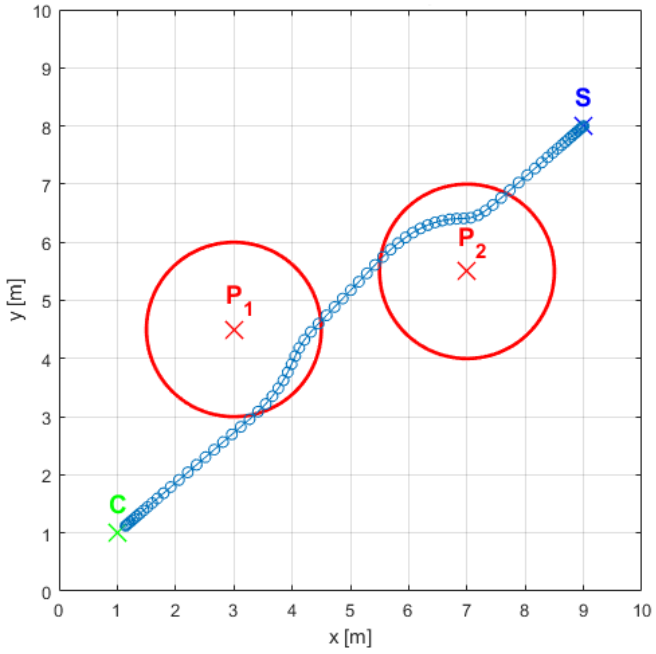
$$l = \frac{Min(\rho, d_C, d_S + \mu, d_{P_1}, ..., d_{P_n})}{\rho} \tag{2.8}$$

Kde $\mathbf{d_{P_n}}$ značí vzdálenost od n-té překážky, $\mathbf{d_C}$ vzdálenost od cílové pozice, $\mathbf{d_S}$ vzdálenost od startovní pozice, μ malou kladnou hodnotu, aby nebyl výsledek první iterace nulový a $\mathbf{l}$ je výsledný relativní faktor rychlosti.

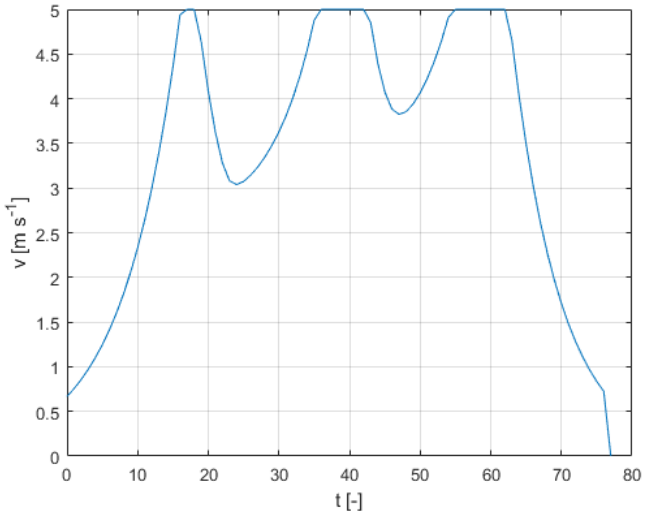
Takto získaný faktor bude určovat, jakou částí maximální rychlosti se má naváděný objekt právě pohybovat. Směr samotného pohybu lze získat z původního vektoru, který je nyní potřeba normalizovat na jednotkovou velikost. Výsledný vztah aktualizace polohy vypadá následovně:

$$O_{t+1} = O_t + \frac{\vec{v_n}(O_t) \cdot l \cdot v_{max}}{f_{FPS}} \tag{2.9}$$

Kde $\vec{v_n}(O_t)$ představuje normalizovaný vektor rychlosti a $\mathbf{v_{max}}$ maximální povolenou rychlost.



Obr. 2.4: 2D úloha s limitovanou rychlostí

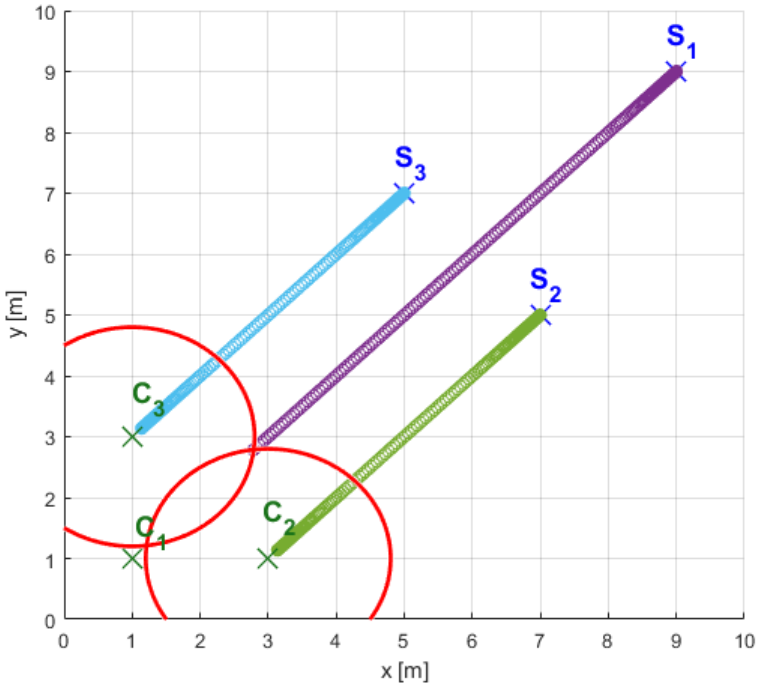


Obr. 2.5: Průběh limitované velikosti žádané rychlosti pohybu

S využitím limitace rychlosti, a tím i tedy maximální velikosti skoku polohy, pomocí upravené rovnice 2.9 byla provedena simulace stejného scénáře, jako je vyobrazen na obrázku 2.2, a s totožnými parametry. Výsledná trajektorie je zobrazena na obrázku 2.4 a správnost limitace rychlosti je možné si ověřit na grafu žádané rychlosti 2.5.

2.3 Ošetření lokálních minim

Jednou z nevýhod použití metody umělých potenciálových polí je ta, že existuje možnost vzniku lokálních minim, kde se naváděný objekt přestane přibližovat k cílové pozici, jelikož bude přitažlivé působení cílové pozice vykompenzováno odpudivým působením překážek.



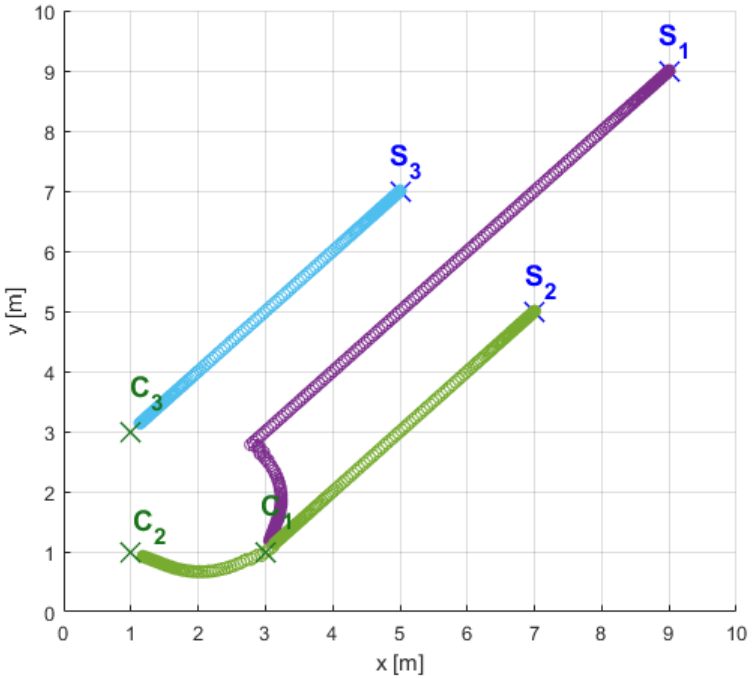
Obr. 2.6: Příklad vzniku lokálního minima

Tento případ je ilustrován na obrázku 2.6, kdy přítomnost obou bočních naváděných objektů blokuje svým působením průchodu prostředního naváděného objektu, a tudíž nedojde k úspěšnému ukončení simulace. Z tohoto důvodu je zapotřebí implementovat systém, který by dokázal tyto situace vyřešit.

Lokální minimum může vzniknout pokud jsou příslušné překážky statické a není je možno pomocí odpudivého působení odsunout. Aby bylo docíleno opětovného

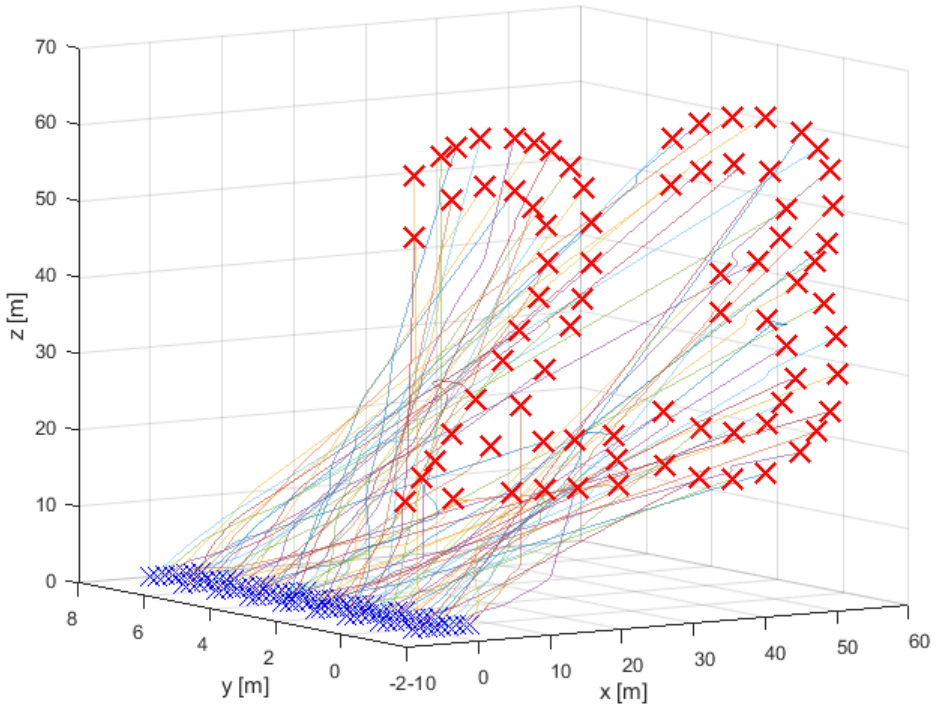
rozpohybování, dojde k záměně cílových pozic, díky čemuž se původně blokující objekt opět začne pohybovat a původně blokovaný objekt zaujme volnou pozici po blokujícím objektu. Pro vyvolání záměny musí být splněny následující podmínky [3]:

- na naváděný objekt působí 2 a více objektů, které již dosáhly cílové pozice
- alespoň jeden z blokujících objektů se nachází blíže k cíli, než samotný naváděný objekt
- výsledná změna polohy naváděného objektu jej vzdaluje od cíle



Obr. 2.7: Trajektorie s využitím záměny cíle

Dle popsaného postupu byl implementován systém pro záměnu cílových pozic a z obrázku 2.7 je jasně patrné, že je možné pomocí něj vyřešit situaci se vznikem lokálního minima, a tím provést úspěšné naplánování letových tras.



Obr. 2.8: Příklad 3D úlohy

Pro ověření funkčnosti plánovací metody umělých potenciálových polí, implementované dle výše popsaných pravidel, bylo provedeno naplánování přeletu mezi dvěma formacemi v 3D prostoru. Výsledné trajektorie jsou zobrazeny na obrázku 2.8.

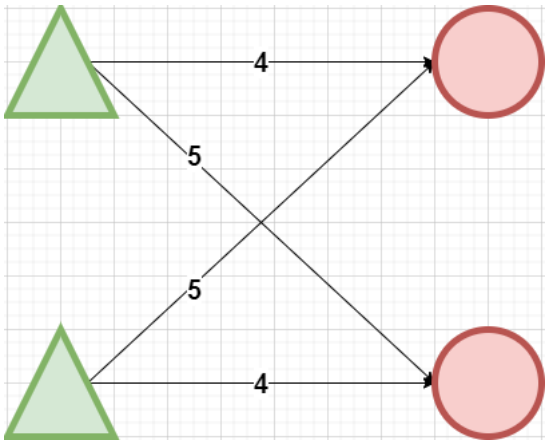
3 Přímé trajektorie

Dalším přístupem k plánování přeletů pro roje dronů je použití přímých trajektorií. V tomto případě jsou jednotlivé trajektorie reprezentovány úsečkou, která spojuje startovní a cílovou pozici. Aby bylo zabráněno vzniku křížení, je nutné zajistit, aby byly vhodně zvoleny páry pozic start-cíl [1]. Značnou výhodou této plánovací metody oproti metodě potenciálových polí je ta, že zde není potřeba simulovat celý průlet prostředím, ale pouze vhodně zvolit již zmíněné páry pozic.

Nedostatkem této metody je její praktická realizovatelnost. V rámci této metody je ošetřeno křížení letových tras, avšak nikoliv již blízký průlet, a tudíž nelze zajistit vzájemné bezpečné vzdálenosti mezi jednotlivými drony. Z tohoto důvodu v této kapitole navrhuji modifikaci této metody, která spočívá v přidání časového zpoždění přeletu, díky čemuž je možné zajistit požadované bezpečné rozestupy, a tím i umožnit praktickou realizovatelnost této metody.

3.1 Přiřazení pozic

Samotný proces plánování přeletů se skládá z několika kroků. První krok spočívá ve vhodném zvolení párů pozic v původní a cílové množině uskupení. Tímto jsou definovány úsečky v prostoru, které budou tvořit samotnou trajektorii pro jednotlivé drony. Cílem tohoto kroku je vhodné přiřazení cílových pozic dronům s ohledem na výpočetní náročnost, délku jednotlivých trajektorií a eliminaci neřešitelných stavů.



Obr. 3.1: Příklad formace s vyznačenými vzdálenostmi

Hlavní komplikací je zde samotný počet možných kombinací, které je možno získat. Předpokládejme, že máme shodně velké množiny startovacích i cílových pozic o velikosti N. Potom je možno pro první startovací pozici zvolit jednu z N pozic

cílových, pro následující N-1 a tímto způsobem pokračujeme, až poslední startovací pozici bude možné přiřadit pouze jednu cílovou pozici. Tímto tedy nastává N! možných kombinací, jak je možno tyto trajektorie naplánovat. Jelikož předpokládáme použití pro roj, který bude čítat desítky dronů, není tedy zvážení všech možných kombinací v rozumném časovém horizontu výpočetně možné.

Z výše popsaného důvodu je vhodné použít řešení z problematiky přiřazení součtem. Zde lze problém reprezentovat pomocí čtvercové váhové matice, kde je cílem k jednotlivým řádkům přiřadit sloupce tak, aby součet prvků, určených zvolenou dvojicí řádku a sloupce, byl co nejmenší. Každý prvek matice představuje váhu použití dané dvojice řádek - sloupec [6]. Pokud bychom takto chtěli reprezentovat tento problém s volbou vhodných cílových pozic pro jednotlivé drony, tak by každý řádek matice odpovídal jedné výchozí pozici, každý sloupec cílové pozici a jednotlivé prvky vzdálenostem mezi danými pozicemi. Cílem je potom nalezení takových párů pozic, kde bude součet všech délek nejmenší. Matice pro formaci, vyobrazenou na obrázku 3.1, by vypadala následovně:

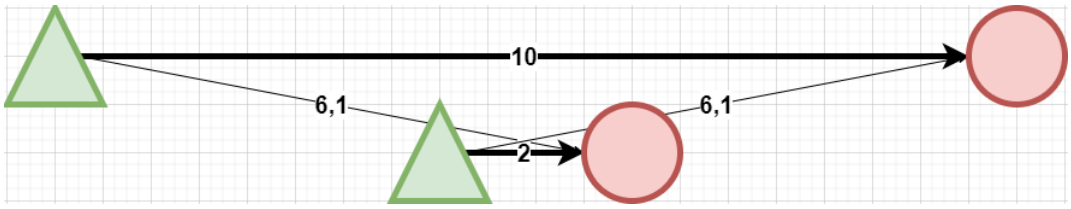
$$C = \begin{pmatrix} 4 & 5 \\ 5 & 4 \end{pmatrix} \tag{3.1}$$

Z této matice lze jednoznačně říci, že optimální přiřazení, z hlediska celkového součtu vzdáleností, nastane pro kombinaci (1,1), (2,2), pro které je celkový součet vzdáleností roven 8.

3.1.1 Důsledky minimalizace součtu vzdáleností

Použití kritéria pro nejmenší součet vzdáleností přináší jisté důsledky na to, jaké trajektorie jsou získány tímto procesem. Jedním z nich je, že díky tomuto postupu je možné nalézt takové dvojice pozic, že během přeletu nedojde ke křížení trajektorií. K tomuto chování dochází díky tomu, že součet dvou přímých tras je vždy menší než součet tras, které se kříží [1]. Tento jev lze ilustrovat na výše uvedeném příkladu, kde právě nejmenší součet délek trajektorií získáme, když nedojde ke křížení.

Nicméně zajištění tras bez křížení není dostatečné pro zajištění bezpečnosti. Pro zajištění bezkolizního přeletu je nutné vzít v úvahu i fyzické rozměry samotných dronů a také potřebný rozestup mezi nimi, aby se předešlo vzájemné kolizi a taktéž aby byla zohledněna skutečnost, že určení naprosto přesné polohy reálného dronu není možné a vždy je zatížena jistou nepřesností naváděcího systému. Z tohoto důvodu není možné použít tento plánovací krok samostatně a je zde potřeba dalšího stupně volnosti pro vyřešení právě těchto komplikací, což je v rámci tohoto plánovacího procesu zajištěno pomocí časového odložení startu přeletu, jež představuje mnou navrhnutou modifikaci této plánovací metody.

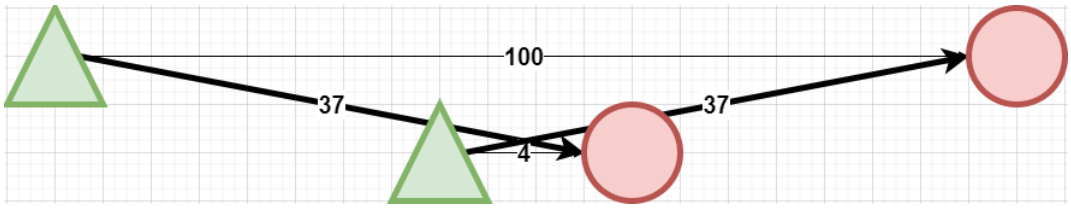


Obr. 3.2: Příklad formace se vznikem neřešitelného stavu

Dalším nedostatkem použití tohoto přístupu je možnost vzniku situací, ve kterých nelze naplánovat bezkolizní přelet ani s odložením startu. Příkladem takového stavu mohou být trajektorie znázorněné na obrázku 3.2, kde by váhová matice vypadala následovně:

$$C = \begin{pmatrix} 10 & 6.1 \\ 6.1 & 2 \end{pmatrix} \tag{3.2}$$

Na základě vzdáleností dojde k přiřazení (1,1), (2,2), jelikož je v tomto případě součet vzdáleností 12, zatímco v druhém případě by byl roven 12,2. Pokud bychom chtěli zachovat minimální vzdálenost dvou jednotek délky během celého letu, tak se nám to u první trajektorie nepodaří, jelikož ji v cestě bude vždy překážet dron z druhé trajektorie, nezávisle na tom, jak bude jejich start vzájemně odložen.



Obr. 3.3: Optimální trajektorie pro váhy rovny druhé mocnině vzdáleností

Tento problém je možno vyřešit úpravou vah tak, že bude použita druhá mocnina vzdálenosti, jak je zobrazeno na obrázku 3.3. Odpovídající váhová matice je pak následující:

$$C = \begin{pmatrix} 100 & 37 \\ 37 & 4 \end{pmatrix} \tag{3.3}$$

Díky této volbě vah budou zvoleny kombinace (1,2), (2,1) se součtem 74, oproti 104, což by byl výsledek použitím původní kombinace. Celková délka tras se tímto zvýšila, avšak díky této metodě je nyní největší délka trajektorie 6,1 délkových jednotek v porovnání s původními 10. Jelikož jsou trasy vykonávány paralelně, došlo tímto i ke zkrácení celkového času na přelet, což je dalším přínosem použití této modifikace.

V rámci implementace této plánovací metody je využito knihovní funkce z důvodu vysoké optimalizace [4]. Výsledné provedení této operace se pohybuje řádově ve vyšších stovkách milisekund pro matici o rozměrech 100 x 100 [5].

Nevýhodou přístupu minimalizace součtu délek trajektorií je možnost vzniku více trajektorií, u kterých dojde ke křížení. Z tohoto důvodu je potřeba implementovat mechanizmy, které dokáží tato křížení identifikovat a zajistit, aby během přeletu nedošlo ke kolizi.

3.2 Detekce křížení

Během plánovacího procesu mohou vzniknout letové trajektorie, pro které by došlo k příliš blízkému přiblížení dvou dronů, ne-li až ke kolizi. Z tohoto důvodu je potřeba identifikovat, zda se dvě trajektorie nenacházejí příliš blízko sebe a v případě, že tomu tak je, zjistit, pro jakou část trajektorie je vzájemná vzdálenost menší, než je bezpečný limit. Tento limit je zadáván uživatelem a jeho velikost se odvíjí od okolností daného přeletu, zejména tedy od rychlosti a fyzických rozměrů jednotlivých dronů. Detekce křížení je tedy první částí modifikace metody přímých trajektorií, která ve výsledku umožní zmíněnou metodu prakticky realizovat.

Tato detekce křížení je provedena vůči veškerým ostatním trajektoriím a tudíž může mít jedna trajektorie i více nebezpečných úseků s jinými trajektoriemi. V případě, že se po celé délce trajektorie nenachází žádný úsek, který by potenciálně mohl představovat nebezpečí, se tato trajektorie považuje za finální. V opačném případě je potřeba určit, zda není potřeba odložit start přeletu a popřípadě o jak velké odložení se jedná.

3.2.1 Hledání nejbližších bodů křížení

Při řešení tohoto problému je možné uvažovat, že jednotlivé trajektorie jsou součástí nekonečné přímky v prostoru, určené pomocí dvou bodů, které představují výchozí pozici a pozici cílovou. Na každé z těchto přímek se následně snažíme nalézt bod, jehož vzdálenost od druhé přímky je nejmenší.

Tyto body lze nalézt pomocí následujících výpočtů [7]:

$$\vec{u} = P_{1konec} - P_{1start} \tag{3.4}$$

$$\vec{v} = P_{2konec} - P_{2start} \tag{3.5}$$

$$\vec{p} = P_{2start} - P_{1start} \tag{3.6}$$

$$\vec{n} = \vec{v} \times \vec{u} \tag{3.7}$$

$$\vec{r} = \vec{p} \times \frac{\vec{n}}{|\vec{n}|^2} \tag{3.8}$$

$$t_1 = \vec{r} \cdot \vec{v} \tag{3.9}$$

$$t_2 = \vec{r} \cdot \vec{u} \tag{3.10}$$

$$Q_1 = P_{1start} + \vec{u} \cdot t_1 \tag{3.11}$$

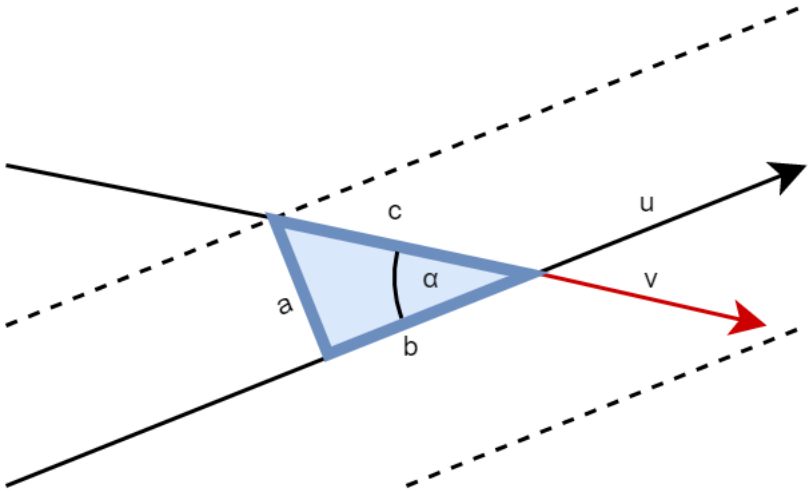
$$Q_2 = P_{2start} + \vec{v} \cdot t_2 \tag{3.12}$$

Kde body $\mathbf{P_1}$ a $\mathbf{P_2}$ odpovídají okrajům jednotlivých trajektorií a $\mathbf{Q_1}$ a $\mathbf{Q_2}$ bodům s nejmenší vzájemnou vzdáleností na příslušných přímkách.

3.2.2 Určení nebezpečného úseku

Dalším krokem při detekci křížení je získání vzdáleností od těchto bodů na jednotlivých přímkách, po kterou bude vzájemná vzdálenost menší, než je bezpečný limit. Nebezpečná zóna okolo samotné trajektorie je aproximována válcem, jehož poloměr je roven velikosti minimální bezpečné vzdálenosti a svou výškou překračuje příslušné konce trajektorie právě o velikost bezpečné vzdálenosti.

Pro získání nebezpečného úseku, nejdříve promítneme přímky do roviny kolmé na normálový vektor, který získáme vektorovým součinem vektorů, reprezentující samotné trajektorie. Tato situace je znázorněna na obrázku 3.4. Následně je potřeba spočítat novou velikost bezpečné vzdálenosti. U té došlo ke zmenšení v důsledku posunutí při promítání do roviny a dopočítáme ji, jako rameno pravoúhlého trojúhelníku, kde přeponu tvoří původní vzdálenost a délka jednoho ramena představuje velikost posunutí při promítání, což odpovídá kolmé vzdálenosti jednotlivých přímek.



Obr. 3.4: Detekce nebezpečného úseku

Nyní je potřeba dopočítat jednotlivé strany pravoúhlého trojúhelníku, vyobrazeného na obrázku 3.4. V tomto trojúhelníku známe velikost **a**, což odpovídá přepočítané bezpečnostní vzdálenosti a úhel **α**, který získáme pomocí skalárního součinu vektorů **u** a **v**.

$$\cos\alpha = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}|\cdot|\vec{v}|} \tag{3.13}$$

$$\sin\alpha = \sqrt{1-\cos^2\alpha} \tag{3.14}$$

$$c = \frac{a}{\sin\alpha} \tag{3.15}$$

$$b = c\cdot\cos\alpha \tag{3.16}$$

Závěrem je určen nebezpečný úsek pomocí podobnosti s trojúhelníkem, vyobrazeném na obrázku 3.4, kde jeho velikost určena na základě poměru vzdálenosti výchozího, případně cílového, bodu trajektorie s průsečíkem přímek, ke kterému je přičtena velikost upravené bezpečnostní vzdálenost, a strany **b**. Tento poměr je limitován tak, aby jeho velikost nepřekročila 1, jelikož je **b** určeno právě pro největší možný úsek **c**.

3.3 Určení přednosti letu a nebezpečných stavů

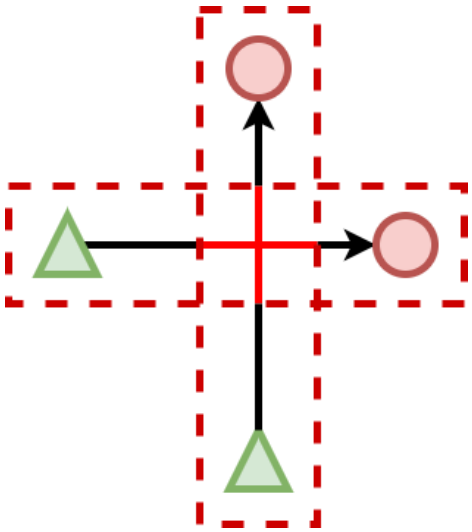
Provedením detekce křížení je možné zjistit, u jakých trajektorií existuje nebezpečí, které vyplývá z přílišného přiblížení. Pro tyto trajektorie je nutné zajistit, aby při plánování s tímto nebezpečím počítáno a aby bylo náležitě ošetřeno. Z tohoto důvodu

plánovací proces umožňuje určit odložení startů o potřebný časový interval a také zajistit pořadí, v jakém drony do nebezpečného úseku vletí.

Další krok plánovacího procesu závisí podle toho, na jaké části trajektorie se nebezpečný úsek nachází. Podle polohy tohoto úseku může dojít k následujícím situacím:

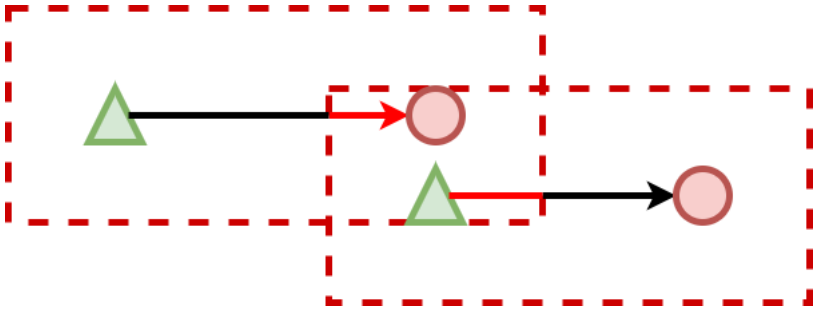
1. Samotná trajektorie pokračuje z obou stran i mimo tento úsek
2. Trajektorie pouze začíná, případně končí, uvnitř nebezpečného úseku
3. Celá trajektorie je součástí nebezpečného úseku

Pokud obě trajektorie pokračují i mimo nebezpečný úsek, tak je zapotřebí zjistit oba drony nevletěly do této oblasti ve stejný okamžik. V tomto případě nezáleží na pořadí, v jakém drony do oblasti vstoupí. Důležité je najít takovou velikost odložení, pro které by právě tato situace nastala, a zajistit, aby výsledné odložení bylo s jistým posunutím od něj vzdáleno.



Obr. 3.5: Ilustrace křížení trajektorií

Poslední případ nastane, pokud se v nebezpečném úseku nachází začátek nebo konec libovolné z trajektorií. V tomto případě je nutné zajistit, aby jeden dron neohrožoval dron druhý během čekání na start přeletu, případně během čekání po jeho dokončení. Postup je v tomto případě obdobný, jako v předchozím případě, a to, že je nejdříve zjištěno takové odložení, pro které by k ohrožení došlo a následně je určeno pořadí, které je během plánování nutné dodržet. Konkrétně tedy, pokud se počátek trajektorie prvního dronu nachází v nebezpečném úseku, tak je možné naplánovat odložení druhého dronu až po tom prvním a zároveň nemůže být odložení druhého dronu menší, než stanovené kritické odložení. Takto je zajištěno, že první dron nebude blokovat cestu druhého dronu.



Obr. 3.6: Ilustrace trajektorií s okrajem v nebezpečném úseku

Případ, kdy by se celá trajektorie nacházela uvnitř nebezpečného úseku, může nastat pouze tehdy, pokud by došlo k chybě uživatele a jednotlivé výchozí nebo cílové pozice nebyly dostatečně navzájem vzdáleny. Během přiřazovacího procesu tento stav nemůže nastat, v důsledku použité metody, avšak je možné, že bude trajektorie takto identifikována, z důvodu použitého zjednodušení pro detekci nebezpečných úseků. Proto pokud tento případ nastane, je nedříve zjištěno zakázané odložení a následně je určena přednost tak, aby letěl dron, který je k místu křížení nejblíže, jako první.

3.4 Plánování odložení startů

Závěrečnou částí navržené modifikace plánovacího procesu je naplánování odložení startů přeletů tak, aby byly splněny požadavky z hlediska předností a vzájemných odložení. Pro jednodušší práci je vhodné reprezentovat jednotlivá odložení startů jako dráhu, kterou by po dobu tohoto odložení danou rychlostí urazil. Díky tomuto je možné přímo pracovat s dráhovými rozdíly přímo v místech křížení a zajistit tak, že budou potřebné bezpečné vzdálenosti dodrženy. Po ukončení celého plánovacího procesu budou tato odložení převedena dle nastavené rychlosti na čekací čas.

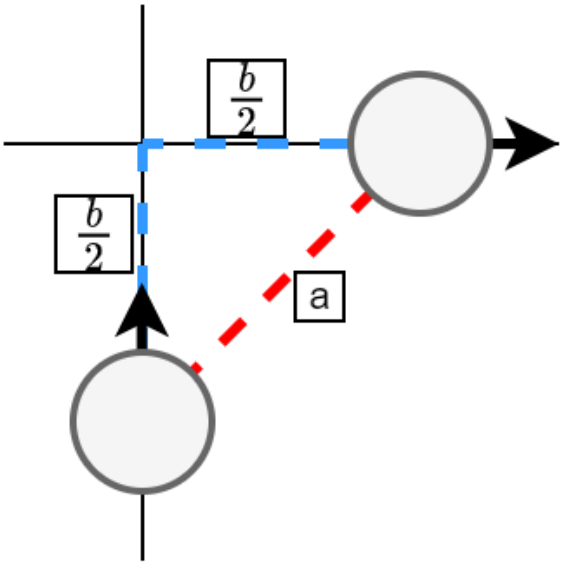
Plánování odložení startů se pouze týká těch trajektorií, u kterých se jejich část nachází v nebezpečném úseku s nějakými dalšími trajektoriemi. Ostatní trajektorie jsou již považované za naplánované a není důvod je v tomto kroku brát v potaz.

3.4.1 Odložení bez přednosti

První částí tohoto plánovacího kroku je naplánování odložení trajektoriím, které nemají požadavek na přednost vůči jiným trajektoriím, jelikož jim nic nebrání ve zvolení odložení. Než dojde k samotnému přidělování odložení, jsou tyto trajektorie

seřazeny dle velikosti od nejdelší po nejkratší. Díky tomuto je menší pravděpodobnost, že dojde ke zpoždění u delších trajektorií a větší u kratších. Tato operace pak přispívá k nižšímu celkovému času na přelet.

Samotná volba odložení probíhá tak, že je nejdříve vytvořen seznam intervalů odložení, ve kterých se výsledné odložení nesmí nacházet. Tyto intervaly jsou vytvořeny jen pro trajektorie, které již mají odložení naplánováno a představují tak omezení. S pomocí takto vytvořeného seznamu hledáme nejmenší možné odložení, které není součástí žádného intervalu. Samotný interval pak vznikne tak, že je sečteno samotné naplánované odložení se zakázaným odložením. Tímto získáme střed tohoto intervalu a jeho spodní a horní hranici získáme odečtením a přičtením bezpečného odstupu. Ten je zvolen tak, aby byl zachován bezpečný rozestup i pro nejvíce nepříznivou situaci, kterou je pravoúhlé křížení v rovině.



Obr. 3.7: Pravoúhlé křížení trajektorií

V tomto případě, pokud by byl zvolen střed intervalu jako odložení, by se drony přesně potkaly v místě křížení. Našim cílem je zvolení takového bezpečného odstupu, kdy nebude jejich vzájemná vzdálenost menší, než je uživatelem zadaný bezpečnostní limit. Ten získáme tak, že danou situaci popíšeme pomocí pravoúhlého trojúhelníku, kde jsou jeho vrcholy tvořeny drony a průsečíkem jejich trajektorií, jak je vyobrazeno na obrázku 3.7. Pokud se jeden dron již nachází za průsečíkem a druhý před, tak součet délek ramen takto vzniklého trojúhelníku bude odpovídat našemu hledanému odstupu. Nyní stačí pouze najít takovou situaci, kdy bude vzájemná vzdálenost dronů nejmenší, což nastane, pokud tento trojúhelník bude rovnoramenný. V tento moment budou délky ramen odpovídat polovině hledaného

odstupu. Za délku přepony potom zvolíme minimální potřebný rozestup a získáme tak vše potřebné k výpočtu odstupu:

$$a = \sqrt{2} \cdot \frac{b}{2} \tag{3.17}$$

$$b = \sqrt{2} \cdot a \tag{3.18}$$

Kde **a** představuje minimální vzdálenost mezi drony a **b** představuje výsledný odstup.

3.4.2 Odložení s předností

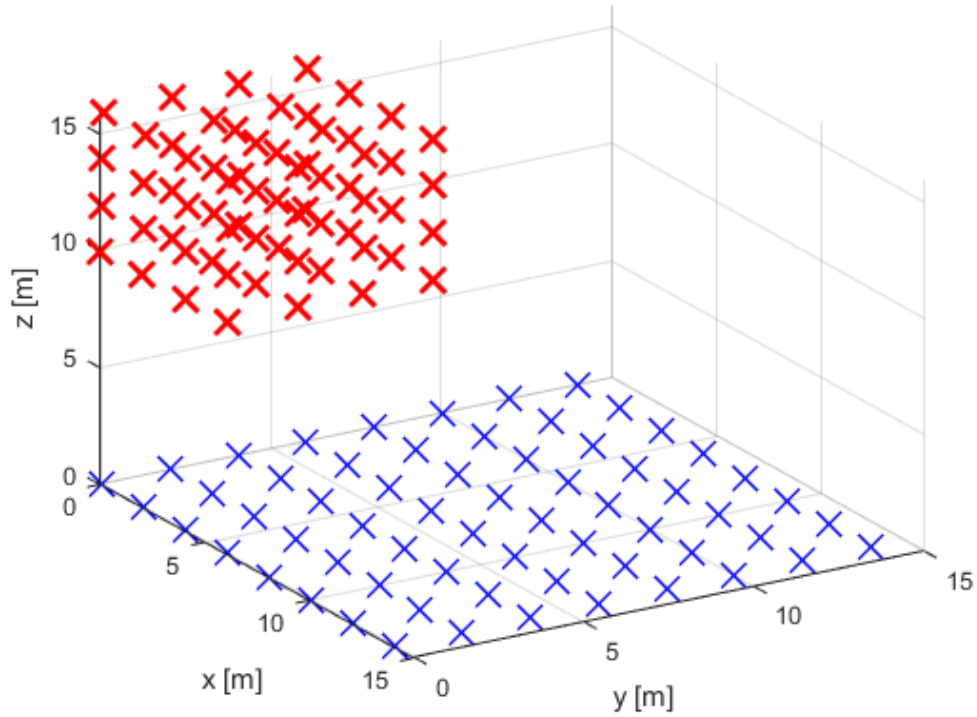
Plánování odložení, která mají závislost na naplánování jiných, jsou provedena jako poslední. Tímto již jen zbývá zajistit správné pořadí naplánování v rámci jednotlivých trajektorií s požadavky. Tyto trajektorie jsou umístěny ve frontě, ze které jsou postupně odebírány na zpracování.

Samotné zpracování probíhá tak, že je nejdříve u dané trajektorie ověřeno, zdali je u veškerých trajektorií, na kterých má tato trajektorie závislost, naplánované odložení. Pokud tomu tak je, provede se naplánování, obdobným způsobem, jako u ostatních trajektorií, jen s tím rozdílem, že nyní hledáme nejmenší možné odložení, avšak které je zároveň větší, než největší odložení ze všech závislostí. Díky tomuto je zajištěno správné pořadí průchodu nebezpečnými úseky.

V případě, kdy nejsou veškerá odložení, na kterých daná trajektorie závisí, naplánována, je tato trajektorie zařazena zpět do fronty na určité místo tak, aby se před ní nacházely veškeré potřebné trajektorie, u kterých ještě není odložení naplánováno.

4 Srovnání plánovacích algoritmů

Pro zvolení vhodného plánovacího algoritmu byla provedena implementace metody přímých trajektorií (PT) a umělých potenciálových polí (UPP) v jazyce Python a následně bylo provedeno naplánování přeletu pro různý počet dronů. Samotné porovnání je zaměřeno na dva parametry. První parametr je časová náročnost provedení naplánování, což odpovídá času, po který bude muset uživatel čekat, než se provede naplánování. Druhý parametr je časová náročnost provedení, který udává množství času potřebnému k přeletu všech dronů do cílové pozice.



Obr. 4.1: Testovací formace pro 64 dronů

Porovnání bylo provedeno na počítači s procesorem Intel i5-8250U a 8GB RAM. Samotná implementace byla provedena v jazyce Python ve verzi 3.9.5 s moduly numpy ve verzi 1.24.2 a scipy ve verzi 1.9.3. Přelet probíhal z rovné plochy do kostky 10 m nad vzletovou plochou, jak je pro případ s 64 drony zobrazeno na obrázku 4.1. Vzájemné rozestupy startovních i cílových pozic byly 2 m a bezpečnostní limit byl nastaven na 1 m a požadovaná rychlost přeletu činila 5 m/s. Metoda potenciálových polí byla použita s následujícími parametry: $K_C = 20$ $K_p = 100$ a $\rho = 1$.

Tab. 4.1: Srovnání časové náročnosti plánování

Počet dronů [-]	64	125	216	343	512
Čas plánování UPP [s]	1,6	8,2	25,1	75,3	196,7
Čas plánování PT [s]	0,5	1,8	6,1	14,6	30,8

Tab. 4.2: Srovnání časové náročnosti provedení

Počet dronů [-]	64	125	216	343	512
Čas letu UPP [s]	5,3	7,9	8,6	10,5	12,5
Čas letu PT [s]	3,9	4,8	5,9	7,4	9,4

Z dosažených výsledků simulace je patrné, že se zvětšujícím se počtem dronů značně roste výpočetní čas pro provedení samotného naplánování, jak je zobrazeno v tabulce 4.1. Výsledky obou metod v tomto případě jsou dostatečně uspokojující pro využití k plánování přeletů pro menší letku dronů. Pro použití pro větší letku by avšak byla potřeba jistá optimalizace implementací. Konkrétně by implementace v kompilovaném jazyce mohla přinést zásadní zrychlení.

Při porovnání výsledků druhého sledovaného parametru, kterým je celkový čas letu, tak je možné si všimnout, že metoda přímých trajektorií vychází v tomto ohledu lépe, jak je možno určit z tabulky 4.2. Tento výsledek je důsledkem chování při navádění pomocí umělých potenciálových polí, kde dochází ke zpomalení, pokud se dva naváděné objekty příliš přiblíží k sobě, kdežto v případě metody přímých trajektorií je tento vyhýbací mechanismus efektivnější, jelikož se naváděný objekt pohybuje stálou maximální rychlostí a je opožděn pouze o nejmenší nutnou dobu.

Na základě dosažených výsledků je pro výslednou implementaci plánovacího doplňku zvolena metoda přímých trajektorií, jelikož vyžaduje menší čas pro provedení naplánování, což je důsledek použití optimalizované knihovní funkce pro přiřazení pozic. Mimo to je také celková doba přeletu kratší, což umožňuje efektivněji využít letový čas reálné letky.

5 Implementace rozšíření

Výsledná plánovací metoda byla implementována společně s dalšími funkcemi ve formě rozšíření 3D modelovacího programu jménem Blender. Jedná se o bezplatný nástroj, který umožňuje jednoduše pracovat s objekty v prostoru a taktéž vytvářet animace. Hlavní výhodou tohoto prostředí je možnost přidání dalších funkcí pomocí rozšíření. Tato rozšíření mají podobu skriptů v programovacím jazyce Python, kterým je umožněno získávat data a ovládat prostředí pomocí Blender API [8]. Právě tato otevřenost a jednoduchost byly hlavními důvody pro volbu tohoto prostředí.

Samotná implementace byla provedena pro Blender ve verzi 3.3.1. Dodatečně byl do prostředí doinstalován modul scpipy ve verzi 1.9.3, ze kterého je použita funkce pro přiřazení pozic na základě váhové matice *linear_sum_assignment*.

Výsledné rozšíření je tvořeno sadou operátorů. Operátor vykoná jistou akci v momentě, kdy je spuštěn. Tyto akce mohou být například vytvoření nového objektu nebo provedení jisté transformace. Tyto operátory taktéž umožňují vytvořit vyskakovací okno, pomocí kterého lze získat od uživatele jisté parametry před provedením samotné akce.

Operátor lze definovat pomocí třídy, která bude dědit vlastnosti z výchozího typu pro operátor. V této třídě je možné definovat následující vlastnosti operátoru:

- Kontextovou nápovědu
- ID operátoru
- Název operátoru
- Parametry operátoru

Chování operátoru je možné následně definovat uvnitř metody **execute**, která je volána při provádění daného operátoru. Pokud operátor vyžaduje nějaké parametry, je možné v metodě **invoke**, která se volá při spuštění operátoru, definovat vyvolání vyskakovacího okna, které vyzve uživatele k zadání potřebných parametrů. Příklad definice operátoru je zobrazen na výpisu 5.1.

Výpis 5.1: Definice operátoru

```
import bpy

class NovyOperator(bpy.types.Operator):
    """Kontextová nápověda k operátoru"""
    # Unikatní ID pro tento operátor
    bl_idname = "unikatni_id"
    # Název operátoru, pomocí kterého jej lze vyhledat
    bl_label = "Název␣operátoru"
    # Povolené akce pro tento operátor
    bl_options = {'REGISTER', 'UNDO'}

    # příklad definice parametru
    vzdalenost: bpy.props.FloatProperty(
        name="Minimální␣vzdálenost", default=5.0,
        min=1.0, max=10.0)
```

Závěrem je potřeba operátor zaregistrovat, což umožní jeho použití, a přidat do nabídky možných akcí funkci, která nám umožní jej následně spustit. Příklad zaregistrování operátoru přidání možnosti jej vyvolat je zobrazen na výpisu 5.2.

Výpis 5.2: Registrace operátoru

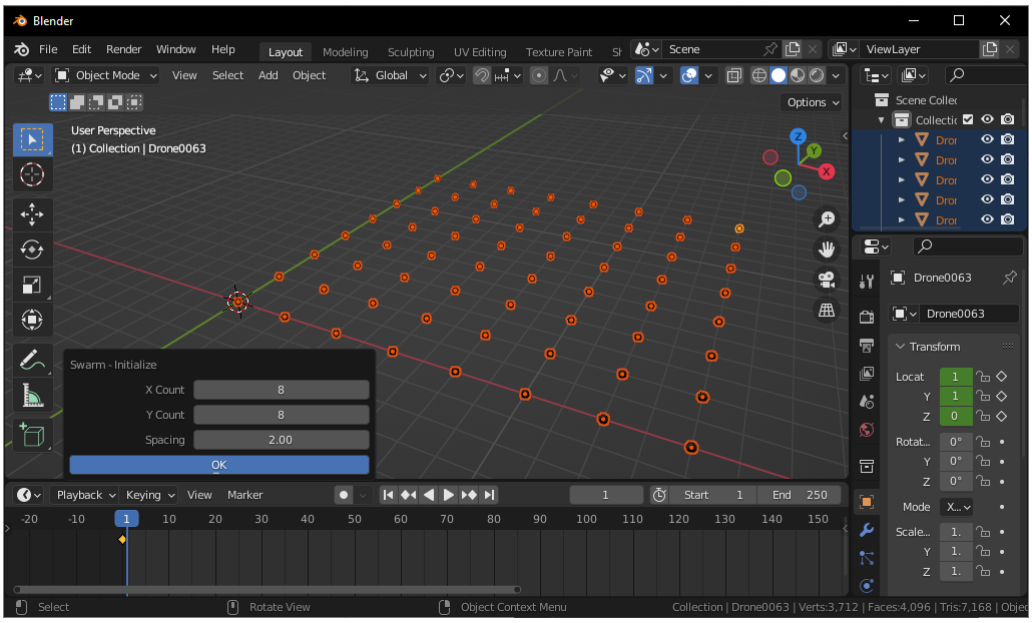
```
# Funkce pro volání operátoru
def operator_menu(self, context):
    self.layout.operator("unikatni_id")

# Registarce operátoru
bpy.utils.register_class(NovyOperator)
# Přidání operátoru do nabídky
bpy.types.VIEW3D_MT_object.append(operator_menu)
```

5.1 Vytvoření letky

První operátor se zabývá vytvořením reprezentace letky dronů. Výchozí formací je obdélník, který je vytvořen dle parametrů, zadanými uživatelem:

- Počet dronů v ose X
- Počet dronů v ose Y
- Rozestup mezi jednotlivými drony

Při provádění akce tohoto operátoru je nejdříve vytvořena reprezentace dronu ve tvaru koule. Následně je mu nastavena barva na černou a pozice na příslušné místo v obdélníku. Závěrem je vytvořen klíčový snímek, který slouží k vytváření animací.

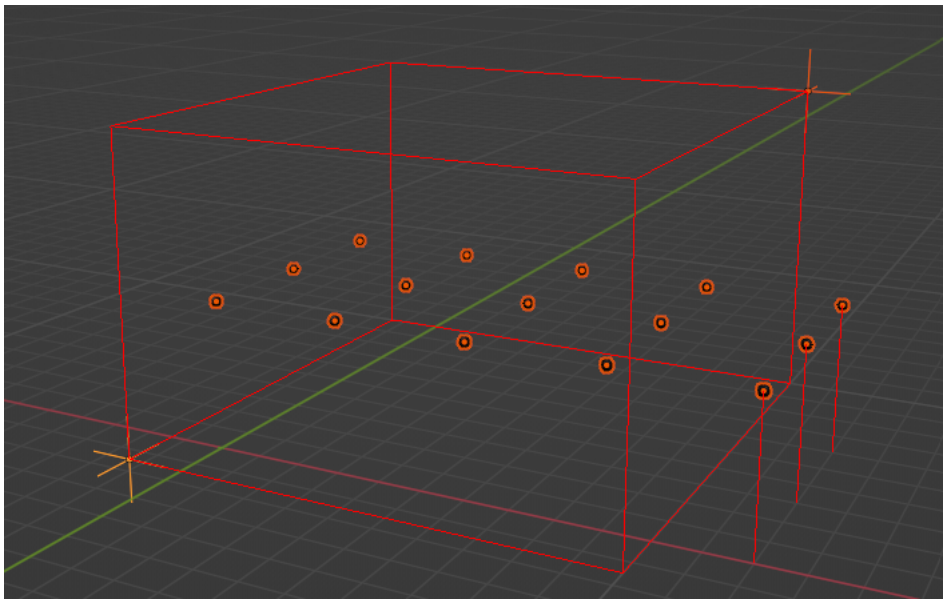


Obr. 5.1: Ukázka použití operátoru pro vytvoření letky

Obrázek 5.1 zobrazuje dialogové okno a vytvořenou reprezentaci letky společně s příslušným počátečním klíčovým snímkem.

5.2 Definice letového prostoru

Definovat letový prostor je možné s využitím příslušného operátoru, jež vytvoří dva body v prostoru, se kterými je možno volně pohybovat. Tyto dva body definují kvádr, představující letový prostor.

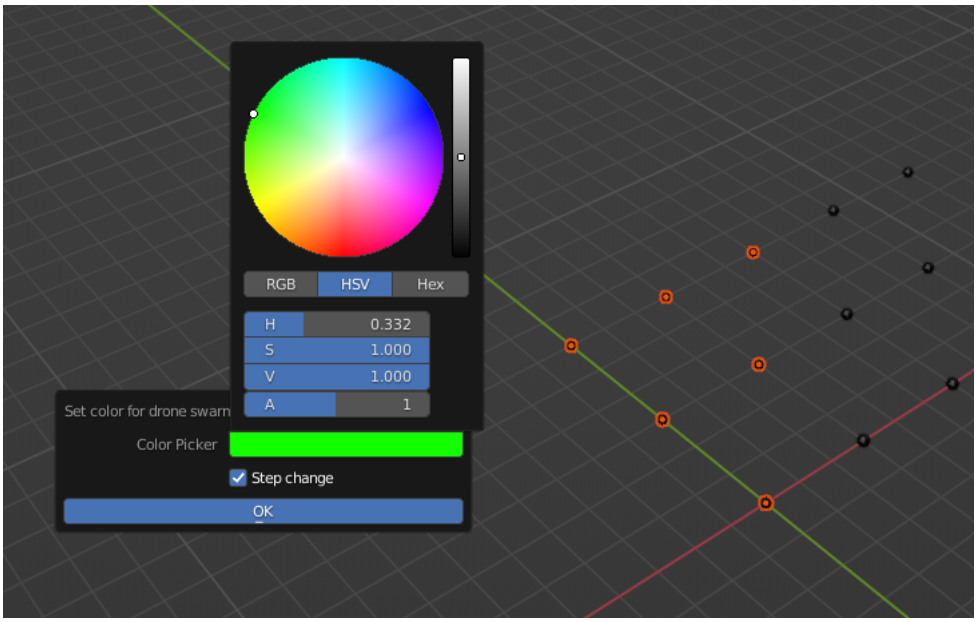


Obr. 5.2: Ukázka definice letového prostoru

Definovaný letový prostor je následně vizualizován pomocí hran v pracovním okně. Zároveň je i graficky znázorněno, pokud nějaký z dronů tento vyznačený prostor opustí, jak je možno vidět na obrázku 5.2.

5.3 Změna vlastností letky

Další operátor umožňuje úpravu vlastností dronů, konkrétně se jedná o jejich barvu. Uživatel nejdříve označí žádané drony a následně zavolá tento operátor. Ve vyskakovacím okně si může zvolit žádanou barvu a zda-li se má změnit skokově, či plynule od přechozího klíčového snímku. Tento operátor rovněž vytvoří klíčový snímek, obsahující zvolené barvy, pro dané drony. V případě, že je zvolena možnost skokové změny, vytvoří se dodatečný klíčový snímek těsně před tím prvním s původním nastavením barev. Díky tomuto dojde ke změně barvy z původní na novou během jednoho snímku a tudíž bude tato změna skoková.



Obr. 5.3: Ukázka nastavení barvy

Obrázek 5.3 zobrazuje příklad použití daného doplňku pro změnu barvy pomocí dialogového okna.

5.4 Plánování přeletu mezi formacemi

Nejvíce komplexním operátorem je ten, jež zajišťuje plánování přeletů z jedné formace do druhé. Uvnitř tohoto operátoru je využíván plánovací algoritmus, který byl vybrán na základě porovnání.

Při použití tohoto operátoru je předpokládáno, že již uživatel provedl vytvoření letky příslušným operátorem a zároveň, že označil objekt, který představuje výslednou formaci. Po spuštění je uživatel vyzván k zadání minimální vzdálenost, která má být podél celého přeletu dodržena a také rychlost, jakou se mají jednotlivé drony pohybovat. Rovněž je uživateli umožněno zvolit, zda-li budou použity vrcholy, nebo plochy označeného 3D modelu, jako cílová formace.

Během provážení samotného operátoru jsou nejdříve dvě pole s pozicemi. První pole obsahuje pozice všech dostupných dronů, které budou sloužit jako zdrojová množina bodů, a to druhé pole bude zahrnovat pozice všech vrcholů či ploch zvoleného objektu, a tím bude tvořit množinu cílů. Následně je zkontrolováno, zda-li jsou počty prvků v jednotlivých polích shodné. V případě, že tomu tak není, jsou přebytečné pozice odstraněny z pole s více prvky.

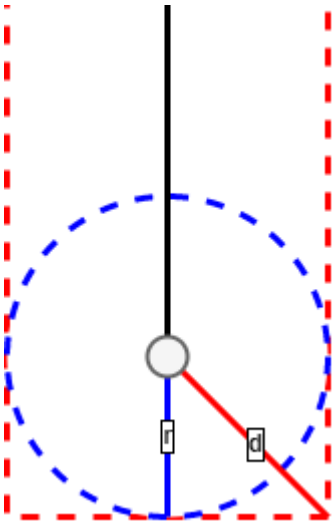
Obr. 5.4: Ukázka časové osy po naplánování přeletu

Po provedení naplánování získáme dvojice bodů, které představují samotnou trajektorii, a odložení, které je potřeba aplikovat. Takto získané odložení je převedeno na počet snímků, o který je klíčový snímek, značící start, vůči počátku přeletu posunut. Následně je také umístěn klíčový snímek s cílovou pozicí tak, aby jeho posunutí vůči startovacímu klíčovému snímku odpovídalo časovému intervalu, za jaký dron při dané rychlosti urazí vzdálenost mezi samotnými body. Výsledná časová osa je zobrazena na obrázku 5.4.

5.4.1 Nedostatek doplňku

Během závěrečného testování byl v tomto doplňku objeven nedostatek takový, že pokud jsou vzájemné pozice ve startovní či cílové množině navzájem příliš blízké bezpečnostnímu limitu, může nastat stav, kdy nebude provedeno naplánování korektně a tento bezpečnostní limit bude porušen.

Následně bylo zjištěno, že toto chování je důsledkem aproximace nebezpečné oblasti okolo letové trajektorie válcem, kde na jeho okrajích je vzdálenost od nejbližšího bodu trajektorie značně vyšší, než je potřebný limit.



Obr. 5.5: Ilustrace chyby aproximace

Obrázek 5.5 ilustruje rozdíl mezi aproximací, která je značena červeně, a skutečnou nebezpečnou zónou v krajní pozici, která je označena modře.

Na základě této situace lze určit poměr mezi největší možnou nebezpečnou vzdáleností plynoucí z aproximace a skutečnou požadovanou nebezpečnou vzdáleností:

$$\frac{d}{r} = \frac{r\sqrt{2}}{r} = \sqrt{2} \tag{5.1}$$

Největší možná uvažovaná bezpečná vzdálenost se může lišit přibližně o 42 % oproti požadované hodnotě. Z tohoto důvodu je pro korektní fungování zajistit, aby vzájemné rozestupy ve startovní a cílové množině nebyly menší, než je tato největší uvažovaná vzdálenost.

5.5 Export letových tras

Tento operátor se zabývá exportem letových tras pro další použití. Exportovaná data zahrnují pozice veškerých dronů společně s jejich aktuální barvou napříč všemi snímky animace. Součástí dat jsou rovněž údaje o počtu použitých dronů a počet snímků za sekundu, pro které byla animace vytvořena.

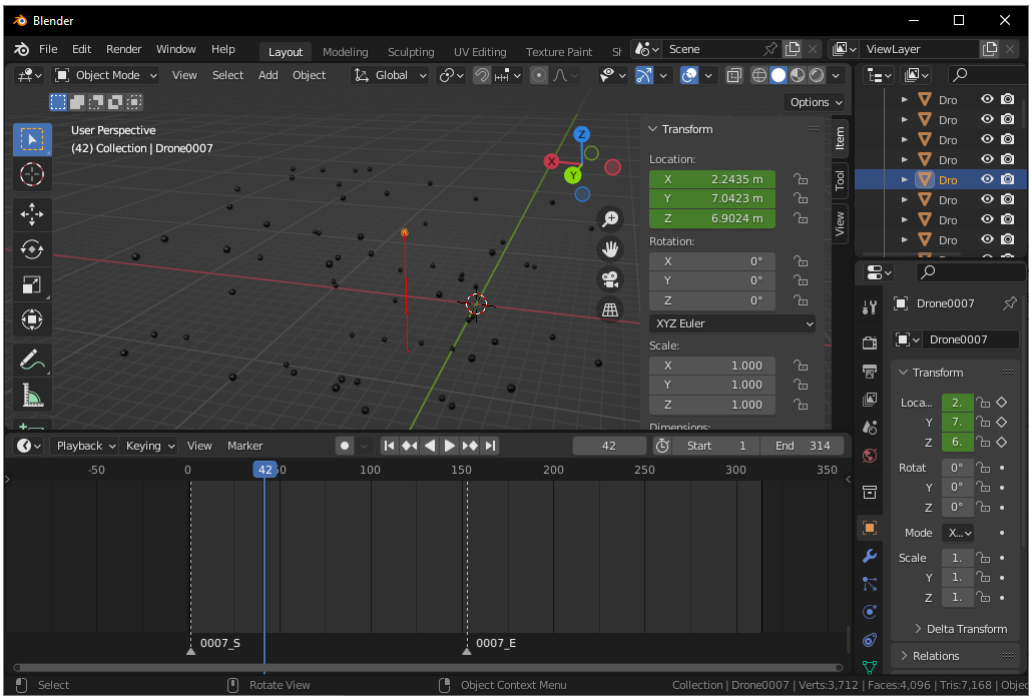
Po spuštění je vyvoláno okno, které umožní zvolit uživateli místo uložení výsledného souboru. Následně je pro každý snímek animace zjištěna barva a pozice dronu. K těmto datům je po provedení této akce přidána informace o počtu dronů společně s počtem snímků za sekundu. Závěrem jsou tato data uložena ve formátu JSON, což značně ulehčuje následnou práci s tímto souborem. Příklad podoby výsledného souboru je znázorněn na výpisu 5.3.

Výpis 5.3: Příklad exportovaných dat

```
{
    "fps": 24,
    "data": {
        "1": {
            "position": [
                [22.5, 0.0, 0.0],
                [22.5, 0.0, 0.1]
            ],
            "color": [
                [0, 0, 0],
                [10, 10, 10]
            ]
        }
    },
    "drone_count": 1
}
```

5.6 Kontrola rychlostí

Provedení kontroly rychlosti pohybu je možné s využitím tohoto operátoru, pomocí kterého lze detekovat úseky, kdy byla překročena horizontální nebo vertikální rychlost. Rozdělení rychlosti na tyto dvě složky umožňuje lépe ověřit, zda-li je daná mise realizovatelná, jelikož samotný dron má maximální rychlosti v těchto směrech rozdílné. Tato skutečnost plyne z toho, že je část vertikální složky tahu využívána pro kompenzaci tíhové síly a tudíž neumožňuje v této ose takovou manévrovatelnost, jako v té horizontální.

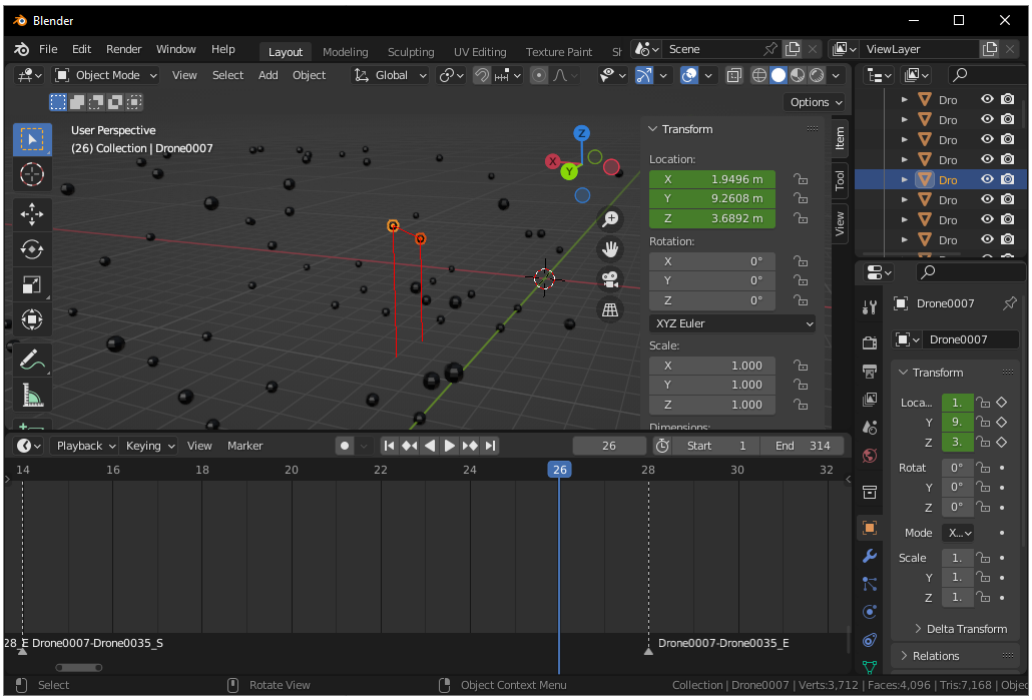


Obr. 5.6: Příklad signalizace porušení stanovené rychlosti

Při překročení rychlosti je vytvořena na časové ose značka s příslušným číslem dronu a také je tato skutečnost signalizována pomocí červeného sloupce, jak je možné vidět na obrázku 5.6.

5.7 Kontrola vzdáleností

Poslední operátor zajišťuje kontrolu vzájemných vzdáleností mezi jednotlivými drony. Tato kontrola je provedena pro každý snímek mise a pomáhá tak detekovat chybu lidského faktoru.

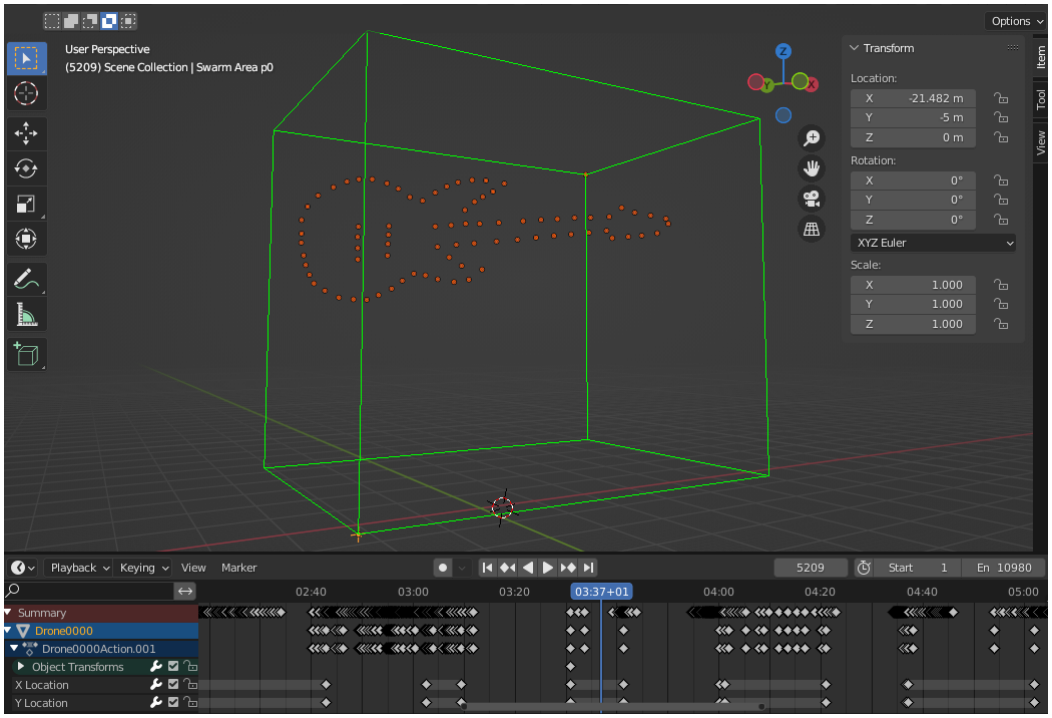


Obr. 5.7: Příklad signalizace porušení bezpečné vzdálenosti

Pokud je detekován úsek, kde je vzájemná vzdálenost libovolných dronů menší než specifikovaný limit, vytvoří se na časové ose značka společně s identifikátorem jednotlivých dronů. Toto porušení je dále také signalizováno červenými liniemi, jak je zobrazeno na obrázku 5.7.

6 Praktický test

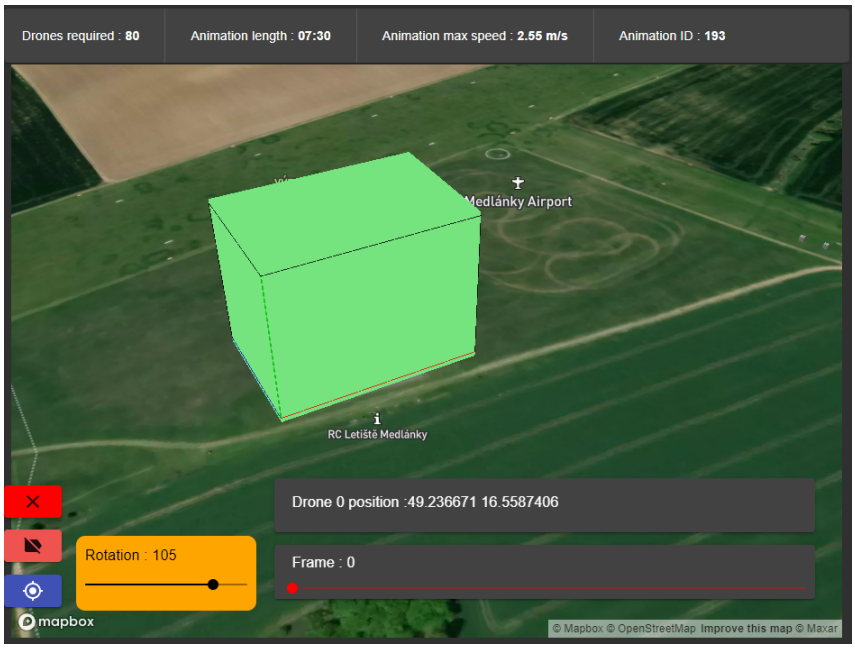
Za pomocí implementovaných doplňků do prostředí Blender byla vytvořena letová mise, která realizuje několik přeletů mezi formacemi v prostoru. Správnost dodržení požadovaných rozestupů a rychlostí bylo zkontrolováno pomocí příslušných doplňků. Závěrem byla pak tato mise exportována pro další použití.



Obr. 6.1: Snímek z prostředí Blender

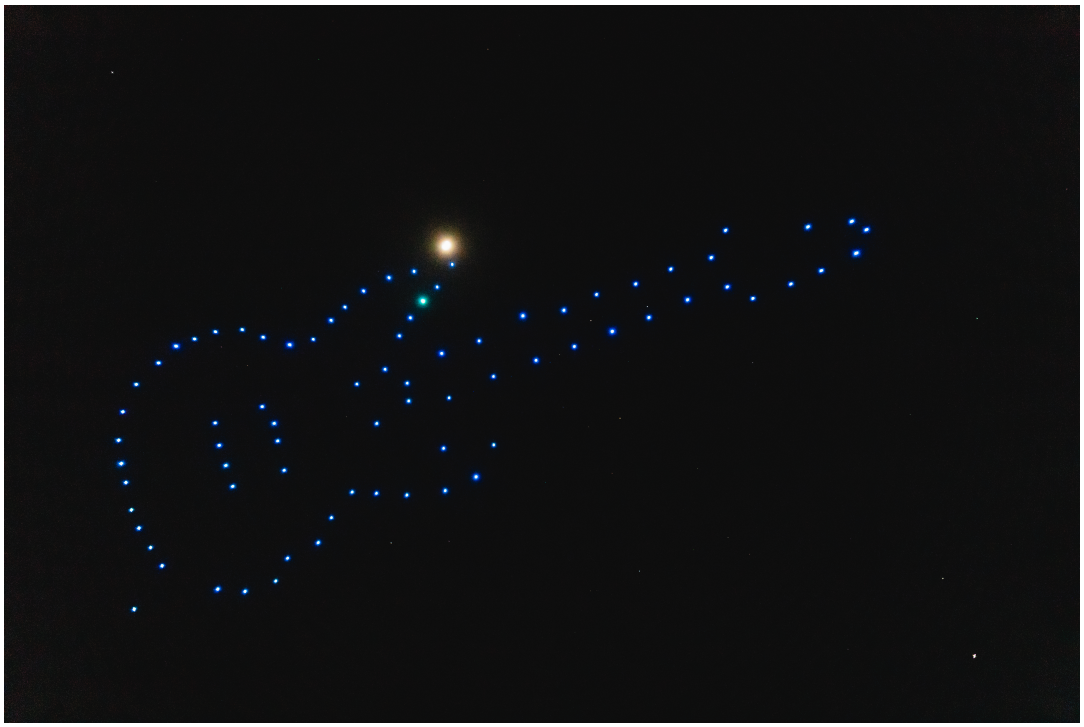
Obrázek 6.1 zobrazuje okamžik z výsledné mise společně s výsekem časové osy pro jeden dron.

Před samotným letem byla vygenerovaná mise načtena do řídící aplikace, která byla realizována mimo rámec této bakalářské práce a pomocí které byla nastavena požadovaná pozice a orientace v reálném světě. Po tomto umístění aplikace vygenerovala příslušné letové trasy pro jednotlivé drony a následně je nahrála do jednodeskového počítače, umístěného na dronu, který zajišťuje provádění mise a předávání povelů řídící jednotce.



Obr. 6.2: Snímek z řídící aplikace

Na obrázku 6.2 je snímek z části řídícího prostředí pro interaktivní volbu umístění výsledné animace v reálném světě.



Obr. 6.3: Snímek reálného testovacího letu

Závěrem byl na zvoleném místě proveden reálný let s letkou čítající 80 dronů dle jednotlivých naplánovaných letových tras. Samotný let je možné považovat za úspěšný a při jeho provedení došlo jen k drobným komplikacím. Konkrétně se nepodařilo jednomu dronu vzlétnout v důsledku chybné kalibrace a další dron kvůli chybné konfiguraci nepřijímal poziční korekce, a tudíž jeho let probíhal ve znatelně nižší výšce, což lze pozorovat na fotografii 6.3. Taktéž je možné si zde povšimnou chyby uživatele, která vznikla při plánování mise, kde jeden dron má nesprávnou zelenou barvu, oproti žádané modré.

Závěr

V rámci této bakalářské práce byl proveden návrh řešení pro plánování bezkolizních letových tras pro roj dronů. Tento algoritmus byl následně společně s dalšími funkcemi implementován ve formě doplňku modelovacího prostředí Blender. Závěrem byla funkčnost tohoto doplňku ověřena na letovém scénáři s využitím zapůjčené dronové letky.

Z počátku práce byla věnována pozornost dvěma přístupům vhodných k plánování letových tras pro roj dronů. První přístup využíval metodu umělých potenciálových polí, která pro správné fungování vyžadovala vhodné nastavení parametrů. Druhý přístup využíval přímé trajektorie a v porovnání s prvním přístupem se jevil jako výpočetně méně náročný, jelikož nebylo potřeba simulovat celou délku letu pro provedení plánování. Avšak značnou nevýhodou byla nemožnost zajistit bezpečné rozestupy mezi jednotlivými drony. Z tohoto důvodu jsem provedl návrh úpravy této metody přidáním možnosti odložit přelet, díky čemuž již tento zmíněný požadavek bylo možné splnit.

Po provedení implementace obou přístupů plánování bylo možné provést porovnání, kde byly sledovány dva parametry. První z nich byl čas potřebný k naplánování. Tento parametr se promítá do samotné použitelnosti realizovaného doplňku a také uživatelské přívětivosti. Dle mých předpokladů vyšla metoda přímých trajektorií jako časově méně náročnější, zejména pak v důsledku použití optimalizované knihovní funkce pro vhodnou volbu pozičních párů. Druhý sledovaný parametr byl celkový čas na provedení přeletu, který bylo potřeba sledovat z důvodu omezeného letového času celé mise, který se odvíjí od výdrže baterií samotných dronů. Zde se opět jevila metoda přímých trajektorií jako vhodnější, jelikož v případě umělých potenciálových polí docházelo k častému zpomalení pohybu dronů, pokud se ocitly v blízkém okolí ostatních dronů.

Se zvolenou metodou byla následně provedena integrace do modelovacího prostředí Blender společně s dalšími doplňky, které umožnily ověření požadovaných omezení na letový prostor, maximální rychlost pohybu a vzájemnou vzdálenost.

Závěr práce zahrnuje využití vytvořených doplňků pro naplánování letové mise a následně její reálné provedení na roji dronů. Nicméně zde byl objeven nedostatek realizovaného doplňku pro plánování přeletu, v jehož důsledku vzniká požadavek, aby nejmenší vzájemná vzdálenost mezi drony ve startovní či cílové množině byla přibližně o 42 % vyšší, než je stanovený bezpečnostní limit.

Z hlediska možného dalšího budoucího rozšíření se zde jistě nachází prostor k vhodnější optimalizaci implementace plánovací metody, a to zejména pak implementací v kompilovaném programovacím jazyce oproti dosavadní implementaci v interpretovaném jazyce Python, což by mohlo přinést výrazné zrychlení procesu

plánování, a tím i umožnit plánování tras pro větší roje v uživatelsky přívětivém časovém horizontu. Taktéž je vhodné při dalším vývoji upravit modelování nebezpečného úseku v okolí letových trajektorií, aby nedocházelo k hraničním situacím, a tím se i umožnilo vytvářet mise s vyšší prostorovou hustotou dronů.

Literatura

[1] SUI, Zezhi, Zhiqiang PU a Jianqiang YI. Optimal UAVs formation transformation strategy based on task assignment and Particle Swarm Optimization. In: 2017 IEEE International Conference on Mechatronics and Automation (ICMA) [online]. IEEE, 2017, 2017, s. 1804-1809 [cit. 2022-11-12]. ISBN 978-1-5090-6758-9. Dostupné z URL: <https://ieeexplore.ieee.org/document/8016091>.

[2] VADAKKEPAT, P., KAY CHEN TAN a WANG MING-LIANG. Evolutionary artificial potential fields and their application in real time robot path planning. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512) [online]. IEEE, 2000, s. 256-263 [cit. 2022-11-21]. ISBN 0-7803-6375-2. Dostupné z: <doi:10.1109/CEC.2000.870304>.

[3] SUN, Hang, Juntong QI, Chong WU a Mingming WANG. Path Planning for Dense Drone Formation Based on Modified Artificial Potential Fields. In: 2020 39th Chinese Control Conference (CCC) [online]. IEEE, 2020, 2020, s. 4658-4664 [cit. 2022-11-02]. ISBN 978-9-8815-6390-3. Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/9189345>.

[4] SciPy documentation [online]. [cit. 2023-04-17]. Dostupné z URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html>.

[5] CROUSE, David F. On implementing 2D rectangular assignment algorithms. IEEE Transactions on Aerospace and Electronic Systems [online]. 2016, 52(4), 1679-1696 [cit. 2023-04-17]. ISSN 0018-9251. Dostupné z: <doi:10.1109/TAES.2016.140952>.

[6] BURKARD, Rainer, Mauro DELL'AMICO a Silvano MARTELLO. Assignment Problems [online]. Society for Industrial and Applied Mathematics, 2012 [cit. 2022-11-28]. ISBN 978-1-61197-222-1. Dostupné z URL: <https://epubs.siam.org/doi/book/10.1137/1.9781611972238>.

[7] HAN, Lejia a John BANCROFT. Nearest approaches to multiple lines in n-dimensional space [online]. [cit. 2022-11-28]. Dostupné z URL: <https://api.semanticscholar.org/CorpusID:125573224>.

[8] Blender. The Software [online]. [cit. 2022-11-29]. Dostupné z URL: <https://www.blender.org/about/>.

A Obsah elektronické přílohy

Elektronická příloha obsahuje samotné rozšíření pro prostředí Blender, jež bylo v rámci této práce vytvořeno. Taktéž je zde možné nalést implementaci plánovací metody pomocí umělých potenciálových polí. Pro vývoj a testování byl použit programovací jazyk Python 3.9.5 a Blender 3.3.1 s dodatečným modulem scipy ve verzi 1.9.3.

```
/ ..... kořenový adresář přiloženého archivu
├── Addon_SwarmPlanner ..... složka rozšíření
│   ├── classes ..... implementace funkcí rozšíření
│   │   ├── SwarmArea.py
│   │   ├── SwarmExporter.py
│   │   ├── SwarmInit.py
│   │   ├── SwarmPainter.py
│   │   └── SwarmPlanner.py
│   ├── planning ..... implementace plánovacího algoritmu
│   │   ├── classes.py
│   │   ├── collision_detection.py
│   │   └── planner.py
│   └── __init__.py ..... inicializační soubor rozšíření
└── APF ..... implementace metody umělých potenciálových polí
    └── APF.py
```