VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PLÁNOVÁNÍ OPTIMÁLNÍ TRAJEKTORIE LETADLA S PŘEKÁŽKAMI

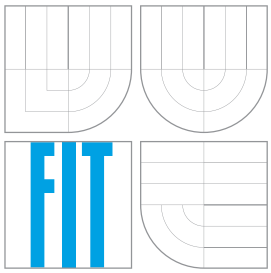
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MAREK OČENÁŠ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PLÁNOVÁNÍ OPTIMÁLNÍ TRAJEKTORIE LETADLA S PŘEKÁŽKAMI

PATH PLANNING OF AIRPLANE WITH OBSTACLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MAREK OČENÁŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN SAMEK, Ph.D.

BRNO 2013

Abstrakt

Cílem této diplomové práce je implementace plánování optimální trajektorie letadla letícího v nižších výškách, které se musí vyhýbat překážkám. Pro metodu plánování se předpokládá statické a předem známé prostředí. V této práci jsou popsány principy, optimálnost a složitost vybraných metod plánování. Na základě vlastností metod je vybrána metoda nejvhodnější k implementaci.

Abstract

The aim of this master's thesis is the implementation of optimal trajectory planning for an airplane flying in lower altitudes, which has to avoid collision with obstacles. For the planning we assume static and fully known environment. There are described principals, optimality and complexity for some chosen methods of planning in this thesis. And based on the methods' characteristics it's chosen the best method for implementation.

Klíčová slova

Plánování, trajektorie, metoda, statické prostředí, překážka, optimální cesta, implementace.

Keywords

Planning, Trajectory, Method, Static Environment, Obstacle, Optimal Path, Implementation.

Citace

Marek Očenáš: Plánování optimální trajektorie letadla s překážkami, diplomová práce, Brno, FIT VUT v Brně, 2013

Plánování optimální trajektorie letadla s překážkami

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana inženýra Jana Samka.

.....
Marek Očenáš
15. května 2013

Poděkování

Děkuji panu inženýru Janu Samkovi za odbornou pomoc a vedení při vytváření této diplomové práce.

Obsah

Kapitola 1

Úvod

Cílem této diplomové práce je prozkoumání několika metod pro plánování trajektorie objektu při pohybu v prostoru, srovnání jejich efektivity a použitelnosti pro plánování pohybu letadla ve trojrozměrném prostoru s vyhýbáním se překážkám a následný výběr nejvhodnější metody pro implementaci aplikace. Jedná se o problém nalezení optimální, tedy nejkratší, cesty z určeného počátečního bodu do určeného cílového místa, přičemž se objekt nesmí srazit s žádnou z překážek v prostoru a nesmí porušit žádná definovaná omezení jeho pohybu vyplývající z jeho pohybových vlastností. Například letadlo nesmí letět příliš pomalu, nesmí příliš strmě stoupat a nemůže příliš prudce zatáčet.

V následující kapitole budou popsány základní druhy plánování a následně popsány principy několika obecně používaných metod pro plánování trajektorie. Protože jednotlivé metody jsou obecně určeny pro plánování trajektorie různých druhů hmotných bodů ve dvourozměrném i trojrozměrném prostoru, při popisu jejich podstaty bude brán v úvahu především princip dané metody plánování ve trojrozměrném prostoru s omezeními pohybu hmotného bodu. Také bude vysvětleno, jakým způsobem metoda řeší vyhýbání se nepohyblivým překážkám a zda je trajektorie nalezená při výpočtu optimální.

Další kapitola se bude věnovat porovnání jednotlivých metod z pohledu jejich použitelnosti pro zadaný problém, efektivity a nalezení optimální trajektorie. Při hodnocení efektivity dané metody bude brána do úvahy její časová a paměťová náročnost, ale také užitečnost jejích výpočtů pro zadaný problém. Například není nutné aby metoda vypočítávala několik různých trajektorií od startovního ke koncovému bodu, protože cílem výpočtu při řešení zadaného problému je jediná optimální trajektorie. Výsledkem srovnání pak bude výběr nejvhodnější metody pro řešení plánování trajektorie s odůvodněním použití zvolené metody.

V kapitole 4 potom bude popsán návrh samotné aplikace, požadavky na její vstupy a především výstup a navržená struktura aplikace. Bude blíže popsán formát vstupních a výstupních dat, možnosti vizualizace výstupních dat a nástroj k ní použitý. Bude také nastíněna hlavní struktura aplikace, rozdělení výpočtu na jednotlivé fáze a podrobnější návrh implementace zvolené metody plánování trajektorie.

Kapitola Implementace 5 se zaměří na popis výsledné aplikace, její strukturu, detailní popis vstupů a výstupů a implementaci zvolené metody plánování trajektorie. Budou popsány algoritmy použité v implementaci, diskutována jejich důležitost a složitost. Bude také popsán přesný formát výstupních dat a možnost jejich dalšího využití, vizualizace.

V závěru práce se zaměřím na časovou složitost výpočtu aplikace, porovnání časové složitosti zvolené metody a její implementace s časovou složitostí ostatních metod. Budou shrnuty vlastnosti aplikace, zejména podmínky pro úspěšné dokončení výpočtu.

Kapitola 2

Popis vybraných metod plánování trajektorie

Plánování pohybu objektu v prostoru spočívá v nalezení cesty z počátečního určeného bodu do cílového bodu nebo regionu. Plánování trajektorie letícího letadla je určeno následujícími typy plánování:

- Neholonomní plánování – neholonomní pohybující se objekt je při pohybu omezen současným směrem. Další pohyb objektu je závislý, kromě jeho současné polohy, na jeho současném směru pohybu, který nemůže být okamžitě změněn. Například stojící auto se může rozjet směrem dopředu nebo dozadu, ale nemůže se rozjet do boku. Pro změnu směru, kterým je objekt otočen a kterým se tudíž může pohybovat, pak objekt musí urazit určitou vzdálenost, protože má zadáno maximální zakřivení pohybu (minimální poloměr zatáčky). Pro letící letadlo platí obdobné omezení, tedy nemůže začít letět bokem, letí vždy ve směru, kterým je natočeno.

- Kinodynamické plánování – pro pohybující se objekt jsou určena omezení jeho rychlosti a zrychlení, které nesmí přesáhnout určitou úroveň. Tato omezení představují pro výpočet diferenciální omezení prvního (rychlost) a druhého řádu (zrychlení). Omezení vyplývají z fyzikální podstaty objektu. Pro letící letadlo je definována jeho nejvyšší, avšak také nejnižší dovolená rychlost, náklon stoupání, apod.

- Plánování trajektorie – označení, které v historii označovalo vymezení funkce pozice a rychlosti v čase pro pohyb robotického ramene. Při plánování trajektorie pohybujícího se objektu se pak jedná o výpočet funkce polohy a rychlosti v čase, při dodržení kinodynamických omezení pohybu objektu.

Letící letadlo představuje neholonomní objekt, letí vždy směrem "za nosem", pro změnu směru pohybu musí změnit úhel natočení nosu. Při svém pohybu také musí dodržovat kinodynamická omezení svého pohybu v čase, která jsou daná jeho podstatou. Jedná se o diferenciální omezení prvního řádu: nejvyšší a nejnižší rychlost, maximální úhel stoupání a klesání, maximální zakřivení zatáčení, apod., a také o diferenciální omezení druhého řádu jako je zrychlení. V následujících podkapitolách budou popsány principy několika metod plánování při využití pro plánování trajektorie takového neholonomního objektu.

2.1 Sampling-based planning

Celý anglický název metody Sampling-based planning under differential constraints [4] by se dal volně přeložit jako Plánování založené na vzorkování s respektováním diferenciálních omezení. Tato metoda plánuje trajektorii objektu v obecně n-rozměrném prostoru s vyhýbáním se překážkám a s respektováním diferenciálních omezení druhého řádu pro pohyb objektu.

2.1.1 Definice prostoru

Prostor, ve kterém se pohybuje objekt, v našem případě letadlo, je chápán jako spojitý n-rozměrný konfigurační prostor, tedy množina všech n-rozměrných vektorů. Každý vektor určuje pozici (souřadnice) bodu v prostoru, na kterých se může letadlo nacházet. Konfigurační prostor se označuje C. Souřadnice, na kterých se momentálně letadlo nachází určují jeho polohu v prostoru. K určení jeho stavu je ale potřeba více údajů. Protože letadlo má diferenciální omezení pohybu nejen prvního řádu, ale i druhého řádu, je potřeba v každém okamžiku znát nejen polohu, ale také k dalších souřadnic popisujících vnitřní stav letadla, například 3 souřadnice vektoru jeho rychlosti. Sloučením tří souřadnic polohy a tří souřadnic rychlosti vznikne 6-rozměrný vektor jednoznačně udávající aktuální stav letadla.

Nad konfiguračním prostorem C je definován stavový prostor X, který rozšiřuje prostor C o k dalších souřadnic. Obecně při plánování nemusí být zapotřebí uchovávat další hodnoty objektu pro popis jeho stavu, pokud není, pak $k = 0$ a $X = C$.

C je n-rozměrný spojitý konfigurační prostor

X je $(n + k)$-rozměrný spojitý stavový prostor, který vznikne nad prostorem C

Konfigurační prostor C označuje celý prostor. V prostoru se ale nachází překážky, kterým se objekt musí vyhnout. Proto je konfigurační prostor C rozdělen na části C_{free} a C_{obs}[1]. $C_{free} \subset C$ označuje část prostoru, která je volná. V této části prostoru se objekt může pohybovat. $C_{obs} \subset C$ označuje překážky, tedy část prostoru, do které objekt nesmí vstoupit. Pro části prostoru C platí:

$$C_{free} \cup C_{obs} = C \quad \wedge \quad C_{free} \cap C_{obs} = \{\} \tag{2.1}$$

Podobně je také stavový prostor X nad prostorem C rozdělen na X_{free} a X_{obs}, navíc také na X_{ric}[2]. X_{ric} označuje stavy, ve kterých objekt nedokáže zabránit srážce (například letadlo letící proti zdi). Přestože v tomto stavu objekt ještě není v kolizi s překážkou, nesmí se do takového stavu dostat. Rozdělit stavový prostor na X_{free} a X_{obs} je jednoduché (není problém určit, kdy se letadlo srazí s překážkou). Naopak výpočet X_{ric} je velmi obtížný, dokonce určit u konkrétního stavu x, zda $x \in X_{ric}$ je obtížné. Proto se X_{ric} v praxi nevypočítává.

2.1.2 Trajektorie

Trajektorie, kterou se metoda snaží vypočítat, označuje průchod objektu prostorem. Jedná se tedy o posloupnost stavů, ve kterých se objekt během své pouti nachází. Protože čas je spojitý, je těchto stavů nekonečně mnoho, což není pro výpočet vhodné. Proto je čas vzorkován. Původně spojitý čas T je rozdělen na časové intervaly Δt. V každém časovém

[1] *free* znamená anglicky volný; *obs* označuje anglické slovo *obstacle*, což znamená překážka.
[2] *ric* označuje *region of inevitable collision*, tedy oblast nevyhnutelné srážky.

intervalu objekt urazí nějakou dráhu v prostoru a přejde z jednoho stavu do jiného. Pohyb objektu prostorem je pak určen stavovou trajektorií, ta je definována jako posloupnost všech stavů, kterými objekt projde od počátečního stavu až ke koncovému, a které jsou od sebe časově vzdáleny přesně o interval Δt.

Objekt, který se pohybuje prostorem, může provádět akce, které ovlivní jeho stav i jeho pohyb (například zvýšení rychlosti). Přechod z jednoho stavu do druhého proto nezávisí jen na předchozím stavu, ale také na akci, kterou objekt provedl. Akce, které může objekt provést definuje akční prostor U, který je opět teoreticky spojitý (letadlo může například zrychlit o různou hodnotu rychlosti). Pro výpočet je potřeba, aby byl počet proveditelných akcí konečný. Pro výpočet je tedy zvolena konečná podmnožina $U_d \subset U$. Každá akce $u(t) \in U_d$ musí být konstantní po celou dobu časového intervalu Δt. Pohyb objektu prostorem lze pak také popsat akční trajektorií. Akční trajektorie je posloupnost akcí $(u_1, u_2, u_3, ...u_k)$ prováděná objektem v každém stavu, kde každá akce je prováděná po celou dobu Δt, tedy až do přechodu do následujícího stavu. Akční trajektorie představuje seznam povelů pro letadlo, jaké má provádět manévry, aby dosáhlo cíle, takže akční trajektorie pro optimální cestu do cíle představuje řešení problému.

2.1.3 Výpočet

Vlastní výpočet trajektorie objektu spočívá ve vytvoření stromu dostupnosti, což je graf znázorňující, do kterých stavů je možné se z počátečního stavu dostat. Příklad stavového stromu znázorňuje obrázek 2.1. Každý uzel grafu představuje některý stav objektu, hrany mezi uzly představují akce prováděné objektem během časového intervalu Δt. Kořenovým uzlem je počáteční stav. Další uzly vzniknou aplikací všech možných akcí, které objekt může provést, jako stavy, do kterých se objekt z předchozího stavu dostane při provedení této akce. Jakmile některý nově vytvářený uzel představuje stav ležící v cílovém regionu, je zřejmé, že cílový region je z počátečního stavu dostupný a výpočet je možné ukončit. Protože během výpočtu není nutné vícekrát počítat se stejnými stavy, je možné (a vhodné) uzly stromu dostupnosti představující stejné stavy objektu sloučit. Vznikne tak graf dostupnosti, který nemusí být stromem, obsahuje však menší množství uzlů.

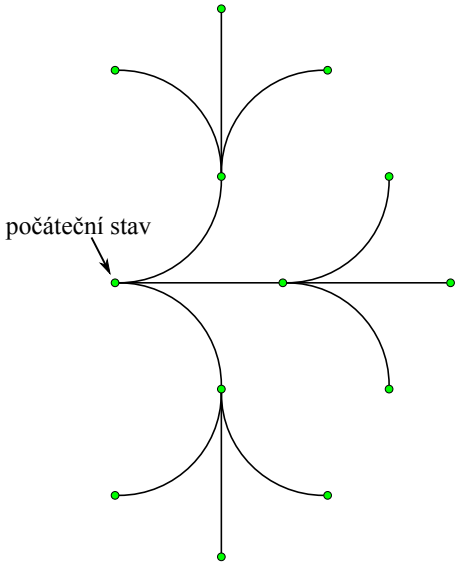
Při konstrukci grafu dostupnosti je nastaven základní časový interval Δt. Systematickým prohledáváním grafu dostupnosti je zjištěno, zda řešení existuje, tedy zda je cílový region z počátečního stavu dostupný. Pokud řešení existuje, je také nalezena cesta v grafu dostupnosti popisující akční resp. stavovou trajektorii objektu. Pokud řešení není nalezeno, sníží se interval Δt a celý výpočet se opakuje s menším časovým krokem. Pokud neexistuje žádné řešení, algoritmus poběží teoreticky do nekonečna, v praxi může být výpočet omezen například určením nejmenšího Δt, se kterým se bude počítat.

Plánování trajektorie pak spočívá v nalezení cesty v grafu dostupnosti z počátečního (kořenového) uzlu do uzlu představujícího stav ležící v cílovém regionu. Postup metody je následující:

1. **Inicializace:** Nastav graf dostupnosti $G(V, E)$, jako neorientovaný prohledávací graf, pro který platí:
$$x \in V \Rightarrow x \in X_{free}$$
tedy každý uzel grafu reprezentuje stavy, ve kterých objekt není v kolizi s překážkou.

2. **Výběr uzlu:** Zvol uzel $x_{cur} \in V$ pro zpracování.

3. **Lokální plánování:** Vygeneruj posun do následujícího stavu $x_r \in X_{free}$. x_r může a nemusí být uzlem v G. Ověř, zda je posun bezkolizní, pokud není, vrať se na bod 2.



Obrázek 2.1: Stavový strom pro dva kroky objektu se třemi možnostmi pohybu.

4. **Vložení hrany:** Pro vygenerovaný posun vlož do E novou hranu. Pokud $x_r \notin V$, pak přidej x_r do V. Pokud x_{cur} leží na dráze jiné hrany $e \in E$, je e rozdělena na dvě části s uzlem x_{cur} mezi nimi.

5. **Kontrola řešení:** Zkontroluj, zda G obsahuje cestu k cíli, pokud ano, ukonči výpočet úspěšně.

6. **Opakování:** Ověř, zda byla splněna některá ukončující podmínka, pokud ano, ukonči výpočet neúspěšně. Jinak se vrať k bodu 2.

Během výpočtů je potřeba pro každý uzel určit, zda stav, který představuje, není v kolizi s překážkou a případně jej vyloučit z dalšího výpočtu. Metoda sama o sobě nedefinuje algoritmus, který toto rozhodne, je však důležité, aby byl algoritmus efektivní. Přestože dva stavy, předchozí a následující, nejsou v kolizi s překážkou, může se překážka vyskytovat mezi nimi. Na kolizi je nutné testovat celý úsek cesty od předchozího stavu až po aktuální, zkoumání kolizí stavu tedy není triviální problém.

Dalším často řešeným dílčím výpočtem je integrace akční trajektorie. Akční trajektorie představuje posloupnost akcí prováděných objektem, sama o sobě ale nic neříká o jeho stavu po provedení každé z těchto akcí. Pro vytvoření stavové trajektorie, je třeba provést výpočet, který na základě předchozího stavu určí následující stav, tedy integraci trajektorie po dobu intervalu Δt. Na přesnosti integrace závisí přesnost celého výpočtu trajektorie.

2.1.4 Optimálnost

Metoda převádí problém plánování trajektorie objektu na problém prohledávání grafu dostupnosti. Protože prohledávaný stavový prostor představuje jen podmnožinu skutečného spojitého stavového prostoru, nemusí být řešení vždy nalezeno. Při hledání řešení bude algoritmus pracovat dokud jej nenalezne nebo dokud nenastane nějaká ukončující podmínka (například dosažení minimálního časového kroku Δt). V tom druhém případě není řešení

nalezeno, nelze ovšem tvrdit že neexistuje. Pro bezpečné vyloučení existence řešení by algoritmus musel pracovat nekonečně dlouho.

Celková efektivita metody a to, zda nalezené řešení (pokud je nalezeno) je optimální, závisí na způsobu systematického prohledávání grafu dostupnosti, tedy na použité metodě prohledávání grafů. To, zda nalezená trajektorie bude optimální, tedy závisí na optimálnosti prohledávací metody. K prohledávání grafu je vhodnější použít informované prohledávací metody, které neprohledávají všechny tedy i zbytečně dlouhé cesty a použít metodu, jejíž nalezená cesta v grafu je optimální, například metodu A^*.

2.2 A Real-time 3D motion planning and simulation scheme for nonholomic systems

Název přeložitelný jako Plánování 3D pohybu v reálném čase a simulační schéma pro neholonomní systémy [8] označuje metodu pro výpočet trajektorie neholonomního objektu v trojrozměrném prostoru s vyhýbáním se překážkám, a to v reálném čase. Prostředí ve kterém se objekt pohybuje nemusí být dopředu zcela známé. Objekt může informace o svém okolí získávat postupně (například radarem) a vypočítává trajektorii vždy na kratší úsek, po jehož uplynutí vypočítá další úsek trajektorie. Pokud je prostředí předem známé, je také možné vypočítat najednou celou trajektorii až do cílového regionu.

2.2.1 Stav letadla

Neholonomním objektem je v našem případě letadlo. Protože letadlo letí vždy směrem, kterým je natočeno a nemůže ho okamžitě změnit, je třeba si stav letadla pamatovat a trajektorii přizpůsobit diferenciálním omezením pohybu letadla. Letadlo má 6 stupňů volnosti, tedy 6 proměnných udávající jeho aktuální stav. Těmito proměnnými jsou:

- $p = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ vyjadřuje polohu letadla v prostoru,
  X, Y, Z: jednotlivé souřadnice polohy letadla.
- $\omega = \begin{bmatrix} \theta \\ \varphi \\ \psi \end{bmatrix}$: vyjadřuje úhel otočení letadla,
  - θ: vertikální úhel, tedy náklon stoupání,
  - φ: úhel natočení letadla na levé či pravé křídlo,
  - ψ: horizontální úhel směru letu, tedy kurz.

Protože letadlo nemůže žádný z úhlů ω změnit okamžitě, ale změna musí být plynulá, musí být křivka, po které se letadlo pohybuje hladká. Tedy zakřivení trajektorie se musí také měnit plynule. To vyžaduje, aby mělo letadlo méně vstupů ovládání, než stupňů volnosti. Vstupy ovládání (řízení) letadla jsou řízení výkonu motorů (ovládá rychlost), výškové kormidlo (ovládá náklon stoupání), směrové kormidlo (ovládá horizontální úhel) a křidélka (ovládají náklon letadla na křídlo). Vstupy ovládání jsou tedy celkem 4, což je méně než 6 stupňů volnosti.

2.2.2 Dynamicky alokované body

Při výpočtu trajektorie s vyhýbáním se překážkám metoda vypočítává tzv. dynamicky alokované body (dále jen DAB). Plánování trajektorie spočívá ve vytvoření posloupnosti DAB, které navádí letadlo tak, aby se vyhnulo překážkám a dospělo k cílovému bodu. Prvním bodem posloupnosti je počáteční bod a posledním bodem bod v cíli. Trajektorie pohybu propojuje jednotlivé DAB. Výsledná trajektorie nemusí procházet přímo DAB, ale sleduje jejich směr.

Takto jsou vytvořeny jednotlivé úseky či segmenty trajektorie. Protože křivka popisující dráhu letadla musí být hladká, stav na konci jednoho segmentu musí být stejný jako stav na začátku následujícího segmentu. Pro každý segment je tedy definován počáteční stav, na začátku je to zadaný počáteční stav a na začátku každého dalšího segmentu stav, ve kterém se letadlo nacházelo na konci předešlého segmentu.

Metoda počítá s předem ne zcela známým prostředím, kde letadlo zná informace o svém bližším okolí, ale může mít jen nepřesné či žádné informace o vzdálenějším okolí. Jednotlivé DAB jsou vypočítávány postupně během výpočtu trajektorie. Jak letadlo získává či zpřesňuje informace o svém okolí, tedy překážkách vyskytujících se v prostoru, zpřesňuje také výpočet trajektorie přidáváním dalších DAB do posloupnosti. Vždy před dosažením jistého DAB se však musí letadlo rozhodnout pro příští úsek trajektorie. Pro následující úsek trajektorie již nesmí být posloupnost DAB změněna a je vypočítána trajektorie spojující dva DAB, tedy trajektorie letadla pro další segment. DAB jsou určovány podle následujících pravidel:

1. Pokud cestu od aktuální pozice objektu k cíli neblokuje žádná překážka, příští DAB je nastaven na úsečku spojující aktuální pozici objektu s cílovým bodem. Vzdálenost tohoto DAB od aktuální pozice objektu odpovídá maximální vzdálenosti, na kterou objekt zjišťuje informace o svém okolí (např. dosah radaru).

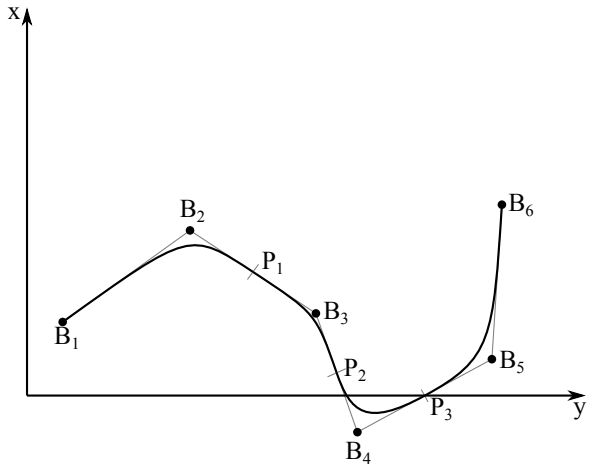
2. Pokud cestu od aktuální pozice objektu k cíli blokuje překážka a žádný DAB není přiřazen k okrajům překážky, pak je nový DAB nastaven v blízkosti nejbližšího okraje překážky s dostatečnou vzdáleností od překážky pro bezpečné zabránění kolize vzhledem k aktuální rychlosti objektu.

3. Pokud cestu od aktuální pozice objektu k cíli blokuje překážka a nějaký DAB již byl přiřazen k okrajům překážky, pak je příští DAB nastaven na přímku procházející přes aktuální pozici objektu a následující DAB. Vzdálenost nového DAB od aktuální pozice objektu je dána maximální vzdáleností detekce překážek objektem.

Propojením jednotlivých DAB vznikne výsledná trajektorie. Segmenty od jednoho bodu k následujícímu mohou být tvořeny buď úsečkou, tedy bez zakřivení nebo křivkou. Jako křivky pro propojování DAB metoda využívá segmentů Eulerovy spirály, o kterých pojednává následující podkapitola.

2.2.3 Eulerova spirála

Metoda používá pro výpočet trajektorie segmenty Eulerovy spirály. Eulerova spirála, je dvourozměrná spirála s lineárně se měnícím zakřivením. Na počátku je zakřivení nulové, poloměr zatáčení je nekonečný. Jak se spirála otáčí okolo středu, zakřivení se lineárně zvětšuje, poloměr zatáčení se snižuje. Po nekonečně otáčkách okolo středu se teoreticky poloměr sníží



Obrázek 2.2: Příklad vytvoření trajektorie z oblouků Eulerových spirál.

na 0. Protože zakřivení Eulerovy spirály se mění lineárně, je křivka hladká a může popisovat skutečnou trajektorii letícího letadla. Tečnu a zakřivení Eulerovy spirály popisují rovnice:

$$Cv = \int_0^S k \mathrm{d}s = k \cdot s$$

$$\theta = \theta_0 \cdot \int_0^S Cv(s) \mathrm{d}s = \theta_0 + \frac{1}{2} \cdot l \cdot s^2$$

Kde Cv představuje zakřivení Eulerovy spirály, k rychlost změny zakřivení, θ_0 počáteční směr spirály (uvažujeme $\theta_0 = 0$) a θ směr spirály.

Souřadnice Eulerovy spirály v rovině jsou funkce:

$$x(s) = \frac{A}{(2k)^{\frac{1}{2}}} \cdot \int_0^\theta \frac{\cos\theta}{\theta^{\frac{1}{2}}} \mathrm{d}\theta$$

$$y(s) = \frac{A}{(2k)^{\frac{1}{2}}} \cdot \int_0^\theta \frac{\sin\theta}{\theta^{\frac{1}{2}}} \mathrm{d}\theta$$

kde A představuje měřítko.

Při výpočtu trajektorie metoda počítá segmenty oblouků Eulerovy spirály, kterými spojuje dynamicky alokované body (DAB) trajektorie, jak naznačuje obrázek 2.2. Pro vytvoření dalšího segmentu trajektorie se použijí počáteční segmenty dvou Eulerových spirál vycházejících z počátečních bodů DAB nebo pomocných bodů uprostřed mezi dvěma DAB. Pro příklad uvedený na obrázku 2.2 pro první segment trajektorie bude jedna Eulerova spirála vycházet z bodu B_1 a druhá z bodu P_1. Oba oblouky Eulerových spirál se potkají uprostřed dráhy mezi body B_1 a P_1, kde výsledná křivka plynule přejde z jedné Eulerovy spirály do druhé. Takto je vytvořena křivka spojující body B_1 a P_1.

Eulerova spirála je křivka ve 2D rovině. Je však možné převést Eulerovu spirálu také pro 3D prostor. Pro výpočet segmentu trajektorie jsou zapotřebí tři body, které jsou propojeny oblouky dvou Eulerových spirál. Tři body v prostoru, které neleží v přímce definují rovinu. V 3D prostoru je tedy definována 2D rovina, ve které leží oba oblouky Eulerových spirál, které dané body propojují. Výpočet Eulerových spirál je tedy prováděn v rovině a výsledné souřadnice jsou přepočítány do souřadnic 3D prostoru.

2.2.4 Optimálnost

Pro tuto metodu bylo dokázáno [8], že nejkratší cesta v rovině je konkatenace (spojení) segmentů úseček a oblouků Eulerových spirál, kde derivace zakřivení je omezená. Protože úsečky i oblouky Eulerových spirál je možné převést i do 3D prostoru, je trajektorie sestavená ze segmentů úseček a oblouků Eulerových spirál nejkratší. Nicméně nejkratší cesta je složená z nekonečného množství takovýchto segmentů. Proto výpočet nejkratší cesty při zachování omezení derivace zakřivení není touto metodou v praxi dosažitelný.

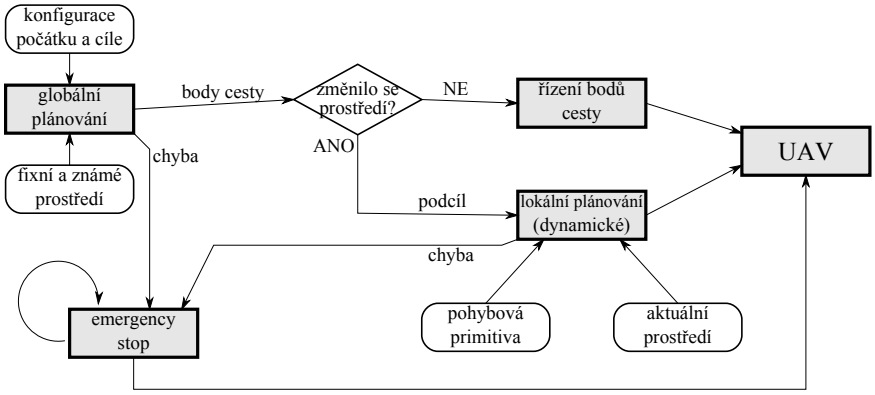
2.3 Efficient Two-phase 3D motion planning for small fixed-wing UAVs

Další metodou pro plánování trajektorie objektů ve 3D prostoru je metoda nazvaná Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3]. Volně přeložený do češtiny je název Efektivní dvoufázové plánování 3D pohybu pro malá UAV s pevnými křídly. Zkratka UAV (Unmanned Aerial Vehicle) označuje bezpilotní letadlo. Metoda je určená především pro plánování trajektorie pro malá bezpilotní letadla (dále jen UAV) pohybující se v prostoru s překážkami, například UAV monitorující dopravní situaci pohybující se mezi vyššími domy. Pro pohyb s danou nejnižší rychlostí mezi budovami či jinými podobnými překážkami je potřeba velká manévrovatelnost a rychlá přizpůsobivost okolním překážkám, vyžaduje tedy plánování v reálném čase.

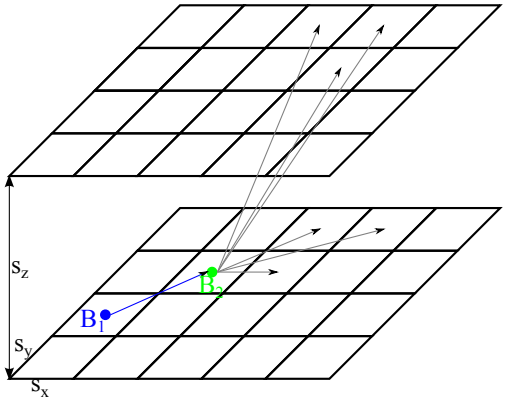
Pro rychlé plánování trajektorie v reálném čase a přitom také nalezení optimální cesty do cíle tato metoda provádí plánování ve dvou fázích. Nejprve je nahrubo vypočítána kinematicky vyhovující bezkolizní cesta v diskrétním 3D prostoru, tedy globální plánování. Takto vypočítaná cesta je nazvána globální cesta. Poté ve druhé fázi plánování je využito lokální plánování pro zpřesnění globální cesty pro její kratší úsek. Lokální plánování se opakuje vždy pro kratší úsek globální cesty, ke kterému se UAV blíží. Postup plánování trajektorie popisuje obrázek 2.3. V tomto obrázku je také naznačen postup pro nouzový únik UAV v případě selhání plánování trajektorie (blok Emergency stop). Nouzové plánování představuje záložní řešení, které navede UAV do bezpečného regionu s nejmenší pravděpodobností srážky. Metoda však nepředepisuje přesný způsob nouzového plánování a pro vlastní výpočet není podstatné, takže se jím v tomto textu nebudu dále zabývat.

2.3.1 Globální plánování

Nejprve je vytvořen diskrétní konfigurační prostor, ve kterém bude vypočítána globální cesta. Diskrétní prostor představuje mřížka. V této mřížce jsou definovány uzly a jejich propojení. Uzel je definován pozicí ve 3D mřížce a propojením z rodičovského uzlu (předcházející uzel v cestě). 3D mřížku si lze představit jako několik 2D mřížek naskládaných na sebe. Pozice uzlu je pak definována jako souřadnice čtverečku, ve kterém se uzel nachází, jak ukazuje obrázek 2.4. Globální cesta představuje posloupnost uzlů umístěných v mřížce



Obrázek 2.3: Postup plánování trajektorie UAV ve dvou fázích.



Obrázek 2.4: Diskrétní 3D mřížka: Definice uzlu a propojení

a jejich propojení od počátečního uzlu do koncového. Propojení uzlů má 26 variant a může být reprezentováno jediným číslem, místo dvou úhlů, horizontální úhel (kurz) φ a vertikální úhel θ (náklon stoupání). S úhlem představujícím náklon UAV na křídlo ψ se nepočítá, protože ani nelze v mřížce zachytit.

Při sestavování diskrétní mřížky aproximující spojitý prostor, je potřeba zvolit její velikost. Podle obrázku 2.4 jsou rozměry mřížky označeny s_x, s_y a s_z. Velikost mřížky může být stanovena libovolně, ale obvykle je určena podle manévrovacích vlastností letadla r_{min} představující minimální poloměr otáčení UAV a θ_{max} označující maximální povolený náklon stoupání/klesání, podle následujících definic:

$$s_x = s_y = \frac{2}{3} \cdot r_{min}$$

$$s_z = s_x \cdot \mathrm{tg}\, \theta_{max}$$

Propojení jednotlivých uzlů může být definováno libovolně, musí však respektovat omezení pohybu letadla. Propojení mezi uzly má 26 možných variant, z nich jsou však pro konkrétní propojení použitelné jen některé. Propojení se rozdělují na:

- horizontální propojení – je množina propojení svírající s propojením s rodičovským

uzlem (předchozí propojení v globální cestě) úhel menší než 90°, pohybuje se jen horizontálně,

- vertikální propojení – mění výšku UAV, množina propojení závisí na schopnostech UAV vertikálně manévrovat.

Složením těchto dvou propojení vznikne výsledné propojení dvou uzlů globální cesty.

Při plánování globální cesty se množina všech uzlů a vyhovujících variant jejich propojení považuje za graf, ve kterém je hledána nejkratší cesta z počátečního uzlu do koncového. Protože pro prohledávání grafu je velmi vhodné použít informované prohledávací metody, doporučená je metoda A^*. Pro takové prohledávací metody je potřeba každý uzel ohodnotit předpokládanou vzdáleností do cíle. Pro toto ohodnocení ovšem Euklidovská vzdálenost není dostatečně dobrou heuristikou, protože uzel je definován kromě pozice také směrem pohybu. Obvykle se proto uzly ohodnocují následujícím způsobem:

1. Určení předpokládané ceny do cíle podle Euklidovské vzdálenosti pro zvolený region okolo počátku.

2. Pokud je třeba heuristika pro aktuální uzel, vypočítaný region se posune na aktuální uzel. Pokud je cíl mimo region, je nalezen uzel nejblíže cíli se stejným rodičovským propojením v regionu a poté je přidána Euklidovská vzdálenost z cíle do tohoto uzlu jako jeho celková předpokládaná cena.

Ve stavovém stromu je metodou A^* nalezena optimální cesta.

Globální cesta je naplánována vzhledem k předem známému prostředí a jsou zanedbány malé překážky, které nejsou vidět v použité mřížce. K naplánování přesné trajektorie je globální cesta obvykle ještě upřesněna lokálním plánováním, i když, pokud se neobjeví malé překážky nebo se prostředí nezmění, je možné přímo sledovat globální cestu, protože respektuje omezení pohybu letadla a je bezkolizní.

2.3.2 Lokální plánování

Lokální plánování kompenzuje nedostatek globálního plánování, neschopnost vyhnout se malým nebo pohybujícím se překážkám. Je spuštěno vždy, když je potřeba provést jemnější plánování přes krátké úseky globální cesty, jak ukazuje obrázek 2.3. Lokální plánování nalezne cestu propojující dva sousední body z globálního plánování pomocí předem určených pohybových primitiv. Použitý konfigurační prostor je stejný jako při globálním plánování a stejně tak je také z výpočtu vynechán úhel náklonu UAV na křídlo ψ.

Lokální bezkolizní cesta vznikne řešením vyhledávacího grafu, kde jednotlivé body cesty jsou propojeny pomocí pohybových primitiv. Pohybová primitiva odrážejí dynamické chování UAV, jsou vypočítána předem podle zadaných omezení pohybu UAV a uložena ve vyhledávací tabulce. Pro každé primitivum je určen počáteční a koncový bod a několik bodů mezi nimi pro detekci kolizí. Nový bod lokální cesty x_{k+1} vznikne z předchozího bodu x_k použitím určitého pohybového primitiva u_k:

$$x_{k+1} = f_d\left(x_k, u_k\right)$$

Při hledání lokální cesty je nejprve sestaven stavový strom, podobně jako v metodě Sampling-based planning [4]. Pro prohledávání grafu je použita metoda Greedy search, která nemusí nalézt vždy optimální řešení. K odhadu ceny bodu do cíle se využijí heuristiky

založené na Dubinsových 2D křivkách, protože chování UAV je podobné chování Dubinsových vozidel, jen je rozšířené do třetího rozměru. Pro rozšíření 2D Dubinsovy křivky do 3D prostoru je možné využít dvě heuristiky:

1. Pro vyhledání 3D cesty se využije 2D křivka propojující dva body ve stejné výšce (zanedbává třetí rozměr). Dráha křivky pak musí být dostatečně dlouhá, aby při zadaném maximálním úhlu stoupání θ_{max} umožnila UAV vystoupat či klesnout o potřebnou výšku do cílového bodu.

2. Pro vyhledání 3D cesty se využijí dvě 2D křivky:

   - horizontální – popisuje cestu UAV v horizontální rovině,
   - vertikální – popisuje cestu výhradně ve vertikální rovině, horizontální chování se zanedbává.

   Dimenze těchto dvou uvažovaných rovin je menší než dimenze konfiguračního prostoru, ale poskytují vyhledávání dostatečně dobré informace o blízkosti k cíli.

2.3.3 Optimálnost

Metoda plánuje trajektorii ve dvou fázích. Plánování globální cesty aproximuje spojitý prostor diskrétní mřížkou představující množinu stavů UAV. Při použití vhodné heuristiky pro ohodnocení každého uzlu předpokládanou vzdáleností do cíle zajistí doporučená metoda prohledávání grafu A^*, že nalezené řešení je v rámci diskrétní mřížky optimální.

Lokální plánování sestavuje stavový strom stavů v rámci úseku globální cesty. K prohledávání stavového stromu se použije metoda Greedy search beroucí v úvahu ohodnocení stavů vhodnou heuristikou, tedy předpokládanou cenou stavu k cíli, může však uváznout v lokálním minimu. Nalezená lokální cesta tedy nemusí být optimální.

2.4 Time-optimal paths for a Dubins airplane

Metoda, jejíž název přeložený do češtiny zní Časově optimální cesta pro Dubinsovo letadlo [2], označuje další metodu pro plánování trajektorie letadla pohybujícího se ve trojrozměrném prostoru. Metoda rozšiřuje výpočet trajektorie Dubinsova auta, který probíhá ve dvourozměrné rovině tak, že autu přidá další souřadnici pro výšku. Tím je výpočet rozšířen do tří rozměrů. Pro plánování trajektorie s výškovým rozměrem rozlišujeme tři případy:

- Cesta s nízkou výškou – letadlo sleduje cestu, která je nejkratší v rovině $x-y$ a během postupu stoupá či klesá s menším úhlem než je maximální povolený do výšky cílového bodu.

- Cesta se střední výškou – pokud je cesta příliš krátká aby umožnila letadlu včas vystoupat či klesnout do výšky cílového bodu je potřeba ji prodloužit přidáním dalších zatáček, ne však smyček. Taková cesta promítnutá do roviny $x - y$ bude delší než nejkratší cesta.

- Cesta s vysokou výškou – vzhledem k velkému výškovému rozdílu počátečního a koncového bodu bude cesta dostatečně dlouhá, aby obsahovala smyčku (letadlo udělá okruh o 360° a vrátí se na původní dráhu, ovšem už v jiné výšce).

Pro výpočet se předpokládá, že Dubinsovo letadlo má nezávislé ovládání výšky a pohybu v horizontální rovině. K popisu časově optimální cesty je využit Pontryaginův Maximální princip jako nutná podmínka, která vyřadí všechny neoptimální cesty.

Plánování trajektorie lze zjednodušit na skládání za sebe jednotlivých elementárních kousků trajektorie vybíraných z knihovny. Tyto kousky jsou nazvané pohybová primitiva, jsou vypočítána předem a uložena v této knihovně. Časově optimální cesta vypočítaná touto metodou je složená ze zatáček minimálního poloměru zatáčení r_{min}, rovných úseků úseček a úseků s proměnlivým zakřivením propojujících ostatní úseky.

Dubinsovo auto se pohybuje ve dvourozměrné rovině. Pohybuje se jen dopředu, konstantní rychlostí a je omezeno v manévrování minimálním poloměrem zatáčení r_{min}. Při rozšíření tohoto modelu do třetího rozměru, tedy po přidání souřadnice výšky vznikne Dubinsovo letadlo. Pro něj platí podobný předpoklad, pohybuje se stále dopředu, konstantní rychlostí a je omezeno minimálním poloměrem otáčení r_{min}. Navíc oproti předchozímu modelu je také omezeno maximálním úhlem stoupání θ_{max}. Dubinsovo letadlo je čtyřrozměrný systém, jehož stav označuje proměnná q:

$$q = (x, y, z, \theta) \in C = \Re^3 \times \mathrm{S}^1$$

kde x, y a z jsou souřadnice v prostoru a θ je úhel svíraný souřadnicovou osou x a směrem pohybu letadla v rovině $x - y$. Vzhledem k omezením pro letadlo platí na sobě vzájemně nezávislá omezení pro θ' a z'. Celý systém lze popsat rovnicí:

$$q' = f(q, u) = f_0(q) + u_z \cdot f_1(q) + u_\theta \cdot f_2(q)$$

kde f_0, f_1 a f_2 jsou vektory popisující let letadla. Pokud uvažujeme $r_{min} = 1$ a $\theta_{max} = 1$, pak tyto vektory budou mít tvar:

$$f_0 = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \\ 0 \end{pmatrix}, f_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, f_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Bez újmy na obecnosti předpokládáme, že počáteční konfigurace je $q_0 = (0, 0, 0, 0)$ a cílová konfigurace je $q_F = (x_F, y_F, z_F, \theta_F)$. Dále předpokládáme, že $|u_z|, |u_\theta| \leq 1$. Pak region ovládání $U = [-1, 1]^2$ a $(u_z, u_\theta) \in U$. Pro každý pár počáteční a cílové konfigurace q_0 a q_F pak hledáme funkci přípustného ovládání u takovou, pro kterou cena cesty (v našem případě čas) je minimální. Funkce ovládání u mapuje ovládání letadla na čas, tedy popisuje, jak bude letadlo letět.

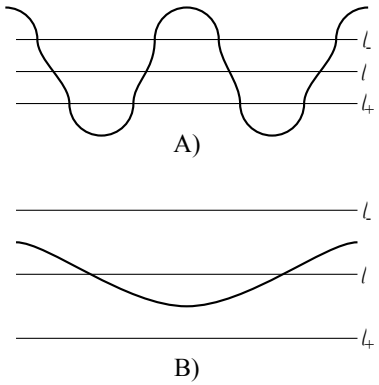
2.4.1 Pontryaginův Maximální princip

Z definice Pontryaginova Maximálního principu (dále jen PMP) v [2] vyplývá, že pro každou optimální trajektorii $q(t)$ spojenou s funkcí ovládání $u(t)$ existuje konstanta λ_0 a vektor λ, který je nenulový pokud $\lambda_0 = 0$:

$$\lambda_0 \geq 0, \qquad \lambda = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_1 \cdot y + c_2 \cdot x + c_4 \end{pmatrix}$$

kde c_1, c_2, c_3 a c_4 jsou konstanty.

Z PMP dále vyplývá, že nejrychlejší cesta pro letadlo může být:



Obrázek 2.5: Příklady Pravidelné cesty.

- cesta nulové délky,
- cesta, která je nejkratší v rovině $x-y$, přičemž výškový rozdíl počátečního a koncového bodu je dostatečně malý – cesta s nízkou výškou,
- cesta, která je právě tak dlouhá, aby umožnila letadlu překonat výškový rozdíl počátečního a koncového bodu vzhledem k θ_{max} – cesta se střední nebo vysokou výškou.

2.4.2 Plánování cesty s předepsanou délkou v rovině

Existují dva způsoby jak zajistit, aby cesta propojující počáteční a cílový bod měla předepsanou délku.

1. Pravidelná cesta

   Trajektorie se pohybuje okolo nejkratší cesty ℓ. V určité vzdálenosti od ℓ ohraničené ℓ_+ a ℓ_- zatáčí letadlo s poloměrem větším nebo stejným jako r_{min}. Za hranicemi ℓ_+ a ℓ_- letadlo vždy zatáčí s minimálním poloměrem. Příklad Pravidelné cesty je ukázán na obrázku 2.5.

2. Abnormální cesta

   (a) Časově abnormální cesta

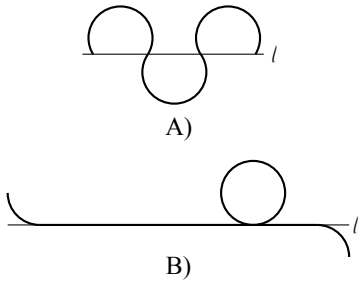
   Je složená výhradně z úseků s minimálním poloměrem zatáčení a z rovných úseků. Cesta může splynout z ℓ nebo se od ní odchýlit. Příklad ukazuje obrázek 2.6.

   (b) Lokálně nejdelší cesta

   Cesta nemůže splynout či se odchýlit od ℓ. Buď celá cesta splývá s ℓ nebo je složená z oblouků kružnice délky menší než π. Příklad ukazuje obrázek 2.7.

2.4.3 Plánování trajektorie

Rozlišují se tři způsoby plánování trajektorie podle horizontální vzdálenosti vzledem k výšce, kterou musí letadlo překonat. Označme nejkratší vzdálenost v rovině $x - y$, kterou musí letadlo urazit jako Δ. Tato vzdálenost Δ označuje délku nejkratší Dubinsovy křivky



Obrázek 2.6: Příklad Časově abnormální cesty.

Obrázek 2.7: Příklad Lokálně nejdelší cesty.

ze stavu $(0, 0, 0)$ do stavu (x_F, y_F, θ_F) (oba stavy v rovině $x-y$). Pak tři způsoby plánování jsou:

1. Časově optimální trajektorie pro nízké výšky

   Sledují nejkratší cestu v rovině $x-y$ se změnou výšky menší než povoluje θ_{max}. Délka optimální cesty je pak právě Δ. Vertikální rychlost je pak určena jako $u_z = \frac{z_F}{\Delta}$.

2. Časově optimální trajektorie pro vysoké výšky

   Pokud je výška cílového bodu dostatečně vysoká, je pro letadlo dostatečný prostor, aby po dosažení cílového bodu v horizontální rovině kroužilo po spirále a stoupalo či klesalo do správné výšky. Délka trajektorie je pak větší nebo rovna $\Delta + 2\pi$. Poloměr spirály lze pak zvolit stejný nebo větší, než je r_{min}. Časově optimální trajektorie se pak skládá z horizontální nejkratší cesty a ze spirály o poloměru $u_\theta = \frac{2\pi}{|z_F|-\Delta}$. V obou těchto částech cesty letadlo klesá či stoupá v úhlu θ_{max}.

3. Časově optimální cesta pro střední výšky

   Pokud je výška cílového bodu taková, že délka trajektorie při dodržení saturace stoupání či klesání v úhlu θ_{max} je $\Delta < |z_F| < \Delta + 2\pi$, pak není dostatek místa pro spirálu při stoupání do cílového bodu. Horizontální křivka je pak prodloužena přidáním dalších oblouků jak definuje Lokálně nejdelší cesta popsaná v předchozí podkapitole nebo Časově abnormální cesta rovněž popsaná v předchozí podkapitole, v případě Časově abnormální cesty však výsledná trajektorie není optimální, protože je delší než je nezbytně nutné pro změnu výšky letadla.

Kapitola 3

Srovnání jednotlivých metod plánování

3.1 Vlastnosti jednotlivých metod

V této kapitole porovnám vlastnosti čtyř zmíněných metod pro plánování trajektorie ve trojrozměrném prostoru, vzhledem k ostatním zmíněným metodám a s přihlédnutím k vlastnostem požadovaným zadáním této diplomové práce. Zadání vyžaduje plánování trajektorie v prostoru, kde informace o všech překážkách jsou známé na začátku plánování a v průběhu letu letadla se nemění. Nalezená trajektorie musí být optimální nebo se musí optimální trajektorii co nejvíce blížit. Časová složitost není kritická, protože výpočet neprobíhá v reálném čase, prvořadá je optimálnost nalezené trajektorie, přesto je vhodné, aby algoritmus ukončil výpočet v konečném čase.

Protože výpočet má všechny informace přístupné už před zahájením letu letadla, je vhodné, aby trajektorie, kterou daná metoda vypočítá, byla také známá ještě před zahájením letu letadla. Všechny informace o trajektorii jsou tedy zpracovány jednorázově jako výpočet předcházející letu letadla nebo jeho simulaci. V průběhu letu letadla nebo jeho simulace už tedy nejsou prováděny žádné další výpočty ohledně plánování trajektorie.

3.1.1 Sampling-based planning

Tato metoda vzorkováním převádí spojité veličiny, kterými jsou konfigurační a stavový prostor, akční prostor (všechny možné akce ovládání proveditelné letadlem) a překážky v prostoru, na diskrétní veličiny. Počítání s diskrétními veličinami umožňuje rozdělit trajektorii na jednotlivé stavy a prohledávat všechny diskrétní možnosti přechodu do dalších stavů, což je způsob, kterým metoda pracuje. Diskrétní veličiny ovšem znepřesňují výpočet, protože se jedná o aproximace původně spojitých veličin.

V porovnání s ostatními dříve uvedenými metodami má metoda Sampling-based planning under differential constraints [4] následující vlastnosti:

- Klady:
  - \+ jednoduchý princip metody,
  - \+ při použití vhodné metody prohledávání grafu je nalezená trajektorie optimální v rámci aproximujícího diskrétního prostoru,
  - \+ je zaručena bezkoliznost trajektorie na úrovni jednotlivých stavů.

- Zápory:
  - metoda sama nedefinuje způsob, jak provést kontrolu kolizí dráhy mezi dvěma stavy,
  - není definována metoda prohledávání grafu, a tedy není zaručena časová složitost,
  - není stanovena heuristika pro ocenění jednotlivých uzlů v prohledávacím grafu pro použití informované prohledávací metody, použití slepé prohledávací metody vede na příliš velkou časovou složitost,
  - k nalezení trajektorie je potřeba prohledávat velké množství stavů, což způsobuje potenciálně velkou časovou i paměťovou složitost,
  - není jednoznačně určený počet akcí ve zjednodušeném akčním prostoru U_d, jejich počet je při tom rozhodující o počtu stavů, které bude nutné prohledávat a o přesnosti vypočítané trajektorie,
  - protože metoda pracuje v diskrétním prostoru, který je aproximací původního spojitého prostoru a protože omezený diskrétní akční prostor U_d omezuje manévrovací možnosti letadla, nemusí být nalezená trajektorie zcela optimální, ale bude s optimální trajektorii blížit.

Metoda vyhledává řešení systematickým prohledáváním stavů v prohledávacím grafu. Pro případ, že řešení neexistuje musí být stanoveny ukončující podmínky, aby prohledávání neprobíhalo zbytečně dlouho nebo donekonečna. Pokud však řešení existuje s ohledem na provedené vzorkování je metodou nalezeno s časovou složitostí danou prohledávací metodou.

Řešení nalezené touto metodou je (téměř) optimální a bezkolizní. Proto je tato metoda vhodná pro plánování trajektorie dle zadání.

3.1.2 A Real-time 3D motion planning and simulation scheme for nonholomic systems

Tato metoda je určena pro vyhledávání trajektorie k cíli v prostoru mezi překážkami, o kterých získává informace průběžně ze senzorů letadla, například radaru. Metoda je určená pro práci v reálném čase a pro práci z průběžně získávanými informacemi. Metoda počítá s tím, že na začátku výpočtu nejsou známy vůbec žádné informace o okolí letadla, kromě malé vzdálenosti, kterou letadlo senzory zmapuje. Pro výpočet trajektorie z počátku až do cíle z informací dodaných před výpočtem lze metodu využít, pokud je nastavena vzdálenost dosahu senzorů letadla na dostatečně velkou, aby letadlo dohlédlo až do cíle.

Vlastnosti této metody pro výpočet trajektorie jsou:

- Klady:
  - \+ hledá trajektorii ve spojitém prostoru, nalezená trajektorie je také spojitá,
  - \+ nalezená trajektorie je hladká, takže může představovat reálnou trajektorii letadla, které není schopné okamžitě změnit zakřivení dráhy pohybu z rovného letu na maximální zatáčení,

    změna rychlosti zakřivení však není definována,
  - \+ časová složitost výpočtu je nižší, což je dáno i tím, že metoda je určená pro práci v reálném čase.

- Zápory:
  - vzhledem k výpočtu trajektorie se letadlo nemůže pohybovat po kružnici,
  - k výpočtu optimální trajektorie by bylo nutné vypočítat nekonečně mnoho segmentů trajektorie, což není v praxi možné,
  - nalezená trajektorie obecně není optimální. Metoda počítá s tím, že letadlo předem nezná všechny informace o překážkách, tedy optimální cestu ani nalézt nemůže, avšak ani v případě, že informace jsou dopředu známé, nevypočítává letadlo trajektorii (vzhledem k odvozovacím pravidlům DAB) tak, aby byla optimální.

Časová složitost metody je lepší než složitost předchozí metody, navíc nalezená trajektorie odpovídá reálným vlastnostem letadla více, díky tomu, že její zakřivení se mění v čase lineárně. Cílem této metody však není nalézt optimální cestu a nalezená trajektorie obecně není optimální. Proto není tato metoda vhodná pro výpočet trajektorie dle zadání.

3.1.3 Efficient Two-phase 3D motion planning for small fixed-wing UAVs

Metoda pracuje ve dvou fázích. První fáze je určena pro výpočet přibližné optimální trajektorie na základě informací o celém prostoru, které nemusí být kompletní a mohou se v průběhu letu letadla měnit. Ve druhé fázi je zpřesněn kratší úsek dříve vypočítané trajektorie na základě přesných a aktuálních informací o blízkém okolí letadla. V této diplomové práci předpokládám prostředí, které se v průběhu letu letadla nemění. V takovém případě je potřeba provést druhou fázi výpočtu pouze pro zamezení kolize s malými překážkami, které první fáze výpočtu zanedbává. Protože informace o celém prostoru jsou k dispozici na začátku výpočtu, je možné první i druhou fázi plánování trajektorie provést hned na začátku a vypočítat celou trajektorii až do cíle.

Dvoufázová metoda pro plánování trajektorie má tyto vlastnosti v porovnání s ostatními:

- Metoda obsahuje záložní řešení "Emergency stop"(viz obrázek 2.3) pro případ, že letadlo nenaplánuje včas a správně další úsek cesty.

  není však určeno, jakým způsobem má záložní řešení plánovat,

  pro tuto diplomovou práci nemá toto plánování význam, protože plánování trajektorie probíhá před tím, než se letadlo vydá na cestu,

- Klady:
  - \+ pro prohledávání grafu stavů je určena prohledávací metoda A^* zaručující optimálnost nalezené cesty v první fázi výpočtu,
  - \+ je určena heuristika pro ocenění každého stavu předpokládanou vzdáleností do cíle, a to přesnější než Euklidovská vzdálenost.
- Zápory:
  - lokální plánování vytváří prohledávací graf podobným způsobem jako metoda Sampling-based planning, pro jeho prohledávání je použita metoda Greedy search, která nemusí nalézt optimální řešení,

protože lokální plánování spojuje pouze dva sousední body cesty z globálního plánování, které je optimální, nebude odchylka lokální cesty velká.

Tuto metodu je možné využít tak, že informace jsou zadané jednorázově a trajektorie vypočítána před zahájením letu letadla. Časová složitost je daná složitostmi prohledávacích metod A^* a Greedy search použitých pro hledání cesty. Nalezená cesta se blíží optimální a je bezkolizní. Tato metoda je vhodná pro plánování trajektorie podle zadání.

3.1.4 Time-optimal paths for a Dubins airplane

Tato metoda pouze rozšiřuje dvourozměrný výpočet trajektorie Dubinsova auta do třetího rozměru. Pro naplánování trajektorie musí být předem proveden výpočet trajektorie v horizontální rovině, tedy obdobný jako výpočet trajektorie Dubinsova auta, který je poté upraven přidáním třetího rozměru. Za předpokladu, že nalezená trajektorie v horizontální rovině je optimální, je pak také trojrozměrná trajektorie letadla optimální. Při výpočtu trajektorie ve třetím rozměru ovšem není zaručeno vyhýbání se překážkám, které ve dvourozměrné horizontální rovině nemusí být patrné či mohou být nepřesné.

Protože metoda nedokáže zaručit bezkoliznost, není vhodná pro plánování trajektorie s vyhýbáním se překážkám, které požaduje zadání této diplomové práce.

3.2 Metoda zvolená pro implementaci

Pro implementaci plánování trajektorie s vyhýbáním se překážkám podle zadání jsou z popsaných čtyř metod vhodné dvě. Jsou to metody Sampling-based planning [4], která nalezne bezkolizní trajektorii blížící se optimální, a Efficient two-phase 3D motion planning for small fixed-wing UAVs [3], která rovněž nalezne bezkolizní trajektorii blížící se optimální. Obě dvě tyto metody zanáší do výpočtu trajektorie chybu vzniklou vzorkováním spojitých veličin, přesto jsou nalezené trajektorie téměř optimální. Ostatní metody nejsou vhodné protože neplánují trajektorii způsobem, který vyžaduje zadání práce. Metoda A real-time 3D motion planning and simulation scheme for nonholomic systems [8] plánuje trajektorii, která není optimální a metoda Time-optimal paths for a Dubins airplane [2] nemůže zaručit bezkoliznost trajektorie v přidaném třetím rozměru.

Ze dvou metod vyhovujících zadání jsem se rozhodl pro implementaci plánování trajektorie použít metodu Efficient two-phase 3D motion planning for small fixed-wing UAVs [3]. Tato metoda provádí vzorkování spojitého prostoru, který nahrazuje diskrétní mřížka. Nepřesnost trajektorie takto vypočítané v první fázi však zpřesňuje výpočet probíhající ve druhé fázi. Tehdy je upřesněna trajektorie pomocí pohybových primitiv. Takto dosažené výsledky dávají přesnější výpočet výsledné trajektorie, což je hlavní důvod výběru této metody.

Kapitola 4

Návrh aplikace

V této kapitole se zaměřím na návrh aplikace vyhledávající trajektorii pro letadlo napříč prostorem. Na samotnou činnost aplikace, její vstupy a výstupy, uživatelské rozhraní, také na strukturu programu a způsob implementace metody hledání trajektorie. Vzhledem k tomu, že kromě nalezení trajektorie, je dalším úkolem také její vizualizace, a to nejlépe v prostoru, ve kterém je nalezena, se také zaměřím na způsob, jakým bude trajektorie vizualizována.

Hlavním úkolem, který bude aplikace vykonávat je prohledání definovaného prostoru, nalezení optimální trajektorie z definované startovní pozice v tomto prostoru do definované cílové pozice. Před zahájením výpočtu bude potřebovat vstupní údaje a po ukončení výpočtu vytvoří výstup, který bude znázorňovat nalezenou trajektorii v určeném prostoru. V průběhu výpočtu nepotřebuje aplikace žádné dodatečné údaje a zároveň také nemá co nabídnout uživateli jako výstup. Aplikace tedy není interaktivní s uživatelem ani v průběhu výpočtu, ani před jeho zahájením, pouze posbírá vstupní údaje a zahájí výpočet.

Samotná aplikace rovněž nemusí provádět vizualizaci nalezené trajektorie. Vhodný formát výstupních dat, tedy nalezené trajektorie a popis prostoru, ve kterém byla nalezena, umožní vizualizovat trajektorii nástrojem dostupným pro takový formát dat. Vzhledem k tomu, že aplikace není interaktivní a nemá grafický výstup, není vhodné vytvářet pro aplikaci grafické uživatelské rozhraní. Program tedy bude fungovat jako konzolová aplikace.

Pro vlastní výpočet bude využita dvoufázová metoda pro 3D plánování pohybu Efficient Two-phase 3D motion planning for small fixed-wing UAVs popsaná v kapitole 2.3. Tato metoda nalezne optimální cestu od startu do cíle v rámci diskrétního prostoru, jako aproximace skutečného spojitého prostoru. Od přesného znění metody se však bude výpočet aplikace odchylovat. Protože není žádáno, aby aplikace pracovala v reálném čase a brala do úvahy pohybující se překážky, bude hlavní dílo metody spočívat v první fázi, globálním plánování. Druhá fáze metody, která má za úkol především upravovat trajektorii vypočítanou v první fázi podle pohybujících se překážek a malých překážek zanedbaných v první fázi, nebude pracovat v reálném čase a nebude muset brát v úvahu pohybující se překážky. Ve druhé fázi plánování tedy bude hlavní zasadit trajektorii vypočítanou v první fázi, tedy v diskrétním prostoru, do spojitého prostoru. Blok pro nouzový únik při selhání plánování, blok Emergency stop, uvedený na obrázku 2.3 není nutný a nebude v aplikaci implementován vůbec.

4.1 Vstupy a výstupy aplikace

Aplikaci budou před zahájením výpočtu předána vstupní data formou argumentů programu a vstupních souborů. Argumenty programu rovněž budou určovat způsob vytvoření výstupu a umístění výstupních souborů. Aplikace potřebuje pro výpočet následující vstupní data:

- Definice vstupního prostoru.
- Určení způsobu vytvoření výstupu.
- Parametry letadla:
  - minimální poloměr zatáčení,
  - maximální povolený úhel stoupání a klesání,
  - velikost letadla.
- Startovní pozice a počáteční směr.
- Cílová pozice a požadovaný směr.

4.1.1 Číselné hodnoty

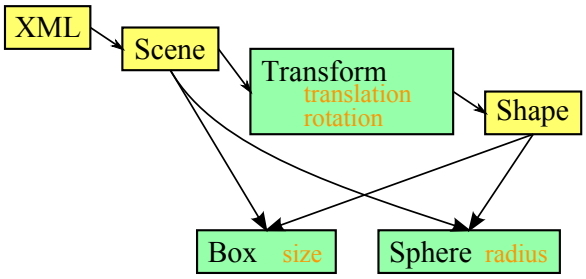
Letadlo, pro které se vyhledává trajektorie, se považuje za neholonomní objekt pohybující se pouze dopředu, zatáčející doleva či doprava v horizontální rovině a stoupající či klesající s definovaným okamžitým náklonem. Vlastnosti letadla jsou předem dané, zbývá určit konkrétní číselné hodnoty. Minimální poloměr letadla určuje schopnost letadla zatáčet. Protože pro výpočet dráhy není podstatná rychlost pohybu letadla, uvažuje se minimální poloměr zatáčení, kterého je letadlo schopné dosáhnout při optimální rychlosti a při výpočtu se považuje za konstantní. Podobně se za konstantní považuje maximální povolený úhel stoupání a klesání. Jako údaj o velikosti letadla je důležitá především jeho šířka, definující o kolik musí střed letadla minout překážku, aby nedošlo ke kolizi. Délka letadla není pro zjišťování kolizí podstatná, vzhledem k tomu, že se letadlo pohybuje vždy dopředu, nikdy bokem. Výška letadla také ovlivňuje jeho schopnost vyhýbat se překážkám, nicméně výška letadla je výrazně menší než jeho šířka. Pokud tedy střed letadla mine překážku dostatečně daleko vzhledem k šířce letadla, je tato vzdálenost dostačující i vzhledem k jeho výšce.

Startovní a cílová pozice spolu se směrem představuje několik číselných hodnot. Pro každou pozici je potřeba definovat souřadnice bodu v prostoru (tři hodnoty), směr letu v horizontální rovině (kurz) a momentální náklon stoupání. Pro každou pozici je tedy potřeba definovat celkem pět číselných hodnot. Pro určení cílové pozice je také potřeba definovat toleranci, jak přesně se musí letadlo do cíle trefit.

Parametry letadla a pozice představují číselné hodnoty, které je možné programu předávat jako jednotlivé argumenty. Protože se jedná o větší počet hodnot, jejichž zadávání při spuštění aplikace do příkazového řádku by mohlo být zdlouhavé, je vhodnější zadávat tyto údaje v konfiguračním souboru, který aplikace po spuštění přečte.

4.1.2 Definice vstupního prostoru

Prostor, ve kterém se bude vyhledávat trajektorie od startovní pozice k cílové, tedy vstupní prostor, je spojitý trojrozměrný prostor, kde každý bod je definován kartézskými souřadnicemi $\langle x, y, z \rangle$. Rovina xy představuje vodorovnou rovinu, souřadnice $\langle x, y \rangle$ tedy představují



Obrázek 4.1: Zjednodušená struktura X3D tagů pro načítání geometrie překážek.

horizontální pozici bodu, $z-$ová souřadnice představuje výšku. Prostor jako takový nemá omezenou velikost, letadlo se může pohybovat kdekoliv včetně míst se zápornou výškou, pokud tomu nebrání překážka. Ve vstupním prostoru jsou především definovány překážky. Jednotlivé překážky budou definovány jako geometrické objekty umístěné v prostoru. Vstup aplikace představuje seznam těchto geometrických objektů.

Protože výstupem aplikace je stejný prostor, liší se pouze předáním znázornění trajektorie, je vhodné aby oba tyto prostory byly popisovány stejným způsobem. Pro formát popisující oba tyto prostory je důležité, aby umožňoval:

- definici překážek jako geometrických objektů,
- znázornění nalezené trajektorie vhodným způsobem,
- animaci pohybu objektu po této trajektorii,
- prohlížení, případně editaci pomocí dostupného nástroje.

Vhodným formátem pro popis prostoru je 3D grafický formát X3D.

4.1.3 Formát X3D

Jedná se o otevřený textový formát pro popis 3D vektorové grafiky [9]. Umožňuje popsat 3D scénu jako kolekci geometrických objektů s definovanými vlastnostmi. Podporuje také statické geometrické transformace objektů, animace objektů, práci s osvětlením, materiály povrchů, reflexní vlastnosti, práce s normálami a spoustu dalších funkcí. Soubor ve formátu X3D může být popsán buď s využitím syntaxe jazyka VRML nebo syntaxe XML. Protože XML je všeobecný snadno čitelný formát, bude navrhovaná aplikace pracovat s X3D soubory využívající syntaxi XML.

X3D je rozsáhlý formát, ze kterého bude v rámci aplikace pro vyhledávání trajektorie využitá jen část jeho funkcí. Pro aplikaci je důležitá především geometrie jednotlivých objektů představujících překážky. Další vlastnosti těchto objektů (barva, odraz světla, apod.) nejsou podstatné pro zjišťování kolizí a aplikace je nepotřebuje načítat. Zjednodušená struktura XML X3D souboru z pohledu aplikace je uvedená na obrázku 4.1, příklad XML X3D souboru je uveden v ukázce 4.1.

Pro výstupní soubor znázorňující nalezenou trajektorii je potřeba využít také animace. Protože se ale jedná o animaci, která bude vždy podle stejné šablony jen se bude měnit umístění jednotlivých bodů podle nalezené trajektorie, bude v programu vytvořen blok, který do XML X3D souboru přidá několik tagů vždy stejným způsobem, pouze s odlišnými

Ukázka kódu 4.1: Příklad XML X3D kódu (zjednodušený).

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D>
        <Scene>
                <Transform translation='0 0 -5'>
                        <Shape>
                                <Box size='1000 1000 10'/>
                        </Shape>
                </Transform>
                <Transform translation='0 0 5'>
                        <Shape>
                                <Sphere radius='10'/>
                        </Shape>
                </Transform>
        </Scene>
</X3D>
```

souřadnicemi jednotlivých řídících bodů animace. Pro vytvoření výstupu bude vytvořena kopie vstupního souboru s prostorem. Do této kopie bude přidán blok tagů znázorňujících trajektorii staticky a blok tagů definujících animaci objektu pohybujícího se po této trajektorii.

4.1.4 Vizualizace X3D

Pro vizualizaci zvoleného grafického formátu X3D je vhodný nástroj z frameworku InstantReality [6], prohlížeč X3D scén, Instant Player [6]. Framework InstantReality je volně přístupný pro nekomerční použití, obsahuje nástroje pro práci s X3D soubory, příklady X3D souborů a také program Instant Player pro prohlížení X3D souborů. Instant Player umožňuje prohlížení souborů ve formátu X3D v XML i VRML syntaxi, provádí animace ve scéně, a proto je vhodným nástrojem pro vizualizaci trajektorie a animace pohybu po trajektorii ve výstupním X3D souboru.

4.2 Struktura aplikace

Výpočet hledání trajektorie bude rozdělen do čtyř následujících hlavních částí:

1. Načtení vstupních údajů.
2. První fáze metody výpočtu – globální plánování.
3. Druhá fáze metody výpočtu – lokální plánování
4. Vytvoření výstupu, jeho přizpůsobení výstupnímu formátu.

Při načítání vstupních údajů aplikace přečte a dekóduje argumenty a vstupní soubory. Pokud bude zaznamenána chyba nebo nekompletnost některých dat, ukončí se aplikace s chybou, pokud ne, zahájí se výpočet. Globální plánování je hlavní a nejsložitější část výpočtu, výsledkem je nahrubo určená trajektorie napříč prostorem. Lokální plánování tuto

trajektorii zpřesní, a výsledkem bude spojitá přesná trajektorie od startu k cíli. Nalezená trajektorie bude zapsána do výstupního XML X3D souboru spolu s animací pohybu objektu po této trajektorii.

4.2.1 Globální plánování

První fáze dvoufázové metody hledání trajektorie zahrnuje:

1. Vytvoření diskrétní mřížky aproximující spojitý prostor.
2. Zanesení startovní a cílové pozice do mřížky.
3. Vytvoření a prohledání stavového prostoru v diskrétní mřížce.
4. Sestavení cesty mřížkou pro další fázi.

Použitá metoda plánování trajektorie Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] definuje rozměry diskrétní mřížky takto:

$$dx = dy = \frac{2}{3} \cdot r_{min}$$

$$dz = dx \cdot \operatorname{tg} \alpha$$

kde dx je velikost buňky diskrétní mřížky v ose x, dy v ose y a dz v ose z, r_{min} je minimální poloměr zatáčení a α je maximální povolený úhel stoupání či klesání. Velikost buňky, a tedy jemnost diskrétní mřížky, je záměrně závislá na manévrovacích možnostech letadla.

Protože startovní a cílová pozice jsou definovány ve spojitém prostoru je potřeba je zanést do diskrétní mřížky. V obou případech to znamená nalezení konkrétní buňky v mřížce, do které tyto pozice spadají. V diskrétní mřížce je také potřeba vytvořit stavový prostor. Jednotlivé stavy reprezentují pozici letadla v některé z buněk diskrétní mřížky a směr, kterým se letadlo pohybuje. Zpracováváním těchto stavů metoda postupně prohledá všechny možné cesty, až nalezne takovou, která vede do cíle. Taková nalezená cesta představuje cestu letadla diskrétní mřížkou. Pro prohledávání stavového prostoru se využívá metoda A^*, která zajišťuje, že nalezená cesta je optimální, tedy nejkratší možná v rámci diskrétní mřížky. Zároveň díky ohodnocování jednotlivých stavů přednostně prohledává cesty směřující více k cíli, což zlepšuje složitost vyhledávání.

Během hledání cesty jsou prověřovány prohledávané buňky diskrétní mřížky, zda nejsou v kolizi s některou z překážek. Jsou ignorovány překážky dostatečně malé, aby se jim letadlo mohlo vyhnout i v rámci jediné buňky diskrétní mřížky. Takové překážky budou brány v úvahu ve druhé fázi plánování.

4.2.2 Lokální plánování

Na základě výsledků předchozí fáze je úkolem lokálního plánování vypočítat přesnou trajektorii ve spojitém prostoru propojující startovní a cílovou pozici. Vstupem je cesta diskrétní mřížkou. Lokální plánování iterativně hledá cestu vždy do následující buňky diskrétní mřížky a z ní zase do další. Takto se postupuje až do cíle. Protože lokální plánování vyhledává trajektorii částmi prostoru, které na bezkoliznost prověřilo globální prohledávání, není třeba ověřovat, zda došlo ke kolizi z překážkami, s výjimkou malých překážek, které globální plánování ignoruje.

Při hledání cesty z jedné buňky diskrétní mřížky do následující je vytvářen stavový prostor pomocí pohybových primitiv. Pohybová primitiva definují možný pohyb letadla ve spojitém prostoru vždy na malou vzdálenost dopředu. K prohledávání stavového prostoru se využívá metoda Greedy-search, která podobně jako metoda A^* využívaná v globálním plánování prohledává cesty směřující k cíli a tím snižuje složitost výpočtu.

Kapitola 5

Implementace

Tato kapitola bude pojednávat o implementaci aplikace podle návrhu z předchozí kapitoly. Bude popsáno, jakým způsobem je implementována aplikace pro vyhledávání trajektorie v prostoru, použitá plánovací metoda a odchylky implementace od přesného znění této metody. Budou popsány algoritmy využité při vyhledávání trajektorie i jejich implementace a případná omezení. Také bude popsán formát vstupních a výstupních dat, načítání a ošetřování vstupních dat, jejich přesný formát, způsob vytváření výstupu a možnost vizualizace výstupních dat.

Aplikace pro vyhledávání trajektorie je implementována jako konzolová aplikace, předpokládá se spouštění z příkazového řádku se zadáním příslušných argumentů pro spuštění programu. K vývoji aplikace jsem použil programovací jazyk Java. Dáno programovacím jazykem, aplikace je interpretovaná, k jejímu spuštění je tedy třeba interpret jazyka Java, pro vývoj jsem použil verzi 1.7.0_09. Výstupem aplikace je nalezená trajektorie zaznačená ve výstupním souboru. K vizualizaci výstupu je potřeba použít vhodný nástroj pro vizualizaci souborů ve formátu X3D (viz kapitola 4.1.3).

Výpočet aplikace je rozdělen do několika základních částí. Každá z těchto částí má vlastní vstup a dílčí výstup pro další část výpočtu. Podle návrhu z předchozí kapitoly je výpočet rozdělen na následující části:

1. Načítání vstupních dat.
   - detekce argumentů,
   - parsování vstupních souborů,
   - načtení konfigurace, inicializace dalšího výpočtu,
   - zpracování načtených geometrických dat.
2. Globální plánování.
   - první fáze použité metody plánování,
   - vytvoření diskrétní mřížky a práce v ní,
   - hledání hrubé trajektorie v diskrétní mřížce.
3. Lokální plánování.
   - zpřesnění předchozí hrubé trajektorie ve spojitém prostoru,
   - hledání dílčích trajektorií mezi jednotlivými body hrubé trajektorie.

4. Vytvoření výstupu.
   - úprava zpřesněné spojité trajektorie pro zápis,
   - kopírování vstupního souboru, zápis trajektorie a animace trajektorie,
   - případné vytvoření logovacího souboru.

Jednotlivé části výpočtu jsou prováděny postupně v uvedeném pořadí. Každá z těchto částí má svůj vlastní dílčí vstup a výstup. Vstupem dané části programu je výstup předchozí části. Každá z uvedených částí výpočtu představuje několik činností, podrobněji budou všechny tyto činnosti popsány v následujících podkapitolách.

Aplikace potřebuje ke svému běhu správně zadaná vstupní data. Zejména je nutné, dodržení správných formátů vstupních dat a jejich korektnost. Pokud nejsou vstupní data zadána správně nebo je aplikace z jiného důvodu nemůže přečíst (například odepření přístupu k souboru) nemůže pokračovat ve výpočtu. Protože se nejedná o interaktivní aplikaci, při takto vzniklé chybě se aplikace pouze ukončí a upozorní textovým výpisem na nekorektnost vstupních dat. Předpokládá se opětovné spuštění aplikace po odstranění chyby. Chybový výstup aplikace představuje textový řetězec definující chybu, který aplikace vypisuje na standardní chybový výstup.

5.1 Načtení a zpracování vstupních dat

Data, která aplikace zpracovává jsou buď číselnými hodnotami nebo jsou strukturována do větších celků a zapsána v souborech. Aplikaci jsou data předávána jako argumenty zadané při spuštění programu a v souborech, které aplikace v přípravné fázi výpočtu přečte. Při spuštění jsou aplikaci předávány následující argumenty:

- --input prostor.x3d

  Určuje soubor ve formátu XML X3D, který obsahuje prostor s překážkami.

- --output soubor.x3d

  Určuje výstupní soubor, do kterého bude zapsána nalezená trajektorie. Soubor bude také ve formátu XML X3D.

- --config konfigurace.xml

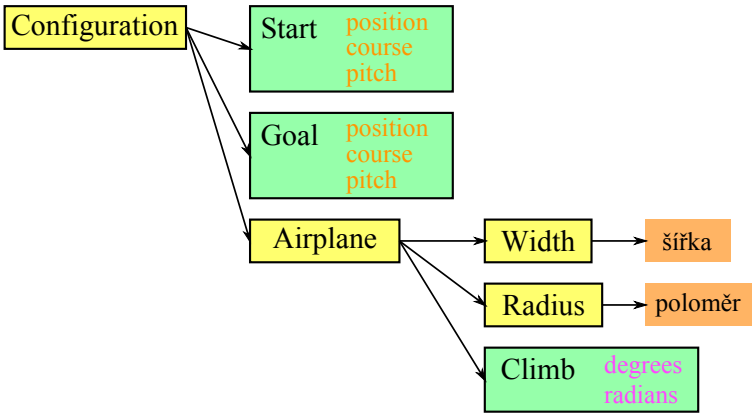
  Určuje soubor ve formátu XML obsahující parametry letadla a další podrobnější data pro výpočet.

- -log informace.txt

  Určuje, zda má být vytvořen logovací soubor a pokud ano, určuje jeho umístění.

- --help – vypisuje stručnou nápovědu ohledně spuštění aplikace. Pokud je zadán tento argument, ostatní argumenty se nevyžadují a aplikace neprovádí žádný výpočet.

- -cz – přepne výpisy aplikace do českého jazyka, jinak zůstanou v jazyce anglickém.



Obrázek 5.1: Struktura XML souboru pro konfiguraci výpočtu.

Samotné argumenty nepředávají aplikaci užitečná data, pouze upravují činnost aplikace a především určují, odkud aplikace vlastní data přečte.

Vlastní data jsou uvedena v souborech a jsou strukturována v přesném formátu. Pro načítání informací o letadle a konfigurace výpočtu je použit soubor ve formátu XML. Pro načítání informací o prostoru, ve kterém se bude letadlo pohybovat je to další soubor ve formátu X3D využívající syntaxi XML. Tento soubor nemusí obsahovat pouze informace potřebné aplikací, ale podle pravidel X3D formátu může obsahovat i další grafické informace o popisu scény, jako barvy objektů, osvětlení, apod. Aplikace z tohoto souboru přečte pouze data potřebná pro výpočet, ostatní data se ignorují.

5.1.1 Konfigurační soubor

Soubor ve formátu XML obsahující konfiguraci výpočtu a parametry letadla, pro které se vyhledává trajektorie. Vnitřní struktura souboru byla navržena pro tuto konkrétní aplikaci a pro její správné fungování je nutné strukturu přesně dodržet. Struktura tohoto XML souboru je znázorněna na obrázku 5.1. Na obrázku jsou znázorněné jednotlivé tagy, které soubor obsahuje a hodnoty, které jsou v nich případně zadány.

K parsování obsahu XML souboru jsem v aplikaci využil DOM [7], standardní knihovnu pro jazyk Java, volně přístupnou. DOM načítá a zpracovává jakékoli soubory ve formátu XML. S využitím této knihovny aplikace čte obsah souboru po jednotlivých tazích a na definovaných místech ve struktuře hledá určité hodnoty. Jak znázorňuje obrázek 5.1, jsou některé hodnoty obsaženy jako atributy tagů, jiné jsou uvedeny jako textové (číselné) hodnoty v tělech tagů.

Z obrázku 5.1 je patrné, že pro konfiguraci výpočtu aplikace jsou zapotřebí následující data v definovaných formátech, jsou uvedeny podle tagů, které daná data obsahují:

- Start – obsahuje informace o startovní konfiguraci letadla:

  position – určuje počáteční pozici letadla v prostoru, obsahuje tři kartézské souřadnice,

  course – určuje počáteční směr letu letadla horizontální směrem (kurz), obsahuje číselnou hodnotu udávající úhel, jednotkou jsou radiány,

pitch – určuje náklon stoupání letadla, náklon stoupání je rovněž uveden jako úhel, který letadlo svírá s horizontální rovinou, kladný úhel představuje stoupání, záporný úhel představuje klesání, úhel musí být vždy ostrý (tedy menší než pravý úhel), jednotkou jsou radiány.

- Goal – podobně jako Start obsahuje informace o požadované cílové konfiguraci letadla.
- Width – určuje šířku letadla, ta se využívá pro stanovení odstupu, jaký si musí střed letadla udržet od překážek, aby nedošlo ke kolizi.
- Radius – určuje minimální možný poloměr zatáčení letadla.
- Climb – určuje maximální povolený úhel stoupání, stejná hodnota určuje také maximální úhel klesání, může být definován buď ve stupních nebo v radiánech:

  degrees – úhel zadaný ve stupních,

  radians – úhel zadaný v radiánech.

V prostoru jsou vzdálenosti, rozměry a velikosti definovány čísly bez jednotky. Vzhledem k tomu, že se tato čísla vždy porovnávají se stejným měřítkem, není podstatné, jaká je jednotka těchto rozměrů. Musí však být použita stejná jednotka pro všechny rozměry, které se v prostoru používají. Aby byla vždy použita stejná jednotka, doporučuje se proto pro daný výpočet definovat použitou jednotku, například základní jednotku metr.

5.1.2 Definice prostoru

Dalším vstupem potřebným pro výpočet je prostor, ve kterém se bude vyhledávat trajektorie. Samotný prostor je nekonečně velký a spojitý, pro pohyb letadla neklade jiná omezení, než ta vyplývající z vlastností a konkrétních parametrů letadla. V prostoru se však vyskytují překážky, kterým se musí letadlo vyhnout. Jak bylo zmíněno dříve v tomto textu, prostor je definován ve formátu X3D. K načítání obsahu XML X3D souboru je opět využita knihovna DOM [7]. Použití této knihovny znamená, že se celý obsah souboru načte do paměti, kde se provede jeho parsování. Aplikace potom z definovaných tagů čte hodnoty popisující překážky.

Strukturu tagů v XML X3D souboru, zjednodušenou pouze na geometrické hodnoty, ukazuje obrázek 4.1. Z tohoto obrázku je patrné, že překážky mohou být buď ve tvaru koule nebo kvádru. Samotné tagy Sphere[1] a Box[2] obsahují pouze rozměry geometrického objektu. V případě koule je to její poloměr, pro kvádr jsou definovány tři rozměry velikosti (v_x, v_y, v_z) pro každou souřadnici. Umístění těchto objektů v prostoru určuje tag Transform, ve kterém je tag příslušného objektu zanořen. Tag objektu nemusí být uvnitř žádného tagu Transform, pak na něj není žádná transformace aplikována. Takový objekt se nachází ve středu souřadnicového systému.

Objekty nacházející se jinde než ve středu souřadnicového systému musí být posunuty, a to aplikováním geometrické transformace. Tag Transform definuje geometrickou transformaci, použitou na všechny objekty uvnitř tohoto tagu. Transformace může být buď posun (*translation*) nebo rotace (*rotation*). Posun p je transformace, která posune střed objektu

[1] Koule
[2] Kvádr

o definovanou vzdálenost od středu souřadnicového systému, je definován třemi souřadnicemi pro posun v každé ose: (p_x, p_y, p_z). Rotace r otáčí objekt okolo jeho středu o definovaný úhel α podle definované osy: (r_x, r_y, r_z, α).

Překážky umístěné v prostoru jsou geometrické objekty definované jejich geometrií a konkrétními rozměry. Definice prostoru je aplikací chápána jako kolekce překážek. Všechny tyto překážky jsou ze souboru přečteny a jejich umístění, geometrie a rozměry jsou uloženy pro použití v dalším výpočtu. Výsledkem načítání vstupního XML X3D souboru je tedy seznam těchto překážek.

Při vytváření seznamu překážek je brán v úvahu způsob využití těchto dat při dalším výpočtu. Seznam překážek je potřebný pro detekci kolizí při hledání trajektorie. Načtená data jsou tedy upravována tak, jak jsou vhodná pro další použití. Načítání překážek tedy spočívá v následujících krocích

1. K rozměrům překážky je připočítána geometrická transformace upravující její umístění v prostoru a otočení.

2. Rozměry každé překážky jsou zvětšeny o velikost letadla a o toleranci, tedy o hodnotu určující o jakou nejmenší vzdálenost musí střed letadla minout skutečnou překážku, aby nedošlo ke kolizi.

3. Je vypočítán geometrický objekt, který se bude používat pro detekci kolize:

   - Překážka tvaru koule je umístěna do prostoru na základě posunu dle geometrické transformace, rotace se nebere v úvahu.
   - Protože pro překážku tvaru kvádru je potřeba brát v úvahu také rotaci, je vypočítán obalující kvádr, který je kolmý na všechny souřadnicové osy (tedy bez rotace) obsahující v sobě původní kvádr. V dalším výpočtu se bude pracovat výhradně s obalujícím kvádrem.

5.1.3 Výpočet obalujícího kvádru

Protože překážky jiného tvaru než koule mohou být libovolným způsobem rotovány, tedy otočeny o libovolný úhel libovolným směrem, je výpočet kolize výrazně složitější. Pro zjednodušení výpočtu kolize s překážkou je místo libovolně otočeného geometrického objektu vypočítáno obalující těleso. K výpočtu obalujícího objektu je využito metody AABB (Axis Aligned Bounding Box[3]) [1]. Podle této metody je vytvořen obalující kvádr kolmý na všechny souřadnicové osy, který je co nejmenší tak, aby v něm byl zcela obsažen původní objekt (překážka). Budoucí výpočet kolize s kvádrem kolmým na všechny souřadnicové osy je jednodušší a tedy rychlejší.

Výpočet obalujícího kvádru probíhá v následujících krocích:

1. Výpočet transformační matice pro rotaci.

2. Vypočítání skutečných souřadnic všech vrcholů původního objektu.

3. Nalezení minimální a maximální souřadnice na každé souřadnicové ose. Tím je určen interval na každé ose, který původní objekt zaujímá.

4. Výpočet středu a velikosti obalujícího kvádru.

[3] v českém překladu Osově orientovaný obalující kvádr

Ve formátu X3D jsou rotace definovány jako otočení objektu o daný úhel kolem dané osy. Příkladem může být otočení objektu o 90° kolem souřadnicové osy x definované jako $(1; 0; 0; 1,570796)$ (úhel je zadán v radiánech). Pro výpočet skutečných souřadnic každého vrcholu původního objektu je potřeba vynásobit původní souřadnice transformační maticí T:

$$(x, y, z) = (x_{puv}, y_{puv}, z_{puv}) \cdot T$$

Proto je potřeba na základě rotace definované osou rotace a úhlem vypočítat transformační matici. Podle [10] probíhá výpočet matice T následovně, uvažme že osa rotace je $n = (n_x, n_y, n_z)$ a úhel otočení je α:

$$T = \begin{pmatrix} \cos\alpha + n_x^2 \cdot (1 - \cos\alpha) & n_x \cdot n_y \cdot (1 - \cos\alpha) - n_z \cdot \sin\alpha & n_x \cdot n_z \cdot (1 - \cos\alpha) - n_y \cdot \sin\alpha \\ n_x \cdot n_y \cdot (1 - \cos\alpha) - n_z \cdot \sin\alpha & \cos\alpha + n_y^2 \cdot (1 - \cos\alpha) & n_y \cdot n_z \cdot (1 - \cos\alpha) - n_x \cdot \sin\alpha \\ n_x \cdot n_z \cdot (1 - \cos\alpha) - n_y \cdot \sin\alpha & n_y \cdot n_z \cdot (1 - \cos\alpha) - n_x \cdot \sin\alpha & \cos\alpha + n_z^2 \cdot (1 - \cos\alpha) \end{pmatrix}$$

Vynásobením souřadnic jednotlivých vrcholů původního objektu tedy získáme souřadnice těchto vrcholů po provedení rotace.

Pro další výpočty je potřeba definovat střed a rozměry obalujícího kvádru. Výsledkem výpočtu obalujícího kvádru jsou tedy souřadnice středu (s_x, s_y, s_z) a rozměry kvádru ve třech osách (x_k, y_k, z_k). Pro určení středu je potřeba nalézt minimum a maximum souřadnic rotovaných vrcholů původního objektu min_x, min_y, min_z, max_x, max_y a max_z. Pro velikost obalujícího kvádru pak platí:

$$\begin{aligned} x_k &= max_x - min_x \\ y_k &= max_y - min_y \\ z_k &= max_z - min_z \end{aligned}$$

Souřadnice středu obalujícího kvádru se pak rovnají:

$$\begin{aligned} s_x &= max_x - \frac{x_k}{2} \\ s_y &= max_y - \frac{y_k}{2} \\ s_z &= max_z - \frac{z_k}{2} \end{aligned}$$

5.2 Globální plánování

První fáze užité dvoufázové metody plánování trajektorie Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] popsané v kapitole 2.3 vyhledává (plánuje) trajektorii v diskrétním prostoru vytvořeném jako aproximace spojitého prostoru. Tento diskrétní prostor je představován diskrétní mřížkou, jejíž rozměry jsou stanovené podle parametrů letadla. Plánování trajektorie v diskrétní mřížce se označuje jako globální plánování, protože vyhledává globální trajektorii napříč celým prostorem od startovní pozice až do cílové.

Vstupem globálního plánování je:

- seznam překážek vyskytujících se v prostoru,
- parametry letadla:
  - minimální poloměr zatáčení,
  - povolený úhel stoupání či klesání,
  - velikost (šířka),

- tolerance, určující o jakou nejmenší vzdálenost musí letadlo minout překážku,
- startovní a cílová konfigurace letadla.

Na základě těchto vstupů je inicializována diskrétní mřížka, vytvořen počáteční stav a poté zahájeno prohledávání prostoru směrem k cíli. Práce s globální mřížkou a prohledávání prostoru popisují následující podkapitoly.

Výsledkem globálního plánování je seznam globálních stavů, tedy seznam buněk diskrétní mřížky, kterými letadlo proletí od startovní pozice do cílové. Seznam je seřazen v pořadí, v jakém letadlo buňky postupně proletí. Pro každý globální stav je určena nejen pozice v prostoru, ale také směr, kterým letadlo danou buňku diskrétní mřížky proletí. Na základě těchto globálních stavů je poté vyhledávána spojitá trajektorie při lokálním prohledávání. Lokální plánování je popsáno v kapitole 5.3.

5.2.1 Diskrétní mřížka

Pro zahájení plánování musí být nejprve vytvořena diskrétní mřížka, ve které se bude trajektorie hledat. Použitá metoda [3] definuje rozměry buněk diskrétní mřížky a pohyb letadla mezi jednotlivými buňkami. Pro globální plánování není důležité, kde v prostoru, který ohraničuje jedna buňka diskrétní mřížky se letadlo přesně nachází, pozice letadla je udávána souřadnicemi buňky v diskrétní mřížce. Rovněž není důležitý skutečný směr letu vyjádřený přesným úhlem, ale směr, kterým se letadlo pohybuje do příští buňky na své cestě.

Zásadní pro snadné vyhledávání trajektorie v diskrétní mřížce je správné stanovení rozměrů jednotlivých buněk mřížky. Označme velikost každé buňky v mřížce dx, dy a dz, každý rozměr definuje velikost buňky v dané ose. Pokud také označíme minimální poloměr otáčení letadla r_{min}, pak platí:

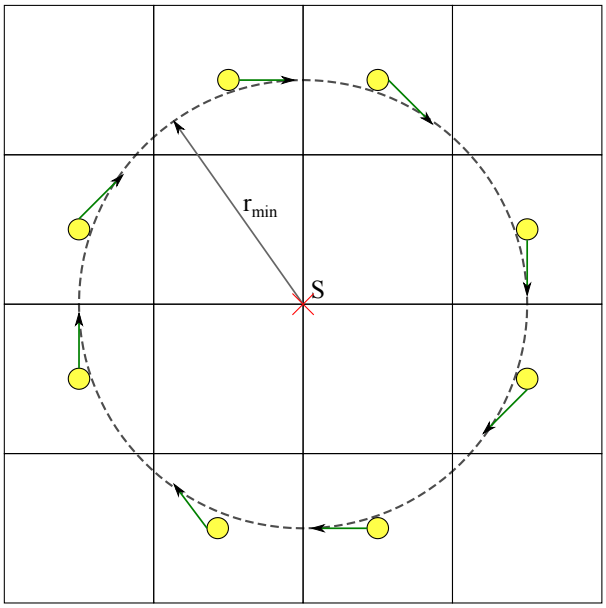
$$dx = dy = \frac{2}{3} \cdot r_{min}$$

V rámci horizontální roviny jsou tedy buňky čtvercové. Stanovení velikosti čtverce podle r_{min} umožní, aby v diskrétní mřížce letadlo zatáčelo na úrovni buněk, jak ukazuje obrázek 5.2. Letadlo může zatočit do další buňky, přičemž zatáčí přesně s poloměrem r_{min}. Z obdobného důvodu je také určen vertikální rozměr buňky dz jako (α značí povolený úhel stoupání):

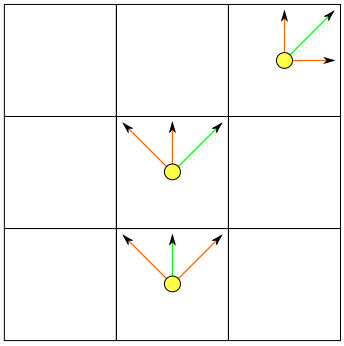
$$dz = dx \cdot \operatorname{tg} \alpha$$

Z obrázku 5.2 je také patrné, že stav letadla není udáván jen pozicí v mřížce, ale také směrem, kterým letadlo letí. Stav ve kterém se letadlo nachází v diskrétní mřížce je označen jako globální stav. Globální stav obsahuje souřadnice buňky diskrétní mřížky, ve které se nachází letadlo a označení směru, kterým letadlo poletí do dalšího stavu. Souřadnice určující pozici letadla v diskrétní mřížce, odpovídají souřadnicím ve spojitém prostoru. Tyto souřadnice označují střed buňky diskrétní mřížky, ve které se letadlo nachází. Směr, kterým může letadlo letět nabývá 8 hodnot, odpovídajících 8 sousedním buňkám, do kterých letadlo může přeletět. Vertikální směr letadla není v globálním stavu zaznamenáván, protože se předpokládá, že stoupání či klesání letadla může začít i přestat kdykoliv.

Pohyb letadla je tak rozdělen na horizontální, pro který platí omezení dané r_{min} a na vertikální, pro který platí omezení dané povoleným úhlem stoupání α. V horizontálním směru má letadlo teoreticky osm možných směrů letu, z nichž ale může v daném globálním stavu využít jen tři. Může pokračovat v předchozím směru letu nebo zatočit doprava nebo doleva. Směry, které připadají pro daný stav v úvahu ukazuje obrázek 5.3. Ve vertikálním



Obrázek 5.2: Zatáčení letadla v diskrétní mřížce.



Obrázek 5.3: Ukázka směrů, kterými může letadlo letět (zelená označuje zvolený směr).

směru má letadlo vždy tři možnosti letu, a to stoupat, udržovat výšku nebo klesat. Protože stoupání či klesání může začít i přestat kdykoli, má letadlo vždy k výběru všechny tři tyto možnosti bez ohledu na předchozí stav. Letadlo tedy může například přejít ze stoupání rovnou do klesání. Z daného globálního stavu má tedy letadlo vždy devět teoretických možností, do které další buňky diskrétní mřížky letět.

Do diskrétní mřížky je také potřeba zanést startovní a cílovou konfiguraci letadla, to znamená vytvořit startovní a cílový globální stav. Globální mřížka je posunuta tak, že startovní pozice odpovídá vždy středu buňky diskrétní mřížky, startovní pozice je tedy v mřížce určena přesně. Směr letu pro startovní a cílový globální stav je určen jako jeden z osmi možných směrů letu v diskrétní mřížce, a to jako směr nejbližší skutečnému směru letu ve spojitém prostoru. Směr letu tedy není určen přesně. Rovněž cílová pozice v diskrétní mřížce není určena přesně, protože mřížka je posunutá pro startovní pozici. Z cílové konfigurace je určena buňka, uvnitř které leží skutečná cílová pozice. Během globálního plánování se hledá trajektorie do takto nepřesně určeného cílového globálního stavu. Výsledná trajektorie je poté zpřesněna až ve fázi lokálního plánování.

5.2.2 Hledání globální cesty

Cesta, kterou vyhledává globální plánování v diskrétní mřížce je označena jako globální cesta. Jedná se o posloupnost globálních stavů, kterými letadlo postupně proletí včetně počátečního a cílového globálního stavu. Globální cesta také představuje konečný výstup fáze globálního plánování.

Hledání globální cesty spočívá v systematickém prohledávání stavového prostoru globálních stavů. K prohledávání je využito metody A^*, která zajišťuje nalezení optimální, tedy nejkratší, cesty a současně směruje prohledávání na stavy vedoucí blíže k cíli, což podstatně snižuje časovou složitost. Vzhledem k tomu, že z každého globálního stavu je možné přejít do devíti dalších globálních stavů, je použití jiné prohledávací metody prakticky vyloučeno, pro neúměrně vysokou časovou složitost. Algoritmus hledání globální cesty se pak skládá z následujících hlavních kroků:

1. Detekce kolize startovního a cílového globálního stavu,

   pokud je startovní nebo cílový stav kolizní, je výpočet ukončen, cesta neexistuje.

2. Vytvoření a inicializace seznamů OPEN, CLOSED a SEARCHED:

   (a) OPEN je seznam všech globálních cest, které připadají v úvahu pro prohledání. Na začátku obsahuje právě jednu globální cestu, cestu ze které musí letadlo začít svůj let, triviální cestu obsahující pouze startovní stav.

   (b) CLOSED je seznam nalezených stavů, kterými nesmí letadlo proletět, tedy stavů, u kterých byla již zjištěna kolize s překážkou. Na začátku je prázdný.

   (c) SEARCHED je seznam stavů, které již byly prohledány a přidány do některé z prohledávaných globálních cest. Na začátku je seznam prázdný, žádné stavy nebyly prohledány.

3. Pokud je seznam OPEN prázdný, je výpočet ukončen, cesta nebyla nalezena. Jinak výpočet pokračuje.

4. Nalezení nejlevnější globální cesty v seznamu OPEN. Hledá se globální cesta s nejmenší cenou.

5. Odebrání této cesty ze seznamu OPEN.
6. Pokud se aktuální stav cesty, tedy stav z něhož se hledá pokračování cesty, již vyskytuje v seznamu SEARCHED, dále se neprohledává a pokračuje se od bodu 3.
7. Aktuální stav prohledávané cesty je vložen do seznamu SEARCHED, tedy již nebude znovu prohledáván.
8. Pokud je aktuální stav shodný s cílovým stavem, je cesta do cíle nalezena. Optimální cestou do cíle je v tomto případě aktuálně zpracovávaná cesta.

   Výpočet je ukončen a nalezená globální cesta je předána do další fáze plánování.
9. Výpočet dalšího pokračování aktuální cesty:
   (a) Určení všech stavů, do kterých může aktuální cesta pokračovat.
   (b) Vyloučení stavů, které již byly vypočítány jako kolizní (stavy vyskytující se v seznamu CLOSED).
   (c) Detekce kolizí pro zbylé stavy, vyloučení stavů, které jsou kolizní. Kolizní stavy jsou přidány do seznamu CLOSED.
   (d) Vyloučení stavů, které již byly prohledány, tedy stavů vyskytujících se v seznamu SEARCHED.
   (e) Pro všechny zbylé stavy, představující možné pokračování aktuální cesty do cíle jsou vytvořeny nové cesty a přidány do seznamu OPEN k dalšímu prohledání. Pro všechny takto vytvořené cesty je vypočítáno jejich ohodnocení.
10. Zpracování aktuální cesty je dokončeno, pokračuje se zpracováním další cesty od bodu 3.

Seznam OPEN obsahuje všechny globální cesty, které připadají v úvahu pro prohledávání. Jedná se o cesty, které by mohly vést k cíli a pro které je potřeba najít pokračování, pokud existuje. Všechny tyto cesty začínají ve startovním stavu a končí v určitém stavu na cestě k cíli. Každá z těchto rozpracovaných globálních cest tedy představuje možný začátek optimální cesty do cíle. Pokud poslední stav takové cesty odpovídá pozicí i směrem cílovému stavu, je taková cesta výslednou cestou propojující startovní stav s cílovým.

Při výpočtu je také nezbytné detekovat kolize globálních stavů a vyloučit ty stavy, které kolidují s překážkou. Tím je zaručena bezkoliznost nalezené cesty. Všechny rozpracované cesty nacházející se v seznamu OPEN jsou bezkolizní. Protože výpočet pro detekci kolize globálního stavu s překážkou je náročný, jsou stavy, u kterých je zjištěna kolize uloženy do seznamu CLOSED. Pokud bude stav se shodnou pozicí, ale ne nutně stejným směrem, prohledáván v dalším výpočtu, místo výpočtu detekce kolize bude tento stav vyloučen na základě zjištění jeho přítomnosti v seznamu CLOSED. Samotné zjištění, zda se stav nachází v seznamu CLOSED je výpočetně výrazně jednodušší.

Pro výpočet je nezbytné rozlišovat, které cesty vedou blíže k cíli a které se od něj naopak vzdalují. Správné určení nejnadějnější cesty, která by mohla vést k cíli s nejkratší délkou je pro výpočet zásadní. Výrazně snižuje časovou složitost, protože se nemusí zbytečně prohledávat nesmyslné cesty vedoucí pryč od cíle nebo zbytečnou oklikou. Fakt, že pro další prohledání se vždy vybírá dosavadní nejlepší cesta navíc zaručuje, že výsledná nalezená globální cesta bude nejkratší možná, tedy optimální. Aby bylo možné rozlišit tyto cesty, je potřeba stanovit vhodnou heuristiku pro ocenění cesty.

5.2.3 Ocenění globální cesty

Ocenění globální cesty umožňuje rozlišit cesty vedoucí kratší vzdáleností k cíli od těch, které k cíli vedou oklikou. Cenu cesty představuje vzdálenost, kterou po dané cestě musí letadlo urazit od startovního stavu až do cílového stavu. Čím vyšší cenu tedy daná cesta má, tím méně je pro další zpracování zajímavá.

Vypočítat cenu cesty musí být možné pro všechny rozpracované cesty, které jsou během výpočtu vytvořeny. Takové cesty začínají ve startovním stavu, ale nevedou až do cílového, jen k němu směřují. Pro takovou cestu platí, že část vzdálenosti potřebné k dosažení cíle po této cestě již letadlo urazilo, zbytek ještě musí urazit. Cena, kterou již letadlo urazilo je označována jako cena od startu $cena_{start}$, cena, která zbývá do cíle je označována jako předpokládaná cena do cíle $cena_{cil}$. Celková cena cesty je pak součtem obou těchto cen $cena = cena_{start} + cena_{cil}$.

Vypočítat cenu od startu je výpočetně jednoduché. Je vypočítávána skutečná vzdálenost, kterou letadlo urazilo od startovního stavu. Na začátku výpočtu, pro triviální cestu obsahující pouze startovní stav je to $cena_{start} = 0$. V každém kroku výpočtu, kdy je do cesty přidán další stav je cena od startu navýšena o vzdálenost, kterou letadlo urazí do tohoto dalšího stavu (označena d) $cena_{start} = cena_{start} + d$. Vzdálenost do dalšího stavu je vypočítána jako euklidovská vzdálenost dalšího stavu (souřadnice (x_N, y_N, z_N)) od jemu bezprostředně předcházejícího stavu na cestě (souřadnice (x_P, y_P, z_P)):

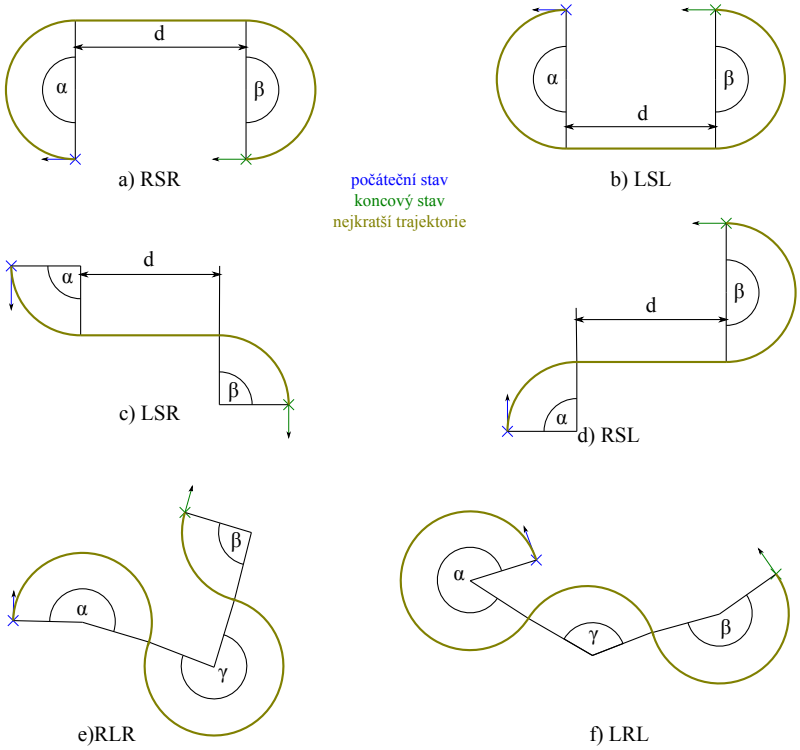
$$d = \sqrt{(x_N - x_P)^2 + (y_N - y_P)^2 + (z_N - z_P)^2}$$

Vypočítat předpokládanou cestu do cíle je výpočetně náročnější. Předpokládaná cena do cíle udává vzdálenost, kterou z daného stavu bude muset urazit letadlo do cílového stavu, pokud poletí po ideální dráze. Pro tento výpočet není možné počítat pouze s euklidovskou vzdáleností, protože takový výpočet nebere v úvah směr, který letadlo poletí. Pro výpočet předpokládané cesty do cíle $cena_{cil}$ se využívá Dubinsových křivek, jak předepisuje metoda [3].

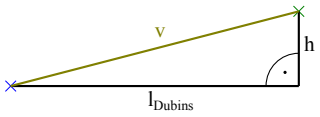
5.2.4 Dubinsovy křivky

Výpočet předpokládané ceny do cíle představuje výpočet teoretické vzdálenosti, kterou musí letadlo urazit z jednoho definovaného stavu do druhého. Pro propojení dvou stavů, určených pozicí a směrem, v rovině slouží Dubinsovy křivky [5]. Dubinsovy křivky představují nejkratší dráhu, kterou musí urazit neholonomní objekt z jednoho bodu do druhého, kde pro oba body je definován směr, kterým musí daný objekt bodem proletět. Vypočítávají se v rovině, tedy 2D prostoru. Existuje šest variant Dubinsových křivek, které ukazuje obrázek 5.4. Trajektorie je tvořena třemi segmenty, které jsou buď rovnými úseky (značené S) nebo úseky opisujícími pohyb po kružnici, a to buď zatáčením doprava (značené R) anebo zatáčením doleva (značené L). Podle toho se pak jednotlivé varianty Dubinsových křivek značí RSR, LSL, RSL, LSR, RLR nebo LRL. Poloměr kružnice, po které je kruhový pohyb opisován je roven poloměru zatáčení letadla r_{min}.

Protože Dubinsovy křivky se vypočítávají jen ve 2D prostoru, je jako délka Dubinsovy křivky určena vzdálenost v horizontální rovině. Označme tuto horizontální vzdálenost l_{Dubins}. Pro výpočet celkové vzdálenosti je potřeba znát také výškový rozdíl obou stavů h. Pokud horizontální vzdálenost l_{Dubins} není dostatečně dlouhá k překonání výškového rozdílu h daného maximálním povoleným úhlem α stoupání či klesání letadla: $l_{Dubins} < \frac{h}{\operatorname{tg} \alpha}$, je ke vzdálenosti přičten jeden nebo více obletů o 360° tak, aby byla horizontální vzdálenost



Obrázek 5.4: Šest variant Dubinsových křivek.



Obrázek 5.5: Rozdíl celkové a horizontální vzdálenosti.

dostatečná: $l_{Dubins} = l_{Dubins} + r_{min} \cdot 2 \cdot \pi$. Celkovou vzdálenost v pak vypočítáme pomocí Pythagorovy věty, jak ukazuje obrázek 5.5:

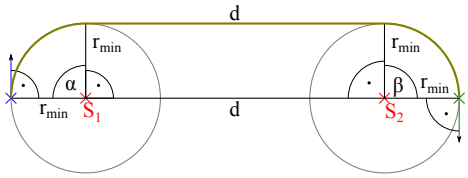
$$v = \sqrt{l_{Dubins}^2 + h^2}$$

Není jasné, která z šesti variant Dubinsových křivek bude pro dané dva stavy nejkratší, jednodušší než určovat, kterou variantu zvolit, je vypočítat všech šest variant, porovnat jejich délky a vybrat tu nejkratší. Dubinsovy křivky značené na obrázku 5.4 a), b), c) a d), tedy křivky RSR, LSL, RSL a LSR, obsahují rovný segment. Délka l těchto Dubinsových křivek je dána vztahem (úhly α a β musí být zadávány v radiánech):

$$l = r \cdot \alpha + r \cdot \beta + d \tag{5.1}$$

Pro zbývající varianty křivek RLR a LRL, které rovný segment neobsahují je jejich délka rovna součtu všech kruhových oblouků (úhly α, β i γ musí být zadávány v radiánech):

$$l = r \cdot \alpha + r \cdot \beta + r \cdot \gamma \tag{5.2}$$



Obrázek 5.6: Výpočet délky Dubinsovy křivky typu RSR (obdobně pro LSL).

Pro vypočítání délky Dubinsových křivek je tedy nezbytné určit, jak velký úhel musí pohybující se objekt opsat po každé z kružnic, případně jak dlouhý je rovný segment spojující dvě kružnice. Pro různé typy Dubinsových křivek probíhá výpočet odlišně.

5.2.5 Výpočet délky Dubinsových křivek

Pro výpočet délky daný vztahem 5.1 resp. 5.2 je nutné určit velikosti úhlů určujících, jak velký oblouk musí letadlo urazit po každé z kružnic a případně délku rovného segmentu. Výsledkem výpočtu je horizontální vzdálenost, kterou musí letadlo urazit mezi dvěma definovanými stavy. V některých případech nemusí cesta určitého typu existovat, v tom případě její výpočet selže a nejkratší varianta se vybírá z ostatních typů Dubinsových křivek. Vždy existuje nejméně jedna varianta.

Pro Dubinsovy křivky typu RSR, podle obrázku 5.6 probíhá výpočet následovně, pro křivky typu LSL je výpočet obdobný:

1. Výpočet středů kružnic S_1 a S_2. Středy kružnic jsou posunuty o vzdálenost r_{min}, představující minimální poloměr zatáčení letadla, vpravo od startovního, resp. cílového stavu.

2. Výpočet velikosti úhlu α jako úhel mezi spojnicí středů obou kružnic a směrem ze startovního bodu do středu S_1.

3. Výpočet úhlu β jako úhel mezi spojnicí středů obou kružnic a směrem z cílového bodu do středu S_2.

4. Výpočet vzdálenosti obou středů d.

5. $l_{Dubins} = d + r \cdot \alpha + r \cdot \beta$ (úhly jsou v radiánech).

Pro Dubinsovy křivky typu RSL (obdobně také pro LSR) probíhá výpočet podle obrázku 5.7 následovně:

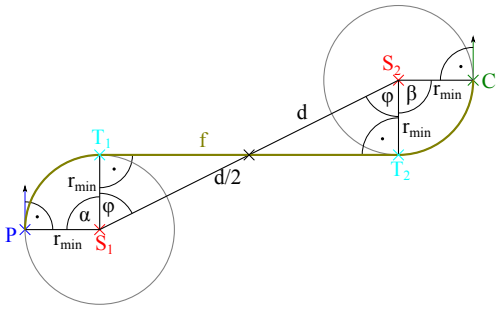
1. Výpočet středů kružnic S_1 a S_2. Středy kružnic jsou posunuty o vzdálenost r_{min}, představující minimální poloměr zatáčení letadla, vpravo od startovního, resp. vlevo od cílového stavu.

2. Výpočet vzdálenosti obou středů d a určení směru jejich spojnice (směr d).

3. Výpočet pomocného úhlu φ jako: $\varphi = \arccos \frac{2 \cdot r_{min}}{d}$.

4. Výpočet směru od S_1 k tečnému bodu T_1 a od S_2 k tečnému bodu T_2.

5. Výpočet pozice tečných bodů T_1 a T_2.

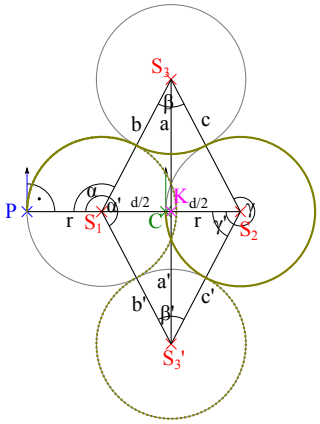


Obrázek 5.7: Výpočet délky Dubinsovy křivky typu RSL (obdobně pro LSR).

6. Výpočet vzdálenosti f mezi tečnými body T_1 a T_2.
7. Výpočet úhlu α jako úhel svíraný úsečkami $|PS_1|$ a $|S_1T_1|$.
8. Výpočet úhlu β jako úhel svíraný úsečkami $|CS_2|$ a $|S_2T_2|$.
9. $l_{Dubins} = f + r \cdot \alpha + r \cdot \beta$ (úhly jsou v radiánech).

Výpočet délky Dubinsových křivek typu RLR (obdobně LRL) zahrnuje dvě varianty, z nichž je vybrána ta kratší, výpočet probíhá podle obrázku 5.8:

1. Výpočet středů kružnic S_1 a S_2. Středy kružnic jsou posunuty o vzdálenost r_{min}, představující minimální poloměr zatáčení letadla, vpravo od startovního, resp. cílového stavu.
2. Výpočet vzdálenosti obou středů d a určení směru jejich spojnice (směr d).
3. V polovině úsečky d, bod označen jako K, sestrojena kolmice k d. Na této kolmici ve vzdálenosti a umístěn střed třetí kružnice S_3, dvě varianty označeny S_3 a S_3'. Vzdálenost a je určena jako $a = \sqrt{(2 \cdot r)^2 - (\frac{d}{2})^2}$.
4. Výpočet úhlu α jako úhel svíraný úsečkami $|PS_1|$ a $|S_1S_3|$, podobně úhel α' jako úhel svíraný úsečkami $|PS_1|$ a $|S_1S_3'|$.
5. Výpočet úhlu β jako úhel svíraný úsečkami $|S_1S_3|$ a $|S_2S_3|$, podobně úhel β' jako úhel svíraný úsečkami $|S_1S_3'|$ a $|S_2S_3'|$.
6. Výpočet úhlu γ jako úhel svíraný úsečkami $|S_2S_3|$ a $|CS_2|$, podobně úhel γ' jako úhel svíraný úsečkami $|S_2S_3'|$ a $|CS_2|$.
7. $l_{Dubins1} = r \cdot \alpha + r \cdot \beta + r \cdot \gamma$ (úhly jsou v radiánech).
8. $l_{Dubins2} = r \cdot \alpha' + r \cdot \beta' + r \cdot \gamma'$.
9. $l_{Dubins} = \min(l_{Dubins1}, l_{Dubins2})$.



Obrázek 5.8: Výpočet délky Dubinsovy křivky typu RLR (obdobně pro LRL).

5.2.6 Detekce kolizí

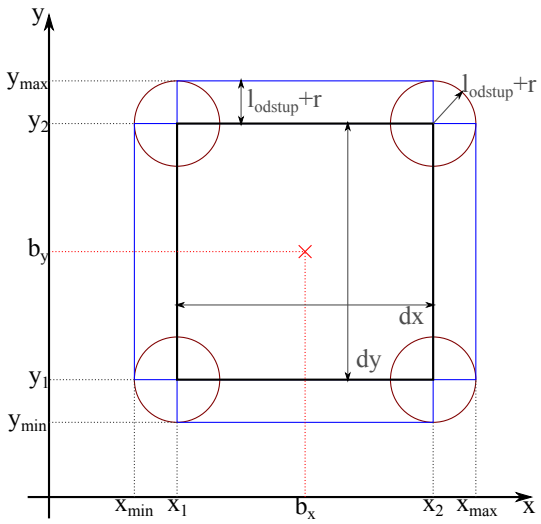
Důležitou podmínkou pro správné nalezení trajektorie napříč prostorem je také bezkoliznost. Globální stavy, které jsou kolizní musí být z dalšího výpočtu vyloučeny. Tím je zajištěno, že žádná z prohledávaných cest není kolizní a tedy i výsledná nalezená cesta je bezkolizní. Pro každý nový globální stav, který je přidáván do některé rozpracované globální cesty je nutné ověřit, zda nekoliduje s některou z překážek, detekci kolizí. Pro usnadnění výpočtu jsou stavy, které byly jednou zjištěny jako kolizní umístěny do seznamu CLOSED. Pokud bude takový stav prověřován v budoucnu, algoritmus na vlastní detekci kolize již nebude nutné provádět (viz bod 9b hledání globální cesty).

Protože letadlo je během výpočtu chápáno jako hmotný bod, musí tento bod reprezentující střed letadla minout překážku s dostatečným odstupem. Tento dostatečný odstup je daný šířkou letadla w a požadovanou tolerancí t. Tolerance určuje, o jakou vzdálenost musí křídlo letadla minout překážku a v aplikaci je stanovena na pevnou hodnotu. Odstup, s jakým musí střed letadla minout překážku je pak roven $l_{odstup} = \frac{w}{2} + t$. Aby bylo zajištěno, že střed letadla mine překážku s dostatečným odstupem, jsou rozměry všech překážek před zahájením detekce kolizí zvětšeny o hodnotu odstupu l_{odstup}.

Globální stav reprezentuje buňku diskrétní mřížky a směr, kterým jí letadlo prolétává. Směr letu není pro detekci kolizí podstatný, podstatná je pozice, tedy umístění buňky diskrétní mřížky v prostoru. Pro výpočet kolize je nutné zjistit, zda dva geometrické objekty mají neprázdný průnik. Pokud mají prázdný průnik, nekolidují. Pokud nějaký průnik mají, jsou v kolizi. Jako geometrický objekt je buňka diskrétní mřížky kvádrem s danými rozměry dx, dy a dz a středem b_x, b_y a b_z. Tento kvádr je navíc kolmý na všechny souřadnicové osy. Překážky, pro které je nutné vypočítávat kolize mají tvar buď koule nebo kvádru.

Méně náročný je výpočet detekce kolize buňky diskrétní mřížky s překážkou tvaru koule. Koule je invariantní vzhledem k rotaci, má ve všech směrech stejnou velikost, poloměr koule r. Pro zjištění, zda koliduje s buňkou diskrétní mřížky je potřeba ověřit, zda střed koule je dostatečně daleko od okraje kvádru představujícího buňku diskrétní mřížky. Výpočet detekce kolize s koulí znázorňuje obrázek 5.9 a probíhá následovně:

1. Pro každou osu je vypočítán interval, který v dané ose zaujímá buňka diskrétní mřížky a k němu je přičten odstup l_{odstup}. Takto je vytvořen kvádr (na obrázku znázorněn modře) pro každou osu:



Obrázek 5.9: Vymezení oblasti pro detekci kolize s koulí.

- Pro osu x je to kvádr zaujímající interval na ose x: $\langle x_{min}, x_{max} \rangle$ a na osách y a z intervaly $\langle y_1, y_2 \rangle$ a $\langle z_1, z_2 \rangle$:

  $x_{min} = b_x - \frac{dx}{2} - l_{odstup} - r$

  $x_{max} = b_x + \frac{dx}{2} + l_{odstup} + r$

  $y_1 = b_y - \frac{dy}{2}$

  $y_2 = b_y + \frac{dy}{2}$

  $z_1 = b_z - \frac{dz}{2}$

  $z_2 = b_z + \frac{dz}{2}$

  Pokud střed koule leží ve všech osách uvnitř těchto intervalů, je koule kolizní s buňkou diskrétní mřížky.
- Pro osy y a z probíhá výpočet obdobně.

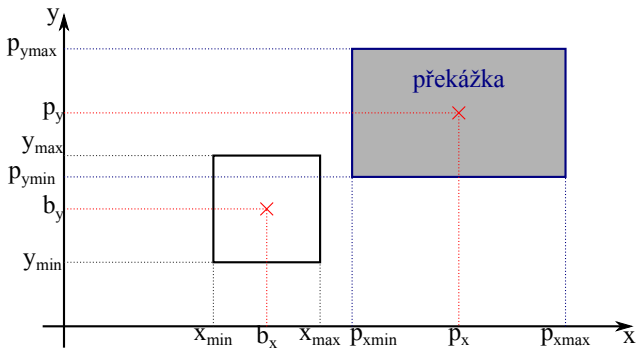
2. Poté je potřeba prověřit, zda se střed koule (překážky) nenachází příliš blízko u vrcholu kvádru buňky diskrétní mřížky.

   Pro každý vrchol kvádru buňky diskrétní mřížky je vypočítána vzdálenost od středu koule (překážky) k. Pokud $k < l_{odstup} + r$ pro některý z vrcholů, pak došlo ke kolizi.

3. Pokud nebyla v předchozích bodech zjištěna kolize, pak ke kolizi nedošlo.

Detekce kolize buňky diskrétní mřížky s kvádrem vypočítává kolizi s obalujícím kvádrem, který obaluje překážku. Obalující kvádr je kolmý na všechny souřadnicové osy, stejně tak kvádr reprezentující buňky globální mřížky. Překážka ve tvaru kvádru má střed v bodě (p_x, p_y, p_z) a rozměry dx_p, dy_p a dz_p. Pro detekci kolize, kterou znázorňuje obrázek 5.10, je potřeba ověřit:

1. Pro každou souřadnicovou osu je vypočítán interval, který na dané ose zaujímá buňka diskrétní mřížky:



Obrázek 5.10: Detekce kolize s kvádrem.

- pro osu x, interval $\langle x_{min}, x_{max} \rangle$:

  $x_{min} = b_x - \frac{dx}{2}$

  $x_{max} = b_x + \frac{dx}{2}$,
- obdobně pro ostatní osy y a z.

2. Pro každou souřadnicovou osu je také vypočítán interval, který na dané ose zaujímá překážka (obalující kvádr):

   - pro osu x, interval $\langle p_{xmin}, p_{xmax} \rangle$:

     $p_{xmin} = p_x - \frac{dx_p}{2}$

     $p_{xmax} = p_x + \frac{dx_p}{2}$,
   - obdobně pro osy y a z.

3. Pokud se intervaly zaujímané buňkou diskrétní mřížky (pro osu x interval $\langle x_{min}, x_{max} \rangle$) a intervaly zaujímané překážkou (pro osu x interval $\langle p_{xmin}, p_{xmax} \rangle$) překrývají ve všech třech osách, došlo ke kolizi.

   Na obrázku 5.10 je vidět, že intervaly $\langle y_{min}, y_{max} \rangle$ a $\langle p_{ymin}, p_{ymax} \rangle$ se překrývají. Protože se ovšem překrývají intervaly jen v jedné ose, ke kolizi nedošlo.

Aby bylo ověřeno, zda je buňka globální mřížky skutečně bezkolizní, je potřeba provést detekci kolize pro všechny překážky, které byly načteny ze vstupního souboru. Pokud je zjištěno, že došlo ke kolizi s některou z překážek, ostatní překážky se již neověřují, jinak je potřeba projít všechny a pro každou z nich detekci kolize vypočítat. Kolize jsou vypočítávány pro všechny překážky v seznamu překážek dodaného předchozí fází výpočtu, načítání a zpracování vstupních dat (viz kapitola 5.2), na rozdíl od přesného znění metody Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] nejsou vynechávány ani malé překážky.

Detekce kolizí s překážkou je náročný výpočet, zejména proto, že se během plánování provádí velmi často pro všechny procházené stavy a všechny překážky. Složitost tohoto výpočtu je proto pro celkovou složitou výpočtu velmi důležitá. Proto detekci kolizí usnadňuje použití seznamu CLOSED k zaznamenání již nalezených kolizních stavů.

5.3 Lokální plánování

Další fáze výpočtu a druhá fáze užité metody plánování trajektorie Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] popsané v kapitole 2.3 je lokální plánování. Navazuje na globální plánování, zpřesňuje nahrubo naznačenou trajektorii nalezenou během globálního plánování a zasazuje ji do spojitého prostoru.

Vstupem lokálního plánování je globální cesta, tedy seznam globálních stavů, kterými letadlo projde od startovního do cílového globálního stavu, seřazený podle toho, v jakém pořadí letadlo jednotlivé stavy proletí. Každý z těchto globálních stavů reprezentuje buňku diskrétní mřížky definovanou souřadnicemi jejího středu a jejími rozměry. Každá z těchto buněk tak zaujímá ve spojitém prostoru určité místo, ve kterém se bude pohybovat trajektorie nalezená v lokálním plánování. Dalším vstupem je přesná startovní a cílová pozice. Výstupem lokálního plánování je seznam lokálních stavů, tedy bodů ve spojitém prostoru, kterými letadlo proletí, seřazený od startovního bodu až do cílového bodu. Tento seznam lokálních stavů se označuje lokální cesta.

Lokální plánování vyhledává trajektorii v prostoru vymezeném buňkami diskrétní mřížky, ve kterých byla nalezena globální cesta v předchozí fázi výpočtu. Protože tento prostor byl již ověřen jako bezkolizní, není nutné, aby byla znovu prováděna detekce kolizí. Lokální plánování proto kolize s překážkami nijak nekontroluje.

5.3.1 Hledání lokální cesty

Během lokálního plánování jsou postupně vyhledávány úseky celkové trajektorie mezi jednotlivými buňkami diskrétní mřížky. Plánování tak probíhá od bodu k bodu. Protože rozměry buňky diskrétní mřížky dx a dy jsou stanoveny jako $\frac{2}{3} \cdot r_{min}$ (r_{min} je minimální poloměr zatáčení letadla), nedává prostor vymezený jedinou buňkou letadlu velké manévrovací možnosti. Pro hledání cesty, jako posloupnosti lokálních stavů, do další buňky diskrétní mřížky je možné využít zjednodušený algoritmus Greedy-search:

1. Inicializace lokální cesty, na začátku obsahuje pouze počáteční lokální stav.
2. Pro všechny buňky globální mřížky se opakuje tento postup:
   (a) Dílčím cílovým bodem je bod na konci aktuální prohledávané buňky globální mřížky, vzhledem ke směru, kterým má letadlo buňku proletět. Pro poslední buňku diskrétní mřížky v globální cestě, tedy cílový globální stav, je dílčím cílovým bodem skutečná cílová pozice.
   (b) Aktuálním stavem je poslední dosud nalezený lokální stav v lokální cestě.
   (c) Pro aktuální lokální stav je určen seznam všech možných následujících stavů, tj. seznam všech lokálních stavů, do kterých je možné se z aktuálního stavu dostat.
   (d) Všechny možné následující stavy jsou oceněny, je vypočítána jejich předpokládaná cena do dílčího cíle.
   (e) Je vybrán následující stav jako stav s nejnižší cenou ze všech možných následujících stavů.
   (f) Pokud je cena následujícího stavu nižší než předcházejícího, je stav přidán do lokální cesty a pokračuje se od bodu 2b, jinak se pokračuje dalším bodem.
   (g) Cena následujícího stavu je vyšší než cena aktuálního stavu, cesta se tedy začíná vzdalovat od cílového bodu. Tím je nalezen stav nejblíže k cílovému bodu.

Úsek trajektorie pro jednu buňku globální mřížky je nalezen celý, pokračuje se hledáním úseku lokální cesty pro další buňku diskrétní mřížky.

(h) Je zvolena další buňka diskrétní mřížky, pro kterou se vypočítá nový dílčí cílový bod, v bodě 2a.

3. Lokální cesta je ukončena v cílovém bodě.

V každém kroku výpočtu je k dosud nalezené cestě přidán jeden lokální stav, směřující k dílčímu cílovému bodu. Ze všech možných následujících stavů je vybrán jediný, který je k dílčímu cíli nejblíže. Ostatní možné stavy jsou zahozeny, v dalším výpočtu se s nimi už nepracuje. Vzhledem k tomu, že prostor buňky globální mřížky, ve kterém se lokální cesta vyhledává je bezkolizní, není nutné prohledávat více různých možných cest, jako provádí globální plánování. Je vyhledávána pouze jedna cesta, která směřuje co nejblíže k dílčímu cíli. Taková cesta je nejkratší možná a tedy optimální.

Vzhledem k omezeným manévrovacím možnostem letadla nemusí být vždy možné přesně dosáhnout cílového bodu. Proto se během výpočtu ověřuje cena následujícího stavu a pokud je vyšší než cena předchozího stavu, je hledání cesty do daného dílčího cílového bodu dokončeno. Podobně je také možné, že nalezená trajektorie nebude ukončena přesně v cílovém bodě, ale bude se nacházet uvnitř buňky diskrétní mřížky reprezentující cílový globální stav, co nejblíže skutečné cílové pozici, co umožní manévrovací schopnosti letadla.

5.3.2 Lokální stav

Lokální stav určuje pozici letadla ve skutečném prostoru a směr jeho letu. Nalezená trajektorie je vyjádřená posloupností lokálních stavů nazvanou lokální cesta. Na rozdíl od globálního stavu určujícího buňku diskrétní mřížky, která zaujímá v prostoru určitý objem, lokální stav je skutečně pouze jeden bod. Podobně rozdíl mezi globální a lokální cestou je ten, že v globální cestě na sebe jednotlivé buňky přesně navazují, zatímco v lokální cestě jsou mezi body, lokálními stavy, mezery, které vyplňuje křivka spojující sousední stavy.

Každý lokální stav je definován těmito údaji:

- souřadnice bodu v prostoru (x, y, z),
- směr, kterým letadlo daný bod prolétává, vyjádřený úhlem α.

Pro každý lokální stav je možné vypočítat možné další lokální stavy, do kterých lze z daného lokálního stavu přejít. Možné následující stavy definuje směr aktuálního lokálního stavu. Pro každý elementární pohyb, který letadlo vykoná mezi jednotlivými lokálními stavy je určen jeden následující stav, do kterého se letadlo po provedení elementárního pohybu dostane. Tyto elementární pohyby jsou nazvané pohybová primitiva. Letadlo má k dispozici celkem devět možností pohybu, tedy devět pohybových primitiv. Letadlo může:

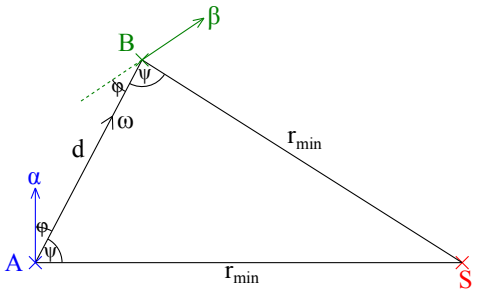
→ letět rovně a udržovat výšku,

→ zatáčet doprava a udržovat výšku,

→ zatáčet doleva a udržovat výšku,

↓ letět rovně a klesat,

↓ zatáčet doprava a klesat,



Obrázek 5.11: Výpočet následujícího lokálního stavu.

↓ zatáčet doleva a klesat,

↑ letět rovně a stoupat,

↑ zatáčet doprava a stoupat,

↑ zatáčet doleva a stoupat.

Pro každé z těchto primitiv je definován výpočet, do kterého stavu se letadlo po provedení odpovídajícího elementárního pohybu dostane.

Dráha jednoho elementárního pohybu je stanovena na takovou délku, aby letadlo při letu z jedné buňky diskrétní mřížky do další udělalo přibližně 15 elementárních pohybů. Vzdálenost, o kterou letadlo změní svoji pozici v prostoru po provedení jednoho elementárního pohybu je $d = \frac{dx}{15}$ (dx značí čtvercový rozměr buňky diskrétní mřížky). Výpočet pozice letadla v následujícím stavu znázorňuje obrázek 5.11.

Podle obrázku 5.11 úhel α definuje, jakým směrem se letadlo pohybuje v horizontálním prostoru v aktuálním lokálním stavu A, úhel β určuje směr, kterým letadlo poletí v následujícím lokálním stavu B. Pokud letadlo letí rovně, pak $\beta = \alpha$. Pokud letadlo zatáčí, pak se směr jeho pohybu změní o $2 \cdot \varphi$. Pak $\beta = \alpha - 2 \cdot \varphi$ při zatáčení doprava anebo $\beta = \alpha + 2 \cdot \varphi$ při zatáčení doleva. Směr, o který se letadlo posune v prostoru, je na obrázku 5.11 naznačen úhlem ω. Pokud letadlo letí rovně, platí že $\omega = \alpha$, jinak $\omega = \alpha - \varphi$ pokud letadlo zatáčí doprava anebo $\omega = \alpha + \varphi$ pokud letadlo zatáčí doleva. Souřadnice bodu B, do kterého letadlo doletí, jsou značeny (x_B, y_B, z_B) a souřadnice bodu A, jsou (x_A, y_A, z_A). Pro x a y souřadnice těchto bodů platí:

$$\begin{aligned} x_B &= x_A + d \cdot \cos\omega \\ y_B &= y_A + d \cdot \sin\omega \end{aligned}$$

Velikost pomocného úhlu φ lze určit ze vztahu:

$$\varphi = \frac{\pi}{2} - \arccos\frac{d}{2 \cdot r_{min}}$$

5.3.3 Výpočet ceny lokálního stavu

Pro rozlišení, který lokální stav vede nejblíže k cíli jsou jednotlivé stavy ohodnocovány cenou představující předpokládanou vzdálenost do cíle, podobně jako je tomu u globálních stavů. Na rozdíl od ohodnocování globálních stavů, popsaném v kapitole 5.2.3, kde je využito výpočtu pomocí Dubinsových křivek [5], pro výpočet předpokládané vzdálenosti do cíle pro lokální stavy postačí euklidovská vzdálenost.

V rámci jediné buňky diskrétní mřížky, ve které se vyhledává lokální cesta, nemůže dojít ke kolizi, protože tento prostor byl již prokázán jako bezkolizní ve fázi globálního plánování. Lokální cesta propojuje body dané globálním plánováním, z každého bodu se tedy vyhledává nejrychlejší cesta do následujícího. Pro posouzení, která cesta do dalšího bodu vede nejrychleji, tedy je nejkratší, je euklidovská vzdálenost vhodným měřítkem.

Pro výpočet ceny lokálního stavu je tedy potřeba souřadnice tohoto lokálního stavu (x_L, y_L, z_L) a souřadnice dílčího cílového bodu, ke kterému aktuální úsek lokální cesty směřuje (x_C, y_C, z_C). Cena lokálního stavu c je pak dána známým vztahem:

$$c = \sqrt{(x_L - x_C)^2 + (y_L - y_C)^2 + (z_L - z_C)^2}$$

5.4 Vytvoření výstupu

Výstupem aplikace je trajektorie propojující v zadaném prostoru startovní a cílovou konfiguraci. Tato trajektorie je součástí prostoru, ve kterém je vyhledávána. Aplikace po nalezení trajektorie vytvoří soubor obsahující informace o prostoru a do tohoto souboru také zapíše nalezenou trajektorii. Výstupem aplikace je tedy jeden soubor ve formátu XML X3D. Dalším volitelným výstupem aplikace je logovací soubor ve formátu TXT nebo LOG, který obsahuje informace o vyhledávání trajektorie.

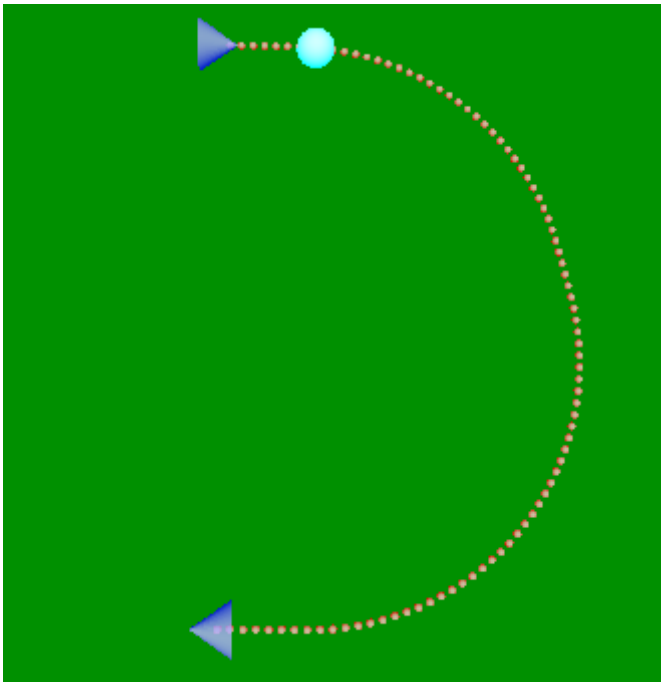
Hlavním výstupem aplikace je tedy XML X3D soubor, který obsahuje informace o prostoru, statické znázornění nalezené trajektorie a animaci trajektorie. Uložení tohoto souboru je definováno jedním ze vstupních argumentů. Výstupní soubor je vytvořen jako kopie vstupního souboru, obsahuje tedy i takové informace o prostoru, které aplikace nezpracovává. Do výstupního souboru je zapsán seznam tagů načtených ze vstupního souboru a zpracovaných knihovnou DOM [7]. Pomocí knihovny DOM je také vytvořena struktura tagů popisujících výstupní data aplikace.

Do výstupního souboru je přidán blok tagů popisujících výstupní data aplikace. Tento blok tagů je přidán na začátek tagu Scene (viz obrázek 4.1). Nově přidaný obsah zahrnuje seznam tagů popisujících nalezenou trajektorii a seznam tagů popisujících objekt pohybující se po této trajektorii a popis animace tohoto objektu.

5.4.1 Úprava dat pro zápis

Trajektorie, kterou aplikace nalezne je definována lokální cestou, tedy seznamem bodů, které objekt postupně projde od startovního až po cílový. Tyto jednotlivé body jsou ve výstupním souboru znázorněny koulemi. Protože těchto bodů může být velmi mnoho, nebylo by vhodné ukládat do výstupního souboru příliš mnoho objektů. Maximální počet takto vytvořených koulí znázorňujících nalezenou trajektorii je proto omezen na 100. Znamená to, že po vypočítání trajektorie, je počet bodů popisujících tuto trajektorii zmenšen, pokud je to potřeba tak, aby těchto bodů nebylo více než 100, včetně startovního a cílového bodu.

Startovní a cílový bod jsou znázorněny geometrickým objektem jiné barvy a tvaru naznačujícího směr letu v daném bodě. Příklad vzhledu výstupu aplikace ukazuje obrázek



Obrázek 5.12: Příklad výstupu aplikace.

5.12. Na obrázku je vidět znázornění startovního a cílového stavu modrými kužely, jejichž orientace naznačuje definovaný směr pohybu v daných bodech. Trajektorie je znázorněná koulemi umístěnými v rovnoměrné vzdálenosti od sebe. Samotný pohybující se objekt je naznačen bílou koulí, která se pohybuje po trajektorii směrem od startu k cíli.

Pro animace v X3D souboru je potřeba zvolit čas, po který bude animace probíhat. Tento čas je aplikací pevně stanoven na $10s$, bez ohledu na délku trajektorie. Rychlost objektu je po celé dráze konstantní. Okamžitě po dokončení cesty se objekt objeví znovu na startu, takto se po trajektorii pohybuje periodicky v intervalech $10s$.

Kapitola 6

Experimentální výsledky a porovnání složitosti

Pro implementaci vyhledávání trajektorie byla v aplikaci využita metoda Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] popsaná v kapitole 2.3. V této kapitole bude porovnána složitost této zvolené metody s ostatními metodami popsanými v kapitole 2. Bude také podrobněji popsána složitost implementovaného algoritmu aplikace zvolenou metodou. Složitost celého výpočtu aplikace je daná především výpočtem trajektorie, probíhajícím ve dvou fázích.

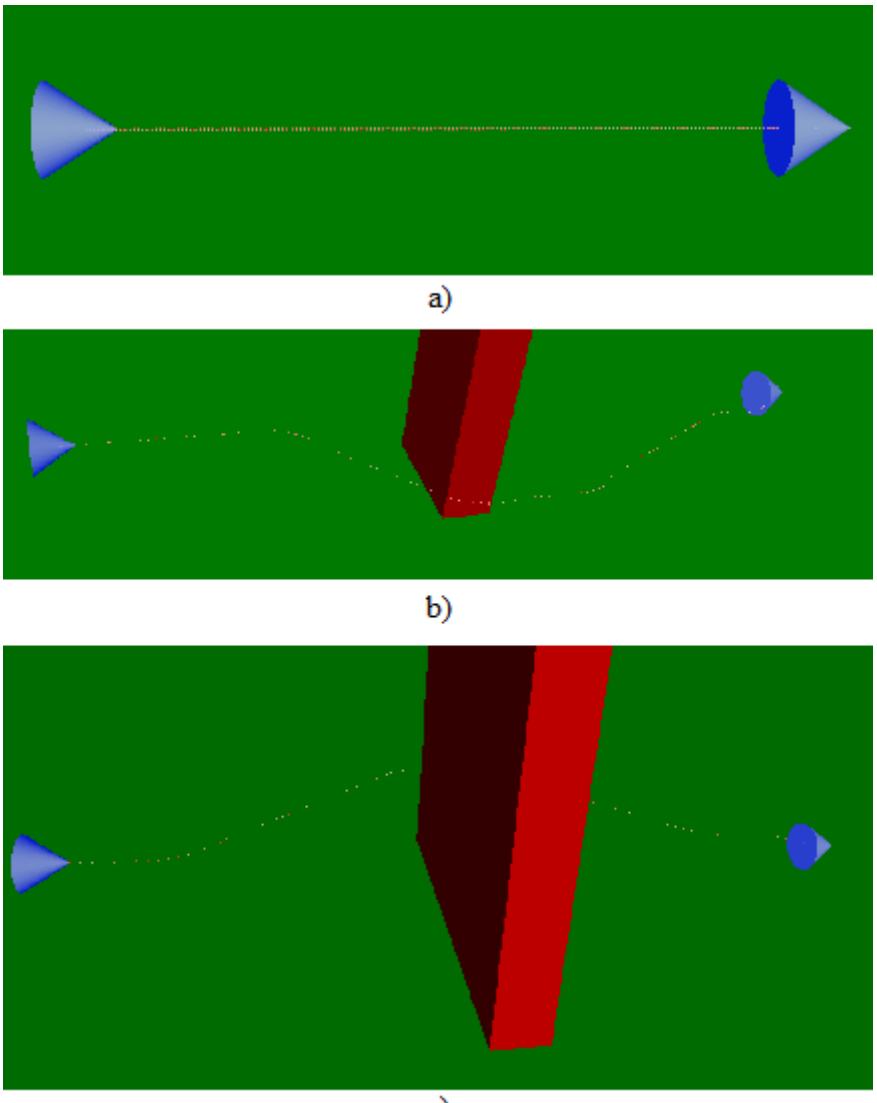
6.1 Složitost aplikace

Hlavní je složitost první fáze použité metody, globální plánování. Složitost je daná především použitou metodou prohledávání stavového prostoru, která určuje, kolik stavů bude nutné prohledat, než bude nalezena posloupnost vedoucí k cíli. V aplikaci je stavový prostor prohledáván metodou A^*. Pro tuto metodu je zásadní použitá heuristika pro oceňování stavů. V kapitole 5.2.3 je popsán způsob výpočtu heuristiky. Zejména výpočet předpokládané ceny do cíle, založený na Dubinsových křivkách je značně složitější než prostý výpočet euklidovské vzdálenosti. Pro správné ocenění stavů je ale nezbytný. Správné ocenění stavů pak umožňuje prohledávat stavy více se blížící k cíli.

V případě, že na cestě k cíli nestojí žádná překážka, jak ukazuje obrázek 6.1a, je prohledáno jen minimální množství stavů, trajektorie směřuje přímo k cíli. Pokud je v cestě menší překážka, jak je znázorněno na obrázku 6.1b, je prohledáno více stavů, protože ideální cesta je zahrazena a výsledná trajektorie musí vést delší cestou. Na obrázku 6.1c je znázorněna větší překážka, která si vynucuje větší okliku oproti ideální přímé cestě a taky větší složitost výpočtu. Srovnání počtu prohledaných stavů a doby trvání výpočtu ukazuje tabulka 6.1. Výpočet, jehož doba trvání je v tabulce 6.1 vyznačena, byl prováděn na PC s 32-bitovým procesorem AMD Turion II Dual-Core Mobile M540 2,40 GHz s operační pamětí velikosti 3 GB.

Počet prohledaných stavů je určen metodou prohledávání stavového prostoru A^*. Pro určení složitosti je také potřeba určit složitost prohledání každého stavu. Prohledávání stavového prostoru je také uvedeno v kapitole 5.2. Prohledání každého stavu tak zahrnuje:

- Prohledání seznamů CLOSED a SEARCHED, pokud se stav vyskytuje v jednom z těchto seznamů, nebude prohledáván.



Obrázek 6.1: Příklad výstupu aplikace.

Obrázek	Počet prohledaných stavů	Doba trvání výpočtu
6.1a	24	3s
6.1b	836	7s
6.1c	3030	4min13s

Tabulka 6.1: Porovnání složitosti výpočtu s různými překážkami.

Velikost seznamů CLOSED a SEARCHED se během výpočtu zvětšuje, jejich procházení se tedy během výpočtu stává stále složitější.

- Vyhledání nejlevnější cesty ze seznamu OPEN.

  Velikost seznamu OPEN se během výpočtu typicky zvětšuje, pro vyhledání nejlevnější cesty je třeba průchod celým tímto seznamem.

- Detekce kolize pro prohledávaný stav.

  Je nutné ověřit kolize pro všechny překážky v prostoru. Protože se tento výpočet provádí pro každý prohledávaný stav, je jeho složitost velmi důležitá.

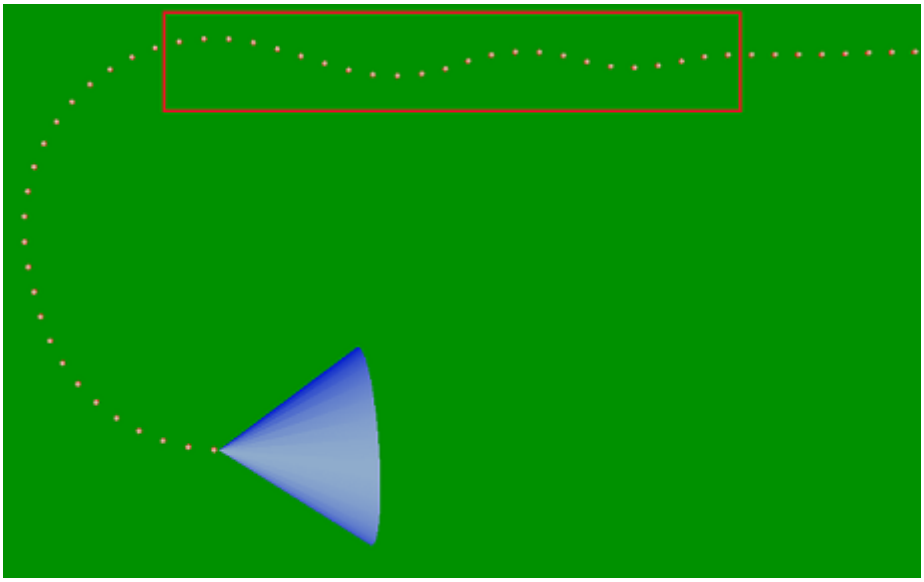
Složitost druhé fáze metody vyhledávání trajektorie, lokálního plánování, je značně nižší. Algoritmus lokálního plánování je popsán v kapitole 5.3. V lokálním plánování není nutné provádět detekci kolizí a budovat celý stavový strom. To výpočet lokální cesty zjednodušuje oproti hledání globální cesty. Pro každý prohledávaný lokální stav je vybrán jediný následující lokální stav, který je nejblíže k dílčímu cíli a ten je také zvolen pro další prohledávání. Celkový počet prohledaných lokálních stavů je tedy minimální, prohledány jsou jen stavy, které se v nalezené trajektorii skutečně vyskytují.

6.1.1 Srovnání složitosti s ostatními metodami

Plánovací metoda Sampling-based planning under differential constraints [4], jejíž princip je popsán v kapitole 2.1, pracuje také na principu vytváření a prohledávání stavového prostoru. Tato metoda však nepoužívá diskrétní mřížku, jednotlivé stavy jsou určovány ve spojitém prostoru, pro přechody mezi stavy jsou vypočítána pohybová primitiva. Pro tuto metodu není stanoven postup prohledávání stavového prostoru, nicméně metoda A^*, využitá v implementované metodě prohledávání, je zřejmě nejvhodnější i pro tuto prohledávací metodu. Při použití vhodné metody prohledávání stavového prostoru a vhodné heuristiky pro oceňování prohledávaných stavů může být složitost metody Sampling-based planning podobná implementované metodě. Zásadní pro složitost obou metod je však algoritmus pro detekci kolizí, který v případě metody Sampling-based planning musí prověřovat kolizi s celým úsekem cesty definovaným pohybovým primitivem. Takový výpočet bude zřejmě složitější než detekce kolize dvou geometrických objektů.

Metoda A Real-time 3D motion planning and simulation scheme for nonholomic systems [8] je popsaná v kapitole 2.2. Tato metoda nevytváří, na rozdíl od implementované plánovací metody, stavový prostor. Trajektorie je vyhledávána na základě vytváření Dynamicky alokovaných bodů (DAB), které jsou určovány podle překážek. Metoda samotná pracuje v reálném čase, trajektorii vyhledává vždy pro omezený prostor na dohled letadla. Časová složitost této metody je nižší, než je složitost implementované plánovací metody. Cenou za nižší složitost je ale neoptimální trajektorie, kterou tato metoda nalezne.

Metoda Time-optimal paths for a Dubins airplane [2] popsaná v kapitole 2.4, provádí hlavní plánování trajektorie v horizontální rovině. Teprve poté je trajektorie upravena do trojrozměrného prostoru. Plánování trajektorie v rovině má nižší složitost, započítání výšky do takto nalezené trajektorie je výpočetně jednoduchou operací. Složitost metody Time-optimal paths for a Dubins airplane je zřejmě nižší než složitost implementované metody.



Obrázek 6.2: Příklad výstupu aplikace.

6.2 Vlastnosti aplikace

Aplikace vyhledává trajektorii ve spojitém 3D prostoru s překážkami. Tento prostor je popsán ve formátu X3D se syntaxí XML. Po úspěšném nalezení trajektorie je vytvořen výstup rovněž ve formátu XML X3D, který obsahuje nalezenou trajektorii a animaci pohybu po této trajektorii. Vzhledem k použité metodě Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] je takto nalezená trajektorie vždy optimální a bezkolizní.

Hlavní výpočet hledání trajektorie probíhá v diskrétní mřížce, která aproximuje spojitý prostor. Optimálnost nalezené trajektorie je zaručena v rámci této diskrétní mřížky, skutečná spojitá trajektorie se může od optimální mírně odlišovat. Také detekce kolizí je prováděna pro buňky diskrétní mřížky, které mají vzhledem k rozměrům letadla poměrně velkou velikost. Proto může být zcela vyloučena z výpočtu cesta, která je vyhodnocena jako kolizní, přestože skutečná velikost mezery mezi překážkami průlet letadla umožňuje.

Po vypočítání trajektorie v diskrétní mřížce je vypočítána také spojitá trajektorie ve spojitém prostoru, tedy lokální plánování. Pro lokální plánování jsou stanoveny body jako dílčí cílové stavy, kterými letadlo postupně proletí. Lokální plánování vyhledává jednotlivé úseky spojité trajektorie tak, že se snaží nelézt cestu co nejblíže k těmto dílčím cílovým bodům. Ne vždy je ale nalezena cesta, která se do tohoto dílčího cílového bodu trefí přesně. To pak způsobí odchylky ve vyhledávání dalšího úseku cesty. Na výsledné trajektorii se tak mohou objevit zákmity. Příklad těchto zákmitů ukazuje příklad výstupu aplikace na obrázku 6.2.

Aplikace vždy nalezne řešení v konečném čase, pokud řešení existuje. Doba nalezení řešení se může dramaticky lišit vzhledem k překážkám, vyskytujícím se na předpokládané ideální cestě do cíle. Dobu trvání výpočtu pro tři různě velké překážky udává tabulka 6.1. Doba výpočtu závisí také na vzdálenosti startovního a cílového stavu a na poloměru zatáčení letadla, podle kterého se stanovuje velikost buňky diskrétní mřížky, i když vliv těchto faktorů není tak dramatický jako vliv překážek.

Pokud řešení neexistuje, může aplikace běžet do nekonečna, prohledávající nekonečně velké množství stavů v prostoru, jehož velikost není omezena. V některých případech může

aplikace odhalit, že řešení neexistuje a ukončí se neúspěšně, v jiných případech poběží do nekonečna. Neexistenci řešení aplikace odhalí v následujících případech:

- Buňka diskrétní mřížky, ve které se nachází startovní nebo cílová pozice, je určena jako kolizní.

  Startovní stav je prohledáván jako první a proto je výpočet ukončen okamžitě, pokud je startovní stav kolizní. Pro cílový stav je kolize explicitně zjišťována.

- Překážka nebo překážky jsou těsně před startovním stavem a neumožňují letadlu se jim vyhnout.

  Není zjištěna žádná cesta, kterou by se letadlo mohlo vydat a není tedy co prohledávat.

- Překážky tvoří okolo startovního stavu uzavřený obal, který vymezuje prostor, do kterého se letadlo může dostat a cílový stav uvnitř není nebo je nedosažitelný.

  Pokud je prostor všech stavů omezen, jsou v konečném čase prohledány všechny stavy. Seznam možných cest do cíle OPEN se vyprázdní, tedy jsou vyčerpány všechny možnosti vedoucí k cíli.

V jiných případech je potřeba prohledat nekonečně mnoho stavů, aby mohlo být zjištěno, že žádný nevede do cíle. Zejména v případě, kdy se překážka nachází těsně před cílovým stavem a neumožňuje tak letadlu do cílového stavu správným směrem nalétnout, ale samotný cílový stav je bezkolizní, není řešení nalezeno a výpočet poběží do nekonečna.

6.2.1 Možná rozšíření

Vzhledem k tomu, že aplikace implementuje definovanou metodu prohledávání prostoru a plánování trajektorie, možná rozšíření aplikace spadají spíše do oblasti poskytovaného výstupu. Aplikace poskytuje výstup ve formátu XML X3D, který umožňuje vizualizaci výstupu uživatelem. Jedním z dalších poskytovaných výstupů by mohl být textový či binární výstup popisující pohyby letadla z pohledu letadla, tedy seznam instrukcí pro letadlo, jak zadaný prostor proletět.

Kapitola 7

Závěr

V této práci jsem prozkoumal a popsal principy několika metod pro plánování trajektorie letadla v prostoru s překážkami. Prostudoval jsem a popsal jsem principy celkem čtyř zvolených metod plánování, Sampling-based planning under differential constraints [4] (kapitola 2.1), A Real-time 3D motion planning and simulation scheme for nonholomic systems [8] (kapitola 2.2), Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] (kapitola 2.3) a Time-optimal paths for a Dubins airplane [2] (kapitola 2.4). Vlastnosti těchto metod jsem vzájemně porovnal, zejména vhodnost použití dané metody pro plánování trajektorie dle zadání.

Ze čtyř popsaných metod jsem vybral dvě metody vhodné pro implementaci (viz kapitola 3.2). Z těchto dvou vhodných metod jsem vybral metodu Efficient Two-phase 3D motion planning for small fixed-wing UAVs [3] (kapitola 2.3) jako nejvhodnější metodu pro plánování trajektorie letadla v prostoru se statickými překážkami a odůvodnil výběr právě této metody.

Popsal jsem návrh aplikace provádějící plánování trajektorie se zaměřením na vstupy a výstupy aplikace a formát vstupních a výstupních dat. Jako formát popisující prostor, ve kterém se trajektorie vyhledává jsem se rozhodl využít otevřený grafický formát X3D [9], pro jehož prohlížení uživatelem lze zvolit například nástroj Instant Player z frameworku Instant Reality [6]. Navrhl jsem strukturu aplikace a způsob implementace zvolené plánovací metody.

Vytvořil jsem aplikaci implementující zvolenou metodu. Pro implementaci jsem zvolil jazyk Java (verze interpretu 1.7.0_09). Implementaci aplikace, rozdělenou do čtyř hlavních fází, jsem popsal v kapitole 5. Popsal jsem použité algoritmy, zejména algoritmus použitý k prohledávání stavového prostoru při globálním plánování, založený na metodě A^*, algoritmus pro detekci kolizí s překážkou (viz kapitola 5.2.6) a algoritmus pro výpočet délky Dubinsových křivek (v kapitole 5.2.4).

Na závěr práce jsem shrnul vlastnosti aplikace, její omezení a případy, kdy nalezne trajektorii, kdy alespoň zjistí, že trajektorie neexistuje a kdy poběží do nekonečna, hledající neexistující trajektorii. Zaměřil jsem se na časovou složitost aplikace, uvedl příklady doby výpočtu na testovacím PC. Porovnal jsem složitost implementované metody s ostatními popsanými metodami. Uvedl jsem možná rozšíření aplikace.

Literatura

[1] Buriánek, I. J.; Kavan, L.: AABB. *Statické detekce kolizí*, 2008: str. 12.

[2] Chitsaz, H.; LaValle, S. M.: Time-optimal Paths for a Dubins Airplane. *46th IEEE Conference on Decision and Control*, 2007.

[3] Hwangbo, M.; Kuffner, J.; Kanade, T.: Efficient Two-Phase 3D Motion Planning for Small Fixed-wing UAVs. *The Robotics Institute Carnegie Mellon University*, 2006.

[4] LaValle, S. M.: Sampling-based Planning Under Differential Constraints. *Planning Algorithms*, ročník 14, 2006: s. 787 – 860.

[5] Shkel, A. M.; Lumelsky, V.: Classification of the Dubins set. *Robotics and Autonomous Systems*, ročník 34, č. 4, 2001: s. 179 – 202, ISSN 0921-8890.

[6] The Fraunhofer Institute for Computer Graphics Research IGD: InstantReality. `http://www.instantreality.org/` [citováno 6.května 2013], 2012.

[7] W3C DOM Interest Group: Document Object Model. 2005, `http://www.w3.org/DOM` [citováno 8.května 2013].

[8] Wan, T. R.; Tang, W.; Chen, H.: A real-time 3D motion planning and simulation scheme for nonholonomic systems. *Simulation Modelling Practice and Theory*, ročník 19, č. 1, 2011: s. 423 – 439, ISSN 1569-190X.

[9] Web3D Consortium: What is X3D? 2013, `http://www.web3d.org/realtime-3d/x3d/what-x3d` [citováno 14.května 2013].

[10] WWW stránky: Matice rotace. 2012, `http://cs.wikipedia.org/wiki/Oto%C4%8Den%C3%AD_(geometrie)` [citováno 8.května 2013].

Příloha A

Obsah CD

Na CD přiloženém k práci jsou uvedeny zdrojové kódy aplikace a příklady jejího použití. Obsah CD je rozdělen do následující adresářové struktury:

- Instant Reality – obsahuje instalační balíčky pro instalaci frameworku Instant Reality, jehož součástí je program Instant Player, vhodný pro prohlížení souborů ve formátu X3D.
- NB_projekt – obsahuje celý projekt aplikace ve vývojovém prostředí NetBeans (verze 7.2.1).
- Priklady – příklady prostorů a konfigurací, pro které se vyhledává trajektorie s logovacími soubory udávajícími dobu výpočtu na testovacím PC.
- Program – obsahuje spustitelný archiv JAR s příkladem vstupních souborů (verze Java interpretu 1.7.0_09).
- Zdrojove kody – všechny zdrojové kódy aplikace v jazyce Java. Tytéž zdrojové kódy jsou obsaženy také v adresáři NB_projekt.