ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Bakalářská práce

Adam Třešňák

Optimalizace tvaru roje bezpilotních helikoptér

Katedra kybernetiky

Vedoucí práce: **Dr. Martin Saska**

Praha 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Adam Třešňák

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Optimalizace tvaru roje bezpilotních helikoptér

Pokyny pro vypracování:

Cílem práce je navržení, implementace a verifikace algoritmu optimalizujícího tvar roje bezpilotních helikoptér pro úlohy typu robotického dohledu. Vstupem algoritmu bude mapa oblasti s vyznačenými regiony, kde mají helikoptéry patrolovat. K jednotlivým územím bude přiřazena prioritní funkce vyznačující, jak důležité je danou oblast hlídat. Výstupem algoritmu bude výsledná pozice helikoptér a trajektorie popisující pohyb helikoptér ze základny do těchto nalezených lokací. Jak výsledná pozice, tak nalezené trajektorie musí splňovat omezení daná pravidly robotického roje. Konkrétně trajektorie musí respektovat omezení na relativní lokalizaci entit roje, jejich bezkoliznost a kinematická omezení pohybu helikoptér.

1. Navrhnout a implementovat vhodnou ohodnocující funkci a reprezentaci částice v algoritmu Particle Swarm Optimization (PSO) [3].
2. Integrovat model helikoptéry a kinematická omezení jejího pohybu [1].
3. Navrhnout a integrovat mechanizmus zajišťující splnění podmínek daných relativní vizuální lokalizací členů roje [2].
4. Experimentálně a statisticky ověřit vliv lokalizační podmínky na chování roje.
5. Ověřit funkčnost metody s využitím dostupné robotické platformy. V průběhu řešení projektu vedoucí práce rozhodne, zda navržená metoda bude otestována krátkým experimentem s 2-3 helikoptérami, na lanovém multi-robotu, případně na systému SyRoTek [4].

Seznam odborné literatury:

[1] Taeyoung Lee; Leoky, M.; McClamroch, N.H.: Geometric tracking control of a quadrotor UAV on SE(3). Decision and Control (CDC), 2010 49th IEEE Conference on , vol., no., pp.5420-5425, 15-17 Dec. 2010
[2] Saska, M. - Krajník, T. - Přeučil, L.: Cooperative Micro UAV-UGV Autonomous Indoor Surveillance. In International Multi-Conference on Systems, Signals and Devices. Piscataway: IEEE, 2012, p. 36. ISBN 978-3-9814766-1-3.
[3] Kennedy, J. and Eberhart, R. C.: Particle swarm optimization. In Proceedings International Conference on Neural Networks IEEE, volume 4, pp. 1942–1948, 1995.
[4] Kulich, M. - Košnar, K. - Chudoba, J. - Faigl, J. - Přeučil, L.: On a Mobile Robotics E-learning System. In Proceedings of the Twentieth European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetics Studies, 2010, p. 597-602. ISBN 978-3-85206-178-8.

Vedoucí bakalářské práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce zimního semestru 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2013

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 3. 1. 2014 .....

Poděkování

Děkuji panu Dr. Martinu Saskovi za ochotné a odborné vedení při vytváření této práce. Zvlášť mu děkuji za jeho cenné rady, poskytnuté studijní materiály a především za konzultace, které pomohly vyřešit řadu problémů, které s sebou práce přinesla. Chtěl bych také poděkovat své rodině a přátelům za pomoc a podporu při studiu.

Abstrakt

Cílem práce je navržení, implementace a verifikace algoritmu optimalizující tvar roje bezpilotních helikoptér (UAV – unmanned aerial vehicle) pro úlohy typu robotického dohledu. Za vstupní parametr algoritmu je považována mapa oblasti s vyznačenými regiony, které mají helikoptéry monitorovat. K jednotlivým územím je přiřazena prioritní funkce vyznačující, jak důležité je danou oblast hlídat. Výstupem algoritmu bude výsledná pozice helikoptér a trajektorie popisující pohyb helikoptér z výchozího umístění do těchto nalezených lokací. Popisovaný postup využívá modifikovaného algoritmu Particle Swarm Optimization (PSO). Jak výsledné pozice, tak nalezené trajektorie musí splňovat omezení daná pravidly robotického roje. Konkrétně musí respektovat omezení relativní lokalizace entit roje a dodržovat mezi entitami bezpečnou vzdálenost, aby bylo zabráněno kolizím. Algoritmus musí rovněž uvažovat kinematická omezení pohybu helikoptér.

Abstract

In this thesis, the design, implementation and verification of the shape optimizing algorithm for swarms of unmanned aerial vehicles (UAVs) is presented. The map with distinguished areas, which are to be monitored, is considered as an input parameter. All of these areas are rated according to their priority. Resulting positions of all UAVs in the swarm and the trajectories leading to this positions are considered as the algorithm output. The presented approach uses modified Particle Swarm Optimization (PSO) algorithm. These found trajectories as well as the resulting positions have to fulfil the constrictions of a robotic swarm. The restriction of motion of real robots model, relative localization among the members of swarm and the collision-free trajectories have to be considered.

Obsah

Seznam obrázků

Seznam pseudokódů

1 Úvod

Roboty se v posledních desetiletích staly nedílnou součástí každodenního života a našly své uplatnění ve většině oborů lidské činnosti. Od domácích spotřebičů až k nasazení v různých průmyslových aplikacích na výrobních linkách, použití v medicíně nebo v kosmonautice. Ve všech oborech můžeme nalézt nespočet různých typů robotů, které se liší velikostí, mírou autonomie, mobilitou, inteligencí, způsobem programování atd.

Pokud jde o rozdělení robotů podle takovýchto hledisek, lze roboty mimo jiné dělit také podle schopnosti samostatně se pohybovat na *stacionární* (stroj bez možnosti volného pohybu s omezeným pracovním prostorem, který je fixován na jedno konkrétní místo) a *mobilní* (jsou schopny samostatného pohybu mezi různými místy pracovního prostředí). Oborem zabývajícím se problematikou samostatně se pohybujících strojů je mobilní robotika.

V mobilní robotice se lze setkat s roboty, které jsou řízené člověkem i s roboty autonomními, které zadaný úkol vykonávají zcela samy. Některé úlohy mobilní robotiky si žádají nasazení a spolupráci většího množství robotů, protože je mimo možnosti jednoho stroje zadaný úkol splnit. V takových úlohách mohou různé roboty plnit různé funkce (např. výrobní linka v automobilovém závodu), nebo spolupracovat při snaze dosáhnout společného cíle (robotický fotbal [6, 7] nebo mapování prostoru [8]). Jinou takovou úlohou může být robotický dohled prováděný rojem bezpilotních helikoptér, kdy je třeba systematicky prohledávat nebo monitorovat oblast natolik rozlehlou, že ji jeden robot sám nedokáže dostatečně pokrýt.

Samotné monitorování nějakého místa či lokality může být provedeno různými senzory vhodně zvolenými podle konkrétního zadání úkolu. Od snímání oblasti kamerou až po detekci intenzity kouře, případně různých plynů, nebezpečných látek nebo radiace v ovzduší. Konkrétní aplikací potom může být hlídání obrazů a cenností v galeriích a muzeích, kontrola lokalit s vysokým nebezpečím vzniku požáru v letních měsících pomocí termovize a detektorů kouře [9] nebo dozor nad vybranými objekty se zákazem vstupu ve vojenských prostorech. Helikoptéry mohou také asistovat záchranářům [10] například při hledání posádek havarovaných letadel v mořských vodách opět pomocí termokamer.

Formace více robotů musí při plnění takovýchto úloh respektovat vzájemná omezení plynoucí z pohybu ve skupině. Především jde o předcházení kolizím mezi jednotlivými členy roje a o zachování jejich vzájemného kontaktu (ať již vizuálního nebo komunikačního). Předpokladem pro plnění některého z uvedených úkolů je mimo jiné znalost polohy všech členů roje, tedy určení souřadnic každého robotu v prostoru. Pro zjištění polohy je možné využít některý ze způsobů globální lokalizace jakým je například GPS (Global Positioning System), nebo triangulace na základě pokrytí mobilních sítí. Jiná metoda, jakou lze zjistit polohu, je snímání optického toku obrazu pořízeného kamerou, kterou je robot vybaven [11]. Na základě dat z kamery lze poměrně přesně určit polohu každého člena roje. Udržení vzájemné lokalizační vzdálenosti mezi členy může navíc minimalizovat

případnou chybu měření pomocí korekce polohy, kdy se pozice přepočítá vůči ostatním členům roje. Tento způsob lokalizace je uvažován i v této práci.

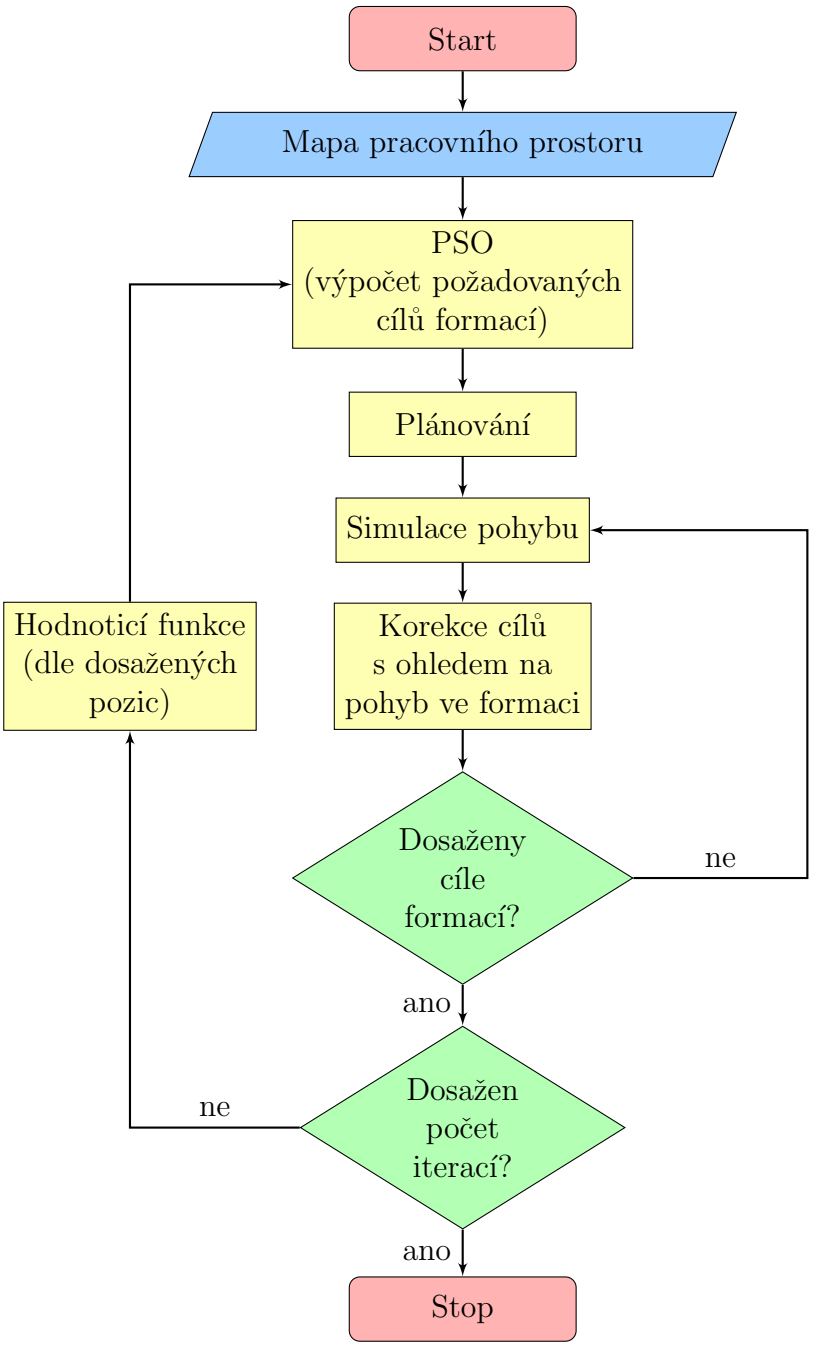
U některých typů úloh je vhodné nebo nutné naplánovat pohyb formace robotů předem. Většinou se jedná o situace, kdy je složité nebo výpočetně a časově náročné reagovat na informace z pracovního prostředí v reálném čase. V takových případech lze využít plánování trajektorií pomocí predikce pohybu. Během plánování musí roboty mimo výše uvedené podmínky pohybu ve formaci respektovat i kinematická omezení, která ovlivňují jejich pohyb.

Obsahem této práce je návrh takového algoritmu, který jako vstupní parametr převezme mapu s vyznačenými oblastmi, které je třeba monitorovat, a případně i fyzickými překážkami. Cílem je nalézt optimální stacionární rozložení (výsledné pozice) roje bezpilotních helikoptér nad těmito oblastmi. Toto rozložení by mělo maximalizovat objem dat získaný palubními přístroji helikoptér.

Pro hledání optimálního tvaru roje helikoptér je využito modifikovaného PSO (Particle Swarm Optimization) [12] algoritmu. Tento algoritmus řadíme mezi stochastické optimalizační metody založené na roji částic, které spolu vzájemně komunikují. Částice mezi sebou sdílejí informace o nejlepším dosaženém výsledku optimalizace a navzájem se jimi ovlivňují. V algoritmu se typicky uvažují bezrozměrní jedinci roje, a proto je nutné pro potřeby této práce algoritmus modifikovat tak, aby fungoval i pro jedince s reálnými rozměry a omezeními. Některé úpravy PSO algoritmu částečně vycházejí z předchozí bakalářské práce [13].

2 Definice úlohy a popis systému

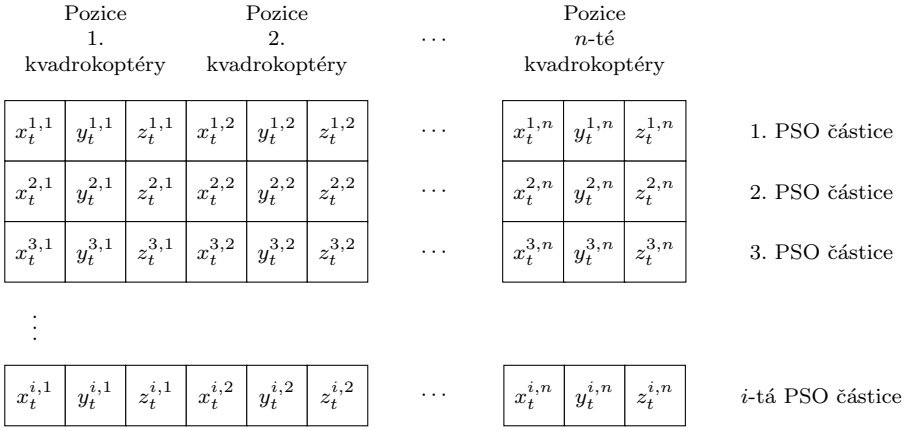
Tato práce se zabývá hledáním optimálního rozložení formace několika kvadrokoptér nad uživatelem definovanými oblastmi, které je třeba monitorovat. Optimalizace využívá počítačové simulace, jejímž výstupem jsou trajektorie pro každého člena formace navzorkované po diskrétních časových krocích. Průběh algoritmu stručně shrnuje diagram na obrázku 1.



Obrázek 1: Schéma algoritmu

Na vstup algoritmu předáme mapu oblasti popisující pracovní prostor, ve kterém bude řešení úlohy probíhat. V mapě jsou zachyceny informace o pozicích oblastí určených k monitorování a dále o výskytu bezletových zón (překážek), které se v prostoru vyskytují. Tvorbě takovéto mapy je věnována kapitola 4.

PSO algoritmus popsaný v kapitole 5 využívá k hledání optimálního řešení několika virtuálních formací kvadrokoptér. Každá formace představuje částici s níž PSO algoritmus pracuje. Tvar celého roje o i částicích skládajících se z n kvadrokoptér reprezentuje obrázek 2. Poloha každé kvadrokoptéry v t-té iteraci je popsána souřadnicemi $[x; y; z]$.



Obrázek 2: Schéma roje PSO algoritmu

Pomocí údajů z mapy jsou na základě pravidel PSO určeny pozice formací, jichž má být během jedné iterace algoritmu dosaženo. V první iteraci probíhá inicializace PSO algoritmu, během které jsou cílové pozice určeny náhodně. V dalších iteracích je využito k výpočtu pozic formací mimo pravidel PSO i tzv. hodnoticí funkce (viz podkapitola 4.2). Hodnoticí funkce určí, jak dobré rozmístění různé formace kvadrokoptér zaujímají vzhledem k oblastem, které chceme monitorovat.

K požadovaným cílům formací je naplánována cesta pomocí grafu viditelnosti a algoritmu na prohledávání stavového prostoru. V této práci je použit A* algoritmus. Popisem těchto metod se zabývá podkapitola 4.3. Je vytvořen seznam bodů, které musí každá kvadrokoptéra každé formace v daném pořadí navštívit, aby se během cesty k cíli vyhnula bezletovým zónám.

Během simulace musí být zohledněna kinematická omezení robotů. Použité kvadrokoptéry lze charakterizovat pohybovými rovnicemi, z nichž vznikne jejich dynamický model. Regulátor navržený za základě tohoto modelu počítá v každém diskrétním časovém okamžiku odchylku současné polohy od požadovaného stavu a určí akční zásahy, které jsou nutné k dosažení cíle. Popisem dynamického modelu a regulátoru se zabývá kapitola 3.

Při pohybu ve formaci je nutné respektovat podmínky pohybu více robotů ve skupině. Roboty musí předcházet vzájemným kolizím a navíc může být zohledněn

požadavek relativní lokalizace, tedy udržení maximální vzájemné vzdálenosti. Tyto faktory ovlivňují, zda se vůbec kvadrokoptéry dokáží dostat až do cílů určených PSO algoritmem. Pokud nelze uvedené podmínky splnit, cíle jsou podle aktuálních pozic upraveny tak, aby mohly být při zachování pravidel pohybu ve formaci dosaženy. Řešení vzájemných kolizí a relativní lokalizaci (udržení maximální vzdálenosti mezi členy formace) je věnována kapitola 6.

Výstupem algoritmu jsou trajektorie pro každého člena formace vedoucí do pozic tvořících optimální rozmístění. Algoritmus musí zajistit, že během pohybu po těchto trajektoriích nedojde ke kolizím mezi členy formace, budou splněny podmínky relativní lokalizace a že trajektorie respektují kinematická omezení kvadrokoptér.

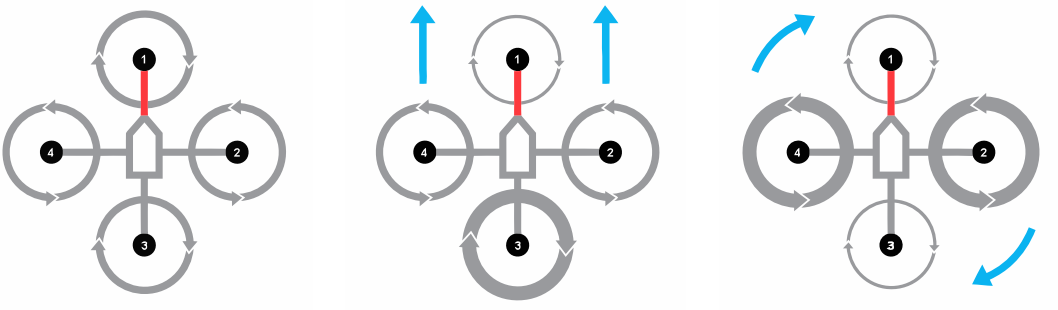
3 Kvadrokoptéra a její dynamický model

3.1 Kvadrokoptéra

Při řešení úlohy se uvažuje použití čtyřrotorové helikoptéry, tzv. kvadrokoptéry, znázorněné na obrázku 3. Kvadrokoptéry mají oproti klasické konstrukci vrtulníku s jedním nosným a jedním vyrovnávacím (ocasním) rotorem některé výhody. Velkou předností strojů takovéto konstrukce je jejich obratnost, což dokazují například experimenty s inverzním kyvadlem [14] nebo série pokusných průletů a přistání ve velmi omezeném prostoru [15]. Za další výhodu lze považovat relativně malý rozměr, díky kterému jsou kvadrokoptéry vhodné i pro nasazení uvnitř budov.

Obrázek 3: Příklad kvadrokoptéry [1]

Kvadrokoptéra je tvořena nosnou konstrukcí (rámem) tvaru kříže. Na konci každého ramene je umístěn jeden ze čtyř rotorů tak, že středy rotorů leží ve vrcholech pomyslného čtverce. Přes střed protilehlé rotory tvoří dvojici, jejíž vrtule rotují stejným směrem. Každá dvojice má jiný směr otáčení rotorů, jak je znázorněno na obrázku 4.



(a) realizace rotorů (b) pohyb vpřed (c) rotace na místě

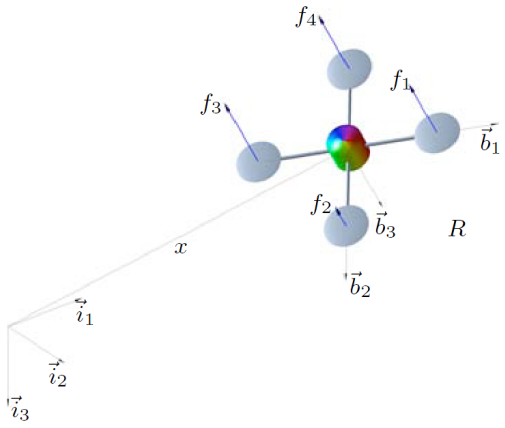
Obrázek 4: Realizace rotorů a pohyb kvadrokoptéry [2]

Při takovéto koncepci se reakční momenty jednotlivých rotorů, v případě že generují stejně velký tah, navzájem kompenzují. Uspořádání tedy plní funkci ocasního rotoru klasické konstrukce vrtulníku, který vyrovnává reakční moment rotoru nosného. V případě, že by se všechny čtyři rotory točily stejným směrem, reakční moment by byl vyrovnáván otáčením celého rámu kvadrokoptéry. To by v podstatě znemožňovalo jakékoliv praktické využití. Ve středu rámu je umístěn řídicí a komunikační modul, případně další zařízení obsluhující připojenou elektroniku. Obrázek 4 dále znázorňuje očekávaný pohyb kvadrokoptéry (modré šipky) při různých točivých momentech různých rotorů (tloušťka šedé šipky je úměrná velikosti točivého momentu).

3.2 Dynamický popis

Nalezené optimální rozmístění roje a naplánované trajektorie pohybu kvadrokoptér pro dosažení tohoto rozmístění musí respektovat kinematická omezení pohybu skutečné kvadrokoptéry. Je tedy nutné vytvořit její dynamický popis [3], na základě kterého bude moci program přibližně odhadovat pohyb skutečné kvadrokoptéry.

Skutečná kvadrokoptéra se skládá ze čtyř identických rotorů umístěných ve vrcholech pomyslného čtvercového rámu, které generují točivý moment a tah f_i kolmý k rovině rámu, jak je znázorněno na obrázku 5.



Obrázek 5: Schéma pro popis dynamiky [3]

Zavedeme inerciální vztažnou soustavu $\{\mathbf{i_1}, \mathbf{i_2}, \mathbf{i_3}\}$ a dále soustavu spojenou s rámem kvadrokoptéry $\{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$. Počátek soustavy spojené s rámem leží ve hmotném středu kvadrokoptéry. Osy $\mathbf{b_1}$ a $\mathbf{b_2}$ leží v rovině definované středy všech čtyř rotorů. Osa $\mathbf{b_3}$ je kolmá na tuto rovinu a směřuje opačným směrem, než je směr tahu vrtulí rotorů.

Pro popis kvadrokoptéry bude dále v textu použito následující značení:

$m \in \mathbb{R}$	celková hmotnost kvadrokoptéry
$J \in \mathbb{R}^{3\times3}$	matice setrvačnosti vzhledem k soustavě spojené s rámem kvadrokoptéry
$R \in \mathsf{SO(3)}$	rotační matice ze soustavy spojené s rámem do inerciální vztažné soustavy je součástí grupy všech rotací kolem počátku trojrozměrného Euklidova prostoru, více v [16]
$\mathbf{\Omega} \in \mathbb{R}^3$	úhlová rychlost soustavy spojené s rámem
$\mathbf{x} \in \mathbb{R}^3$	poloha hmotného středu v inerciální vztažné soustavě
$\mathbf{v} \in \mathbb{R}^3$	rychlost hmotného středu v inerciální vztažné soustavě
$d \in \mathbb{R}$	vzdálenost hmotného středu a středu i-tého rotoru v rovině $\mathbf{b_1}$, $\mathbf{b_2}$
$f_i \in \mathbb{R}$	tah generovaný vrtulí i-tého rotoru v ose $-\mathbf{b_3}$
$\tau_i \in \mathbb{R}$	točivý moment generovaný vrtulí i-tého rotoru v ose $\mathbf{b_3}$
$f \in \mathbb{R}$	celkový tah, $f = \sum_{i=1}^{4} f_i$
$\mathbf{M} \in \mathbb{R}^3$	celkový moment v soustavě spojené s rámem kvadrokoptéry

Z definice rotační matice $R \in \mathsf{SO(3)}$ (rotační matice R náleží do množiny speciálních ortogonálních grup, platí $R^T R = I$) plyne, že směr osy $\mathbf{b_i}$ soustavy spojené s rámem kvadrokoptéry lze vyjádřit jako $R\mathbf{e_i}$ v inerciální vztažné soustavě, kde $\mathbf{e_1} = [1;0;0], \mathbf{e_2} = [0;1;0], \mathbf{e_3} = [0;0;1] \in \mathbb{R}^3$. Odtud také lze v inerciální soustavě vyjádřit celkový tah jako $-fR\mathbf{e_3} \in \mathbb{R}^3$.

Pohybové rovnice kvadrokoptéry lze podle [3] zapsat jako

$$\dot{\mathbf{x}} = \mathbf{v}, \tag{1}$$

$$m\dot{\mathbf{v}} = mg\mathbf{e_3} - fR\mathbf{e_3}, \tag{2}$$

$$\dot{R} = R\hat{\Omega}, \tag{3}$$

$$J\dot{\Omega} + \mathbf{\Omega} \times J\mathbf{\Omega} = \mathbf{M}, \tag{4}$$

kde $\hat{\cdot} : \mathbb{R}^3 \to \mathsf{SO(3)}$, a tedy

$$\hat{\Omega} = \begin{pmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{pmatrix}. \tag{5}$$

3.3 Regulátor

Navržený regulátor převzatý z [3] slouží k výpočtu akčních zásahů (celkového tahu rotorů a celkového momentu v soustavě spojené s rámem kvadrokoptéry) nutných k dosažení cíle. Tyto akční zásahy se počítají na základě odchylek od požadovaného výsledného stavu (poloha v prostoru a natočení rámu). Při znalosti těchto akčních veličin můžeme pomocí pohybových rovnic určit přibližnou polohu kvadrokoptéry v dalším časovém okamžiku.

Odchylky od výsledného stavu lze vyjádřit jako

$$\mathbf{e_x} = \mathbf{x} - \mathbf{x}_d, \tag{6}$$

$$\mathbf{e_v} = \mathbf{v} - \mathbf{v}_d, \tag{7}$$

kde index d značí požadovaný stav.

Z výše spočtených odchylek a zvolených kladných konstant regulátoru k_x, k_v, k_R, k_Ω můžeme následně určit potřebnou změnu vektoru $\mathbf{b_3}$ a vyjádřit další potřebné vztahy jako

$$\mathbf{b_{3}}_d = -\frac{-k_x\mathbf{e_x} - k_v\mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_d}{\| - k_x\mathbf{e_x} - k_v\mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_d\|}, \tag{8}$$

$$\mathbf{b_{2}}_d = \frac{\mathbf{b_{3}}_d \times \mathbf{b_{1}}_d}{\|\mathbf{b_{3}}_d \times \mathbf{b_{1}}_d\|}, \tag{9}$$

$$R_d = [\mathbf{b_{2}}_d \times \mathbf{b_{3}}_d, \mathbf{b_{2}}_d, \mathbf{b_{3}}_d], \tag{10}$$

přičemž vektor $\mathbf{b_{1}}_d$ je dalším vstupním parametrem regulátoru a značí směr, kterým má být v cílové pozici kvadrokoptéra natočena. Význam plyne z obrázku 5.

Na základě uvedených vztahů lze určit i odchylky rotační matice a úhlové rychlosti

$$\mathbf{e_R} = \frac{1}{2}(R_d^T R - R^T R_d)^\vee, \tag{11}$$

$$\mathbf{e_\Omega} = \Omega - R^T R_d \Omega_d, \tag{12}$$

kde $\cdot^\vee : \mathsf{SO(3)} \to \mathbb{R}^3$.

Z výše uvedeného lze dopočítat potřebné akční zásahy f a M.

$$f = -(k_x\mathbf{e_x} - k_v\mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_d)R\mathbf{e_3} \tag{13}$$

$$\mathbf{M} = -k_R\mathbf{e_R} - k_\Omega\mathbf{e_\Omega} + \mathbf{\Omega} \times J\mathbf{\Omega} - J(\hat{\Omega}R^T R_d\mathbf{\Omega}_d - R^T R_d\dot{\mathbf{\Omega}}_d) \tag{14}$$

Uvažujeme, že tah každé vrtule je přímo úměrný točivému momentu generovanému příslušnou vrtulí. První a třetí vrtule (viz obrázek 4) se točí po směru hodinových ručiček a druhá a čtvrtá opačně. Při tom vytvářejí tah f_i v záporném směru osy $\mathbf{b_3}$. Točivý moment generovaný vrtulí i-tého rotoru lze vyjádřit jako $\tau_i = (-1)^i c_{\tau f} f_i$ pro určitou konstantu $c_{\tau f}$. Jelikož rotory mohou dosáhnout pouze omezeného točivého momentu, využijeme vztahu (15) k vypočtení celkového tahu a celkového momentu, který respektuje možnosti rotorů. Z rovnic spočteme hodnoty f_i, a pokud je to nutné, shora nebo zdola je omezíme podle rozsahu tahu rotoru. Poté je opět dosadíme do vztahu a vyjádříme celkový tah f a celkový moment $\mathbf{M}$.

$$\begin{pmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} \tag{15}$$

Při znalosti hodnot f a $\mathbf{M}$ lze dále určit polohu, rychlost, prvky rotační matice a úhlovou rychlost soustavy spojené s rámem kvadrokoptéry v dalším časovém okamžiku jako

$$\mathbf{v_{new}} = \frac{(mg\mathbf{e_3} - fR\mathbf{e_3})}{m}\Delta t + \mathbf{v}, \tag{16}$$

$$\mathbf{x_{new}} = \mathbf{v_{new}}\Delta t + \mathbf{x}, \tag{17}$$

$$\mathbf{\Omega_{new}} = J^{-1}(\mathbf{M} - \mathbf{\Omega} \times J\mathbf{\Omega})\Delta t + \mathbf{\Omega}, \tag{18}$$

$$R_{new} = RR_{\Omega}, \tag{19}$$

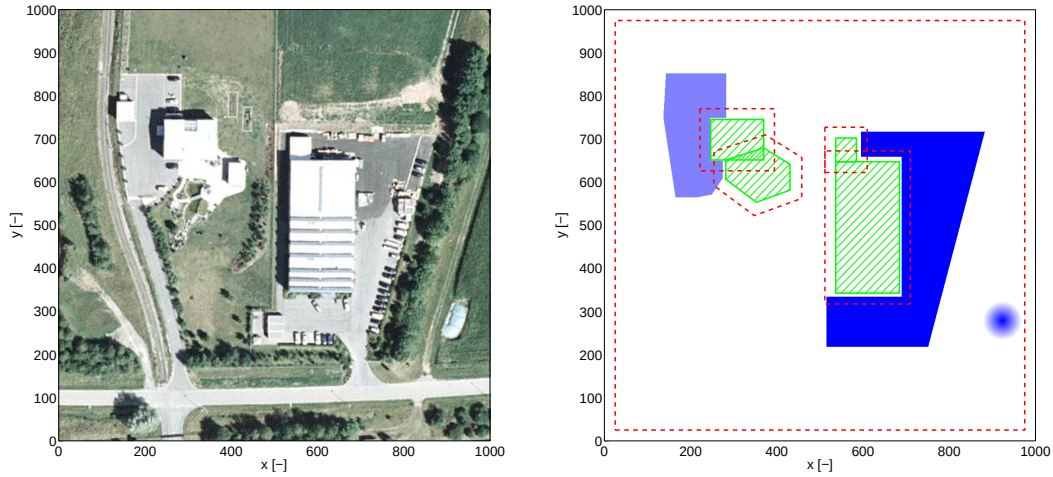
kde Δt je časový krok (další vstupní parametr regulátoru) a $R_\Omega \in \mathsf{SO(3)}$ je rotační matice kolem osy určené jednotkovým vektorem $[\Omega_x, \Omega_y, \Omega_z] \in \mathbb{R}^3$ vyjádřená pomocí Rodriguezovy rotační formule.

4 Simulační prostředí

Prostředí, v němž probíhá řešení úlohy, je trojrozměrný pracovní prostor se souřadným systémem $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$. Rozměry tohoto prostoru v rovině $\{\mathbf{x}, \mathbf{y}\}$ odpovídají rozměrům oblasti, kde budou kvadrokoptéry v praxi nasazeny. Vertikální rozměr z je omezen dostupem[1], nebo nastavením maximální povolené letové výšky. Mapa oblasti (viz obrázek 6b), která je vstupním parametrem celého algoritmu, obsahuje informaci o lokalitách, které je třeba monitorovat (modré oblasti) a o bezletových zónách, resp. o překážkách (zeleně šrafované polygony), které se v prostoru vyskytují.

4.1 Tvorba mapy

Mapa oblasti je tvořena dvěma složkami. První složku představuje matice $A_{m,n}$, ve které je zachyceno rozložení lokalit, které chceme monitorovat. Druhá složka je seznam vrcholů bezletových zón při pohledu shora (souřadnice $[x; y]$), přičemž bezletová zóna je definována jako obecný konvexní n-úhelník. Podkladem pro vytvoření takovéto mapy může být například satelitní snímek oblasti, ve které mají kvadrokoptéry operovat. V ilustračním příkladu bude úkolem monitorovat části firemního areálu zobrazeného na obrázku 6a.



(a) satelitní snímek oblasti [17]

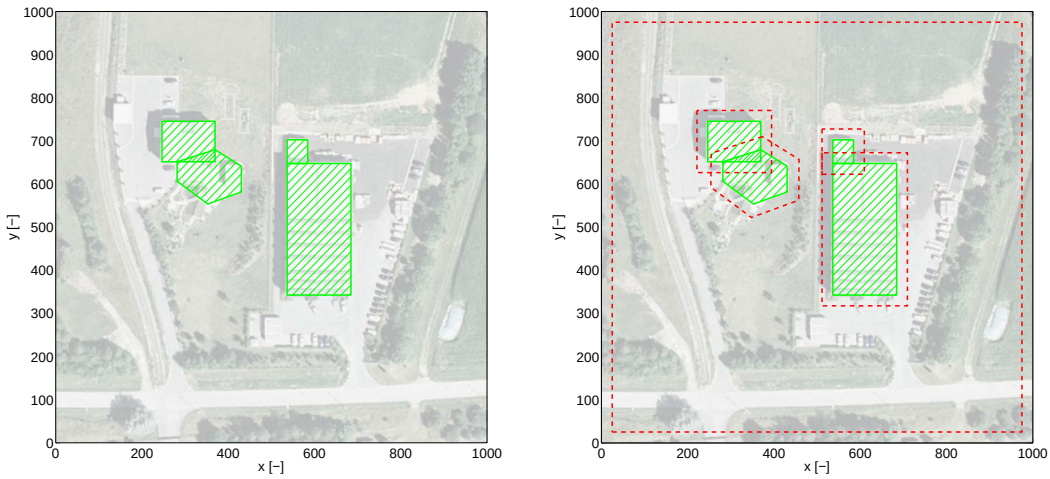
(b) hotová mapa pro simulaci

Obrázek 6: Mapa pro simulace

Nejprve jsou do mapy zakresleny bezletové zóny, jak je patrné z obrázku 7a. Plánovací algoritmus musí zajistit, aby žádná z kvadrokoptér nenarušila konvexní obal bezletové zóny. Za tímto účelem jsou překážky rozšířeny o ochranné pásmo vyznačené na obrázku 7b, jehož šířka je úměrná délce dráhy, na které kvadrokoptéra dokáže z maximální rychlosti zastavit. Při tom vycházíme z předpokladu,

[1] největší možná výška, které lze letadlem nebo vrtulníkem dosáhnout při plném výkonu motorů

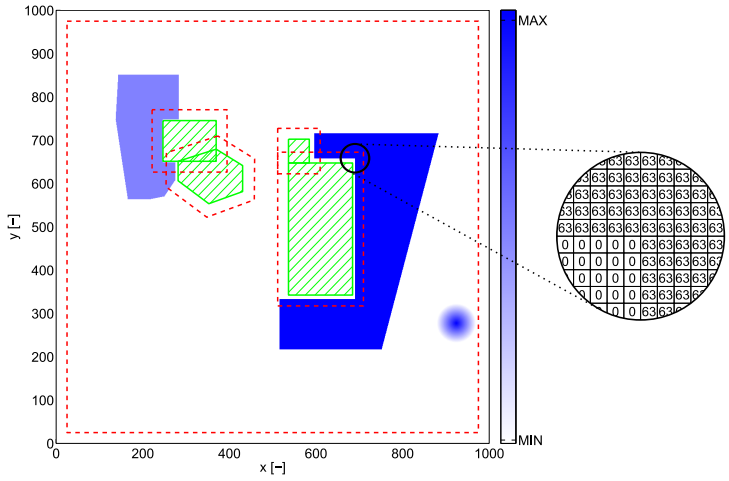
že délka této dráhy je větší než rozměry kvadrokoptéry. Obdobné ochranné pásmo je vytvořeno i kolem celé mapy, aby se kvadrokoptéry nemohly dostat mimo uvažovaný prostor.



(a) vložení bezletových zón (b) rozšíření zón o ochranné pásmo

Obrázek 7: Zakreslování bezletových zón podle satelitního snímku

Poté jsou do mapy zaneseny lokality určené k monitorování (modrá pole na obrázku 8). Taková lokalita může mít různou důležitost odpovídající potřebě ji monitorovat. Důležitost budeme definovat hodnotou na škále $0 - 63$, kdy hodnota 63 (tmavá barva) má prioritu nejvyšší. Algoritmus při výpočtu logicky upřednostňuje místa s vyšší prioritou v situaci, kdy nelze dosáhnout úplného pokrytí oblastí kvůli nedostatečnému počtu nasazených kvadrokoptér. Do matice $A_{m,n}$ je do každého prvku vložena hodnota, která odpovídá prioritě daného místa.

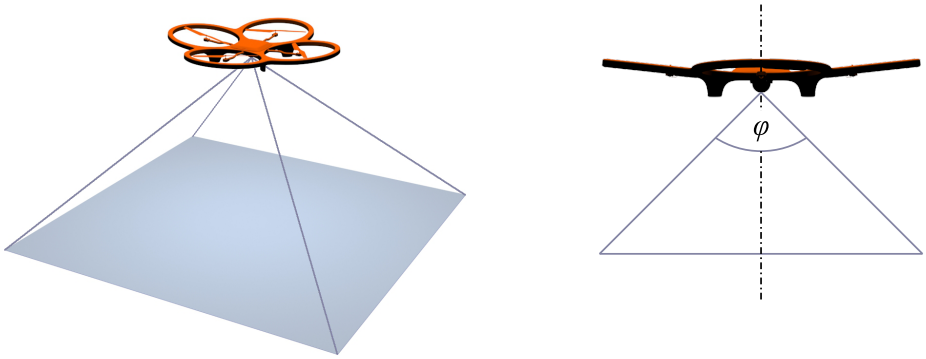


Obrázek 8: Vkládání lokalit určených k monitorování

4.2 Hodnoticí funkce

Pro ohodnocení kvality aktuální polohy roje kvadrokoptér v pracovním prostoru je třeba vytvořit hodnoticí funkci.

V ilustračním příkladu hlídání průmyslového areálu předpokládáme osazení každého člena formace kamerou. Kamera je instalována na rám kvadrokoptéry tak, že míří kolmo dolů a osy souřadného systému snímané oblasti $\{\mathbf{x}', \mathbf{y}'\}$ jsou vždy rovnoběžné s osami $\{\mathbf{x}, \mathbf{y}\}$ simulačního prostředí. Nechť zorný úhel kamery je $\varphi = \frac{\pi}{2}$ a snímaná oblast čtvercového půdorysu, jak je znázorněno na obrázku 9.



(a) snímaná oblast (b) zorný úhel kamery

Obrázek 9: Snímaná oblast (3D model [4])

Při znalosti pozice kvadrokoptéry v simulačním prostředí (souřadnice $[x; y; z]$) lze snadno dopočítat velikost snímané oblasti pomocí

$$r = z \tan \frac{\pi}{4}, \tag{20}$$

kde číslo r udává délku poloviny strany snímaného čtverce a s jeho znalostí můžeme snadno dopočítat pozice všech vrcholů čtverce v rovině $\{\mathbf{x}, \mathbf{y}\}$.

Dále je definována optimální letová výška a jí příslušný obsah snímaného čtverce, při které jsou snímky pořízené kamerou v dostačující kvalitě (obsahují optimální objem dat). Objem dat obsažený ve snímku lze vyjádřit vztahem

$$aoi = \min(aoi_{opt}, \frac{S_{opt}}{S} aoi_{opt}), \tag{21}$$

kde aoi je objem dat ve snímku, aoi_{opt} je objem dat, který lze v optimální letové výšce získat, S_{opt} je obsah snímaného čtverce příslušející optimální výšce a S je obsah snímané oblasti odpovídající aktuální výšce kvadrokoptéry.

Je definována nulová matice $Z_{m,n}$, jejíž rozměry odpovídají rozměrům matice $A_{m,n}$. V matici $A_{m,n}$ jsou zaneseny lokality, které chceme monitorovat. Pro všechny členy formace opakujeme postup popsaný v následujícím odstavci.

Z matice $Z_{m,n}$ vytvoříme submatici $Y_{o,p}$, jejíž pozice v matici $Z_{m,n}$ je dána výše získanými souřadnicemi vrcholů snímaného čtverce, zaokrouhlenými na nejbližší celočíselné hodnoty. Prvky submatice $Y_{o,p}$, jejichž hodnota je nižší než aktuální *aoi*, jsou nahrazeny hodnotou *aoi* získanou podle vztahu (21). Submatice $Y_{o,p}$ je potom vrácena na své původní místo v matici $Z_{m,n}$.

Hodnotu f hodnoticí funkce získáme nakonec pomocí vztahu (23) jako součet všech prvků matice $R_{m,n}$, přičemž

$$R_{i,j} = \min(0, A_{i,j} - Z_{i,j}). \tag{22}$$

$$f = \sum_{i=1}^{m} \Big(\sum_{j=1}^{n} R_{i,j} \Big) \tag{23}$$

4.3 Plánování trajektorií

Během optimalizace rozložení formace v prostoru jsou kvadrokoptérám postupně generovány nové cíle. Jelikož prostředí obsahuje bezletové zóny (překážky), je nutné najít mezi aktuální polohou a definovaným cílem takovou trajektorii, která se těmto zónám vyhne. Požadované trajektorie budeme hledat pomocí grafu viditelnosti a algoritmu pro hledání nejkratší cesty v takto získaném grafu.

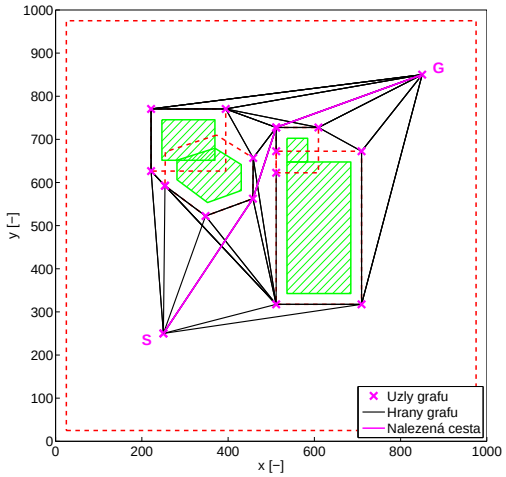
4.3.1 Graf viditelnosti

Graf viditelnosti [18] je seznam uzlů (bodů, souřadnic v prostoru) a hran (spojnice mezi nimi) spojujících dva uzly, mezi kterými je přímá viditelnost (není mezi nimi překážka). Uzly v ilustračním příkladu představují vrcholy ochranného pásma bezletových zón, počáteční bod S a cílový bod G. Hrany jsou potom definovány jako takové spojnice dvou uzlů, jejichž žádná část neleží v žádném ochranném pásmu. Ostatní členy formace, resp. vrcholy jejich konvexního obalu jako uzly grafu neuvažujeme, neboť pozice jednotlivých kvadrokoptér se neustále mění a bylo by nutné neustále přepočítávat trajektorie, což by zvýšilo výpočetní náročnost. I když je simulační prostředí trojrozměrné, graf viditelnosti je zde využit pouze v rovině $\{\mathbf{x}, \mathbf{y}\}$. Důvod tohoto postupu je společně se způsobem předcházení kolizím popsán v kapitole 6. Graf viditelnosti pro ilustrační příklad je znázorněn na obrázku 10.

Předností této metody je fakt, že nalezne řešení vždy, pokud nějaké existuje, a tato nalezená cesta je při použití vhodného prohledávacího algoritmu nejkratší možná. Naopak nevýhodou grafu viditelnosti je to, že trajektorie plánuje v těsné blízkosti překážek. Další nevýhodou je výpočetní náročnost v případě, že je graf značně rozlehlý (velký počet uzlů a hran). Pro demonstraci použití optimalizační metody s integrovaným plánováním pohybu je ale dostačující.

4.3.2 A* algoritmus

A* algoritmus [19] je algoritmus určený k prohledávání stavového prostoru. Zaručuje nalezení optimálního řešení, pokud takové existuje. Stavový prostor v tomto případě odpovídá grafu viditelnosti, v němž hledáme nejkratší cestu. Algoritmus je řazen mezi informované metody, což znamená, že uchovává informaci o celém stavovém prostoru. Tato informace mu umožňuje odhadnout, jak daleko se nachází od požadovaného cílového stavu. Odhad je realizován pomocí tzv. heuristické funkce $h(n)$. Čím lepší je heuristická funkce, tím rychleji dojde k nalezení optimálního řešení při prohledání menší části stavového prostoru. Na obrázku 10 je znázorněna optimální cesta z bodu S do bodu G nalezená A* algoritmem v grafu viditelnosti.



Obrázek 10: Graf viditelnosti a cesta naplánovaná A* algoritmem

Všechny prvky stavového prostoru (všechny uzly grafu) jsou ohodnoceny funkcí

$$\hat{f}(n) = g(n) + h(n), \tag{24}$$

kde $g(n)$ je cena dosavadní cesty z počátku S do uzlu n a $h(n)$ je odhad ceny z uzlu n do cíle G. Celkovou cenu $\hat{f}(n)$ se snažíme minimalizovat. Cena dosavadní cesty $g(n)$ je pro tento případ definována jako součet euklidovských vzdáleností mezi dvěma sousedními uzly na trase vedoucí do aktuálního uzlu. Dále musí platit, že heuristická funkce $h(n)$ je přípustná, tedy že odhad ceny není nikdy větší, než je skutečná cena $h(n)^*$.

$$0 \leq h(n) \leq h(n)^*; \forall n \tag{25}$$

Jako heuristickou funkci uvažujeme euklidovskou vzdálenost mezi aktuálním a cílovým uzlem, neboli

$$h(n) = \sqrt{(x_{actual} - x_G)^2 + (y_{actual} - y_G)^2}. \tag{26}$$

Takováto heuristická funkce je přípustná, neboť v simulačním prostředí bude odpovídat skutečné zbývající vzdálenosti $h(n)^*$, nenachází-li se mezi aktuální pozicí a cílem bezletová zóna, nebo bude docházet k prodlužování odhadované zbývající dráhy kvůli oblétání bezletových zón.

Algoritmus si udržuje dva seznamy uzlů OPEN a CLOSED a dále seznam rodičů obsahující informaci, ze kterého uzlu bylo dosaženo následující pozice v grafu. V seznamu OPEN jsou uchovávány expanzí[2] získané uzly, které samy dosud expandovány nebyly. Pokud je v průběhu expanze navštíven uzel, který je již zahrnut v seznamu OPEN, proběhne porovnání cen obou cest ze startu k tomuto uzlu (analogicky platí pro seznam CLOSED). Seznam rodičů je případně upraven podle trasy s nižší cenou. V seznamu CLOSED se ukládají uzly, které již byly expandovány. Před každou další expanzí se ze seznamu OPEN vybere nejvhodnější uzel na základě minimální hodnoty funkce $\hat{f}(n)$. A* algoritmus popisuje pseudokód 1.

```
Input: StartNode – počáteční pozice
       EndNode – cílová pozice

OPEN := [StartNode];                                   // seznam OPEN
CLOSED := [ ];                                         // seznam CLOSED
while OPEN ≠ [ ] do
    current = best(OPEN);   // vyber uzel s nejlepší hodnotou f̂(n)
    OPEN := OPEN − current;          // odeber vybraný uzel z OPEN
    CLOSED := CLOSED + current;// přidej vybraný uzel do CLOSED
    if current = EndNode then
        return path;                 // vrať cestu z počátku do cíle
    else
        E := expand(current);                  // expanduj vybraný uzel
        foreach e_i ∈ E do
            if (e_i ∉ OPEN) and (e_i ∉ CLOSED) then
                OPEN := OPEN + e_i;
            if (e_i ∈ OPEN) and (f̂(e_i ∈ E) < f̂(e_i ∈ OPEN)) then
                OPEN := OPEN − (e_i ∈ OPEN);
                OPEN := OPEN + (e_i ∈ E);
            if (e_i ∈ CLOSED) and (f̂(e_i ∈ E) < f̂(e_i ∈ CLOSED)) then
                CLOSED := CLOSED − (e_i ∈ CLOSED);
                OPEN := OPEN + (e_i ∈ E);
return failure;                  // vrať chybu - cesta nebyla nalezena
```

Pseudokód 1: A* algoritmus

[2]Expanzí se navštíví všechny sousední uzly, kterých lze z aktuálního uzlu dosáhnout (neleží za překážkou).

5 Particle Swarm Optimization algoritmus

Algoritmus PSO [12] (Particle Swarm Optimization) řadíme mezi stochastické optimalizační metody založené na rojích částic, které spolu vzájemně komunikují. Tato metoda byla vyvinuta J. Kennedym a R. C. Eberhartem a poprvé představena v roce 1995. Každá částice (jedinec) roje nese informaci o své aktuální poloze, rychlosti a o nejlepším dosaženém výsledku hledání společně s jeho polohou v prohledávaném prostoru. Částice mezi sebou tyto informace sdílejí a navzájem se jimi ovlivňují. Algoritmus iterativně udává jedincům roje jejich novou polohu a rychlost na základě polohy celkově nejlepšího výsledku hledání, nejlepšího výsledku hledání konkrétní částice a polohy a rychlosti této částice z minulé iterace.

Mezi výhody tohoto algoritmu patří jeho snadná implementace a možnost použití v prostředí s mnoha lokálními extrémy. S rostoucím počtem jedinců se snižuje pravděpodobnost uváznutí v lokálním optimu. Za nevýhodu naopak lze považovat například fakt, že algoritmus nezaručuje nalezení globálního optima.

5.1 Varianty PSO algoritmu

Algoritmus začíná náhodným rozmístěním n částic v prostoru a poloha každé částice je ohodnocena hodnoticí funkcí f. Každé částici je dále přidělena náhodná počáteční rychlost. Pohyb roje počítá algoritmus v diskrétních časových krocích, jejichž počet je předem definován (odpovídá počtu iterací algoritmu).

5.1.1 Základní varianta

Podstatou základní varianty PSO algoritmu, z níž ostatní modifikace vycházejí, jsou rovnice (27) a (28),

$$\mathbf{v}^{\mathbf{i}}_{\mathbf{t+1}} = \mathbf{v}^{\mathbf{i}}_{\mathbf{t}} + c_1 r_1 (\mathbf{p}^{\mathbf{i}} - \mathbf{x}^{\mathbf{i}}_{\mathbf{t}}) + c_2 r_2 (\mathbf{p}^{\mathbf{g}} - \mathbf{x}^{\mathbf{i}}_{\mathbf{t}}), \tag{27}$$

$$\mathbf{x}^{\mathbf{i}}_{\mathbf{t+1}} = \mathbf{x}^{\mathbf{i}}_{\mathbf{t}} + \mathbf{v}^{\mathbf{i}}_{\mathbf{t+1}}, \tag{28}$$

kde význam použitých symbolů je následující:

$\mathbf{x}^{\mathbf{i}}_{\mathbf{t}}$	poloha i-té částice v t-tém diskrétním časovém kroku (iteraci) algoritmu
$\mathbf{v}^{\mathbf{i}}_{\mathbf{t}}$	rychlost i-té částice
$\mathbf{p}^{\mathbf{i}}$	pozice nejlepšího dosaženého výsledku hledání i-té částice
$\mathbf{p}^{\mathbf{g}}$	pozice celkově nejlepšího dosaženého výsledku hledání
c_1, c_2	parametry osobní a sociální inteligence
r_1, r_2	náhodná čísla z intervalu $\langle 0, 1 \rangle$

V základní variantě se vyskytují dva parametry (konstanty c_1 a c_2), které je třeba ladit. Tyto konstanty udávají váhu, s jakou částice reagují na informaci o svém a o celkovém nejlepším dosaženém výsledku hledání. Podle zdroje [12] jsou obě konstanty nastaveny stejně, a sice na hodnotu $c_1 = c_2 = 2$. Vhodnost tohoto nastavení bude později ověřena sérií experimentů a konstanty budou nastaveny podle nejlepšího dosaženého výsledku.

Základní PSO algoritmus popisuje pseudokód 2. Význam symbolů použitých v pseudokódu je totožný s významem uvedeným dříve. Navíc se zde objevuje symbol p^i, který znamená hodnotu hodnoticí funkce f odpovídající pozici $\mathbf{p^i}$. Hodnotu funkce f se snažíme minimalizovat.

Input: numberOfIterations – počet iterací algoritmu
numberOfParticles – počet částic roje
c_1, c_2

$swarmInitialization()$; `// rozmísti částice do prostoru, ohodnoť pozice hodnoticí funkcí a nastav počáteční rychlost`

for t = 1; t ≤ numberOfIterations **do**
  for i = 1; i ≤ numberOfParticles **do**
    $f_t^i = evaluateFitnessFcn(\mathbf{x_t^i})$;
      `// spočti hodnoticí funkci pro aktuální polohu` $\mathbf{x_t^i}$
    if $f_t^i \leq \mathrm{p^i}$ **then**
      $\mathrm{p^i} = f_t^i$; `// aktualizuj hodnotu nejlepšího výsledku`
      $\mathbf{p^i} = \mathbf{x_t^i}$; `// aktualizuj pozici nejlepšího výsledku`
  $\mathbf{p^g} = getPositionWithBestFitness(\mathbf{p})$;
    `// vrať pozici celkově nejlepšího výsledku`
  for i = 1; i ≤ numberOfParticles **do**
    $\mathrm{r_1} = rand(0, 1)$; `//` $\mathrm{r_1}$ `budiž náhodné číslo z intervalu` $\langle 0, 1 \rangle$
    $\mathrm{r_2} = rand(0, 1)$;
    $\mathbf{v_{t+1}^i} = \mathbf{v_t^i} + \mathrm{c_1 r_1}(\mathbf{p^i} - \mathbf{x_t^i}) + \mathrm{c_2 r_2}(\mathbf{p^g} - \mathbf{x_t^i})$;
      `// urči rychlost částice v další iteraci`
    $\mathbf{x_{t+1}^i} = \mathbf{x_t^i} + \mathbf{v_{t+1}^i}$; `// urči polohu částice v další iteraci`

Pseudokód 2: Základní varianta PSO algoritmu

Pro úspěšné nasazení je třeba definovat základní entitu (podobu částice roje), s jakou bude algoritmus pracovat. V práci [13] je za jedince roje považován jeden robot, resp. kvadrokoptéra. Takový přístup je vhodný, pokud skupina robotů hledá v uvažovaném prostoru globální optimum odpovídající konkrétnímu místu. V ilustračním příkladu ovšem globální optimum představuje ideální rozložení roje. Za částici roje budeme považovat formaci n kvadrokoptér a podle toho změníme vektory, které popisují jednotlivé jedince. Například vektor polohy i-té částice $\mathbf{x_t^i}$ v t-té iteraci bude mít v trojrozměrném pracovním prostoru velikost $3n$ prvků

a tvar podle vztahu (29), který odpovídá tvaru roje PSO algoritmu znázorněnému na obrázku 2.

$$\mathbf{x_t^i} = [x_t^{i,1}; y_t^{i,1}; z_t^{i,1}; x_t^{i,2}; y_t^{i,2}; \ldots; z_t^{i,n-1}; x_t^{i,n}; y_t^{i,n}; z_t^{i,n}] \tag{29}$$

Takto budou změněny i vektory rychlosti částic a pozice nejlepších dosažených výsledků hledání. Algoritmus v každé iteraci určí nové souřadnice všech kvadrokoptér ve formaci. Mezi stávající a novou požadovanou polohou je naplánována cesta pomocí grafu viditelnosti a A* algoritmu popsaného v podkapitole 4.3.

5.1.2 Klasická rozšíření PSO algoritmu

V základní variantě algoritmu se dnes standardně uvažuje i použití konstanty setrvačnosti, kterou lze měnit vliv rychlosti z minulé iterace. V tomto případě se jedná o bezrozměrnou konstantu, kterou je vynásobena rychlost částice z minulé iterace, a tím je ovlivněno její působení v iteraci současné. Implementace je provedena drobnou změnou vztahu (27), a sice

$$\mathbf{v_{t+1}^i} = w\mathbf{v_t^i} + c_1 r_1(\mathbf{p^i} - \mathbf{x_t^i}) + c_2 r_2(\mathbf{p^g} - \mathbf{x_t^i}), \tag{30}$$

kde w je konstanta setrvačnosti.

Hodnotu setrvačnosti je vhodné volit z intervalu $\langle 0; 1\rangle$, čímž je zaručeno, že se vliv rychlosti z minulé iterace nezvyšuje. Částice je při snižující se hodnotě w více ovlivňována pozicemi nalezených nejlepších výsledků. Hodnotu setrvačnosti lze zvolit mnoha způsoby. Například konstantní, náhodnou nebo takovou, která se bude lineárně snižovat na zvoleném intervalu s rostoucím počtem iterací. V algoritmu použijeme snižující se hodnotu setrvačnosti. To zaručí, že na počátku běhu programu bude vliv rychlosti z minulé iterace dostatečně velký na to, aby zabránil případnému uváznutí v lokálním extrému. Naopak později, kdy se očekává přiblížení ke globálnímu optimu, bude vliv minulé rychlosti nízký, čímž se sníží pravděpodobnost, že by se částice od globálního optima začala vzdalovat. Aplikované (zeleně vyznačené) rozšíření je včleněno do pseudokódu 3.

Další změnou, která je v základní variantě algoritmu běžná, je omezení rychlosti pohybu částic v rámci jedné iterace. Stejně jako v případě konstanty setrvačnosti lze rychlost jedinců stanovit jako konstantní po celou dobu běhu algoritmu, nebo ji lineárně snižovat s rostoucím číslem iterace na stanoveném intervalu mezi maximální a minimální hodnotou. Druhý postup je výhodnější, neboť značně redukuje kmitání kolem nalezeného extrému v závěru běhu algoritmu, kdy se pohyb částic předpokládá kolem globálního optima. Naopak na počátku je díky velkému kmitání prohledána větší část prostoru, což výrazně přispívá k nalezení globálního extrému. Toto omezení je v pseudokódu 3 vyznačeno červeně.

Input: numberOfIterations – počet iterací algoritmu
numberOfParticles – počet částic roje
c_1, c_2
w_{min}, w_{max} – hranice intervalu setrvačnosti
v_{min}, v_{max} – hranice intervalu maximální rychlosti částice

$swarmInitialization()$; `// rozmísti částice do prostoru, ohodnoť pozice hodnoticí funkcí a nastav počáteční rychlost`

for $t = 1; t \leq$ numberOfIterations **do**
  for $i = 1; i \leq$ numberOfParticles **do**
    $f_t^i = evaluateFitnessFcn(\mathbf{x_t^i})$;
      `// spočti hodnoticí funkci pro aktuální polohu` $\mathbf{x_t^i}$
    if $f_t^i \leq p^i$ **then**
      $p^i = f_t^i$; `// aktualizuj hodnotu nejlepšího výsledku`
      $\mathbf{p^i} = \mathbf{x_t^i}$; `// aktualizuj pozici nejlepšího výsledku`
  $\mathbf{p^g} = getPositionWithBestFitness(\mathbf{p})$;
    `// vrať pozici celkově nejlepšího výsledku`
  for $i = 1; i \leq$ numberOfParticles **do**
    $r_1 = rand(0, 1)$; `//` r_1 `budiž náhodné číslo z intervalu` $\langle 0, 1 \rangle$
    $r_2 = rand(0, 1)$;
    $w = w_{max} - (w_{max} - w_{min})\frac{i}{numberOfIterations}$;
      `// urči setrvačnost vzhledem k aktuální iteraci`
    $\mathbf{v_{t+1}^i} = w\mathbf{v_t^i} + c_1 r_1(\mathbf{p^i} - \mathbf{x_t^i}) + c_2 r_2(\mathbf{p^g} - \mathbf{x_t^i})$;
      `// urči rychlost částice v další iteraci`
    $v_{max} = v_{max} - (v_{max} - v_{min})\frac{i}{numberOfIterations}$;
      `//urči` v_{max} `vzhledem k aktuální iteraci`
    if $|\mathbf{v_{t+1}^i}| > v_{max}$ **then**
      $\mathbf{v_{t+1}^i} = \frac{\mathbf{v_{t+1}^i}}{|\mathbf{v_{t+1}^i}|} v_{max}$; `// omez rychlost na` v_{max}
    $\mathbf{x_{t+1}^i} = \mathbf{x_t^i} + \mathbf{v_{t+1}^i}$; `// urči polohu částice v další iteraci`

Pseudokód 3: PSO s použitím setrvačnosti a omezením rychlosti částic

5.1.3 Varianta s detekcí překážek

Základní verzi PSO algoritmu lze mnoha způsoby modifikovat podle úlohy, pro kterou je použita. Rozšířením, které není pro PSO algoritmus běžné, je varianta detekující dostupnost cíle. Jelikož hledání globálního optima probíhá v prostředí obsahujícím překážky, je třeba zaručit, že algoritmem spočtený cíl bude pro kvadrokoptéru dostupný. Nesmí se tedy nacházet uvnitř polygonů definujících ochranné pásmo bezletových zón. V souladu s tímto požadavkem byla vytvořena tato modifikace algoritmu.

Pokud je cílový bod detekován uvnitř ochranného pásma bezletové zóny, nebo mimo uvažovanou mapu, začne se vektor ze současné do požadované pozice po definovaných krocích zkracovat. Velikost tohoto vektoru, odpovídajícího rychlosti částice v další iteraci, je zkracována o délku r_{cut} tak dlouho, dokud není cíl dostupný (leží v uvažované mapě mimo překážku). Jelikož startovní pozice všech členů roje jsou voleny tak, aby ležely v mapě mimo překážky, je i při nulovém vektoru rychlosti splněna podmínka dostupnosti cíle.

Varianta algoritmu s detekcí překážek je popsána pseudokódem 4. Změny v algoritmu jsou vyznačeny červeně.

Jinou variantou by bylo nechat PSO algoritmus spočítat nové cíle. V algoritmu je zahrnut prvek náhody, díky kterému se bude nově spočítaná cílová pozice s vysokou pravděpodobností lišit od té původní. Velkou změnu pozice nově spočítaných cílů lze ovšem očekávat jen na začátku běhu programu, kdy se kvadrokoptéry nacházejí daleko od optimálních pozic. V závěru běhu programu budou změny cílových pozic při opakovaném výpočtu jen minimální, čímž roste pravděpodobnost, že se špatná cílová pozice při opakovaném přepočítání nezmění v dostupnou. Z těchto důvodů je v této práci použito zkracování vektoru rychlosti.

5.1.4 Varianta respektující rozměry částice

Na rozdíl od optimalizačních procesů využívajících v PSO algoritmu jedince v podobě bezrozměrných částic, je třeba zohlednit fakt, že v popisované úloze jsou použity reálné roboty. Kolizím mezi kvadrokoptérami během pohybu ze startu S do cíle G v rámci formace je věnována kapitola 6. Aby ovšem bylo alespoň teoreticky možné cílů dosáhnout, bylo nutné vytvořit funkci, která zajistí, aby algoritmem spočtené cíle ležely v dostatečné vzdálenosti od sebe.

Je stanoven minimální odstup r_o, který mezi sebou musejí kvadrokoptéry udržovat. Jeho velikost vychází ze vzdálenosti, ve které se dvě kvadrokoptéry nacházejí v kolizi, a z délky ochranného pásma zmíněného dříve. Vzdálenost mezi každými dvěma cíli kvadrokoptér musí být větší než r_o. Pokud tomu tak není, je postupováno podle schématu na obrázku 11. Každý cíl je postupně testován na počet nevyhovujících sousedních cílů v rovině $\{\mathbf{x}, \mathbf{y}\}$. Příslušné kružnice, definující bezpečný odstup, jsou vyznačeny šedou barvou. Nevyhovující sousedé cíle G_A jsou G_B a G_C, nevyhovující soused cíle G_C je pouze G_A atd. Jsou utvořeny vektory z každého nevyhovujícího souseda do testovaného cíle. Tyto vektory jsou dále upraveny, aby jejich velikost nepřímo úměrně odpovídala vzdálenosti mezi

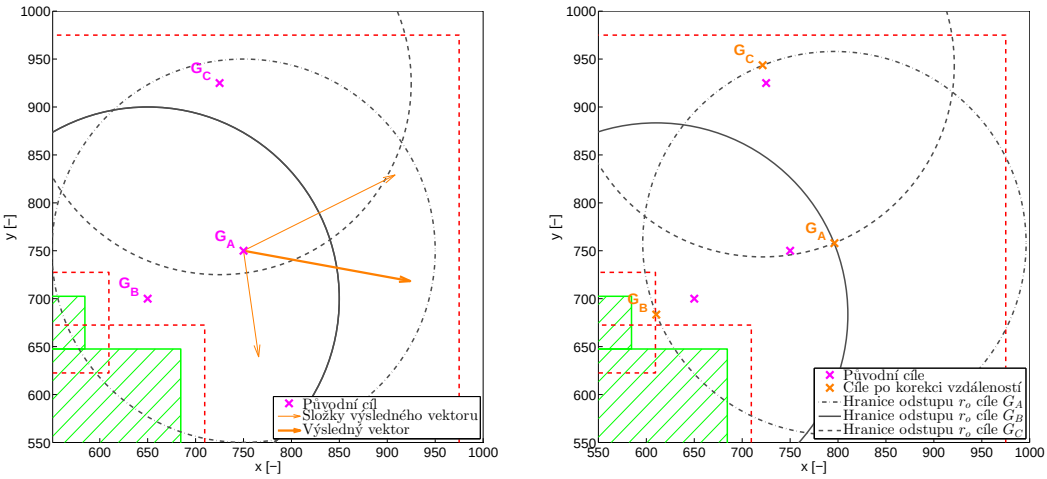
cílem a jeho nevyhovujícím sousedem. Následně jsou sečteny a výsledný vektor je znormován a přičten k pozici testovaného cíle. Tím se testovaný cíl o jednotkovou délku vzdálí od nevyhovujících sousedů. Výsledný vektor na obrázku 11a znázorňuje směr, kterým bude cíl G_A posunut v prvním kroku korekce. Nejedná se o velikost ani o celkový výsledný směr posunutí.

Iterativně se takto otestují všechny cíle a celý postup se opakuje, dokud není podmínka minimálního odstupu r_o splněna. Při testování je rovněž kontrolována poloha posunutého cíle vzhledem překážkám a hranicím simulačního prostoru. Dojde-li k posunutí cíle na nevyhovující pozici, je poslední krok úprav zrušen a pokračuje se testováním dalšího cíle. Pokud není možné výše uvedeným způsobem cíle zpřístupnit, jsou PSO algoritmem spočteny jiné cíle. Začlenění tohoto ošetření do PSO algoritmu je popsáno v pseudokódu 4 (označeno zeleně).

Při určování minimálního odstupu r_o můžeme vzít v úvahu i optimální letovou výšku (viz podkapitola 4.2). Jelikož je zřejmě nevýhodné, aby se oblasti snímané z optimální výšky překrývaly, hodnotu r_o můžeme na základě popisu hodnoticí funkce volit jako

$$r_o = \max(r_o, 2z_{opt} \tan \frac{\varphi}{2}), \tag{31}$$

kde z_{opt} je optimální letová výška a φ zorný úhel kamery.



(a) posun původních cílů

(b) cíle po korekci vzdáleností

Obrázek 11: Korekce vzdáleností cílů

Input: numberOfIterations – počet iterací algoritmu
numberOfParticles – počet částic roje
c_1, c_2
w_{min}, w_{max} – hranice intervalu setrvačnosti
v_{min}, v_{max} – hranice intervalu maximální rychlosti částice
r_{cut} – parametr hledání nového cíle
r_o – minimální odstup kvadrokoptér

swarmInitialization(); // rozmísti částice do prostoru, ohodnoť pozice hodnoticí funkcí a nastav počáteční rychlost

for $t = 1; t \leq$ numberOfIterations **do**
  for $i = 1; i \leq$ numberOfParticles **do**
    $f_t^i = evaluateFitnessFcn(\mathbf{x_t^i})$;
      // spočti hodnoticí funkci pro aktuální polohu $\mathbf{x_t^i}$
    if $f_t^i \leq p^i$ **then**
      $p^i = f_t^i$; // aktualizuj hodnotu nejlepšího výsledku
      $\mathbf{p^i} = \mathbf{x_t^i}$; // aktualizuj pozici nejlepšího výsledku
  $\mathbf{p^g} = getPositionWithBestFitness(\mathbf{p})$;
    // vrať pozici celkově nejlepšího výsledku
  for $i = 1; i \leq$ numberOfParticles **do**
    $r_1 = rand(0, 1)$; // r_1 budiž náhodné číslo z intervalu $\langle 0, 1 \rangle$
    $r_2 = rand(0, 1)$;
    $w = w_{max} - (w_{max} - w_{min})\frac{i}{numberOfIterations}$;
      // urči setrvačnost vzhledem k aktuální iteraci
    $\mathbf{v_{t+1}^i} = w\mathbf{v_t^i} + c_1 r_1(\mathbf{p^i} - \mathbf{x_t^i}) + c_2 r_2(\mathbf{p^g} - \mathbf{x_t^i})$;
      // urči rychlost částice v další iteraci
    $v_{max} = v_{max} - (v_{max} - v_{min})\frac{i}{numberOfIterations}$;
      // urči v_{max} vzhledem k aktuální iteraci
    if $|\mathbf{v_{t+1}^i}| > v_{max}$ **then**
      $\mathbf{v_{t+1}^i} = \frac{\mathbf{v_{t+1}^i}}{|\mathbf{v_{t+1}^i}|} v_{max}$; // omez rychlost na v_{max}
    $\mathbf{x_{t+1}^i} = \mathbf{x_t^i} + \mathbf{v_{t+1}^i}$; // urči polohu částice v další iteraci
    foreach (quadrotor $q \in \mathbf{x_{t+1}^i}$) **do**
      if $unreachableGoal(\mathbf{x_{t+1}^{i,q}})$ **then**
        $\mathbf{x_{t+1}^{i,q}} = cutVelocityVector(\mathbf{x_{t+1}^{i,q}}, r_{cut})$;
          // najdi náhradní cíl
    while $distanceAmongGoals(\mathbf{x_{t+1}^i}) < r_o$ **do**
      foreach (quadrotor $q \in \mathbf{x_{t+1}^i}$) **do**
        $\mathbf{x_{t+1}^{i,q}} = editGoal(\mathbf{x_{t+1}^{i,q}})$;
          // posuň cíl od nevyhovujících sousedů
    $\mathbf{v_{t+1}^i} = \mathbf{x_{t+1}^i} - \mathbf{x_t^i}$; // uprav vektor rychlosti podle cíle

Pseudokód 4: PSO s detekcí překážek a zohledňující rozměry částic

5.2 Ladění parametrů

V uvedených variantách PSO algoritmu se vyskytuje několik parametrů, jejichž nastavení ovlivňuje rychlost konvergence. Konkrétně jde o konstanty c_1, c_2 a w, o maximální rychlost částic v_{max} a o velikost minimální vzdálenosti cílů určených PSO algoritmem. Ideální nastavení uvedených parametrů pro tuto úlohu bude zjištěno na základě statistických údajů, které získáme opakovaným spuštěním PSO algoritmu pro kombinace různých hodnot těchto parametrů.

Ladění parametrů PSO algoritmu bude probíhat bez ohledu na omezení daná pohybem ve formaci. S částicemi roje se tedy pracuje jako s bezrozměrnými entitami, které se nemohou srazit a neuvažuje se ani požadavek na zachování vzájemného kontaktu (maximální vzdálenost mezi částicemi). Tím odpadá nutnost počítat pohyb kvadrokoptér postupně s malým časovým krokem, což je časově i výpočetně velice náročné. Cílových pozic spočtených PSO algoritmem je vždy dosaženo, a proto v rámci jedné iterace stačí pouze vyhodnotit hodnoticí funkci f pro určené souřadnice. Vhodnost tohoto postupu bude zpětně ověřena při simulacích s omezeními plynoucími z pohybu ve formaci.

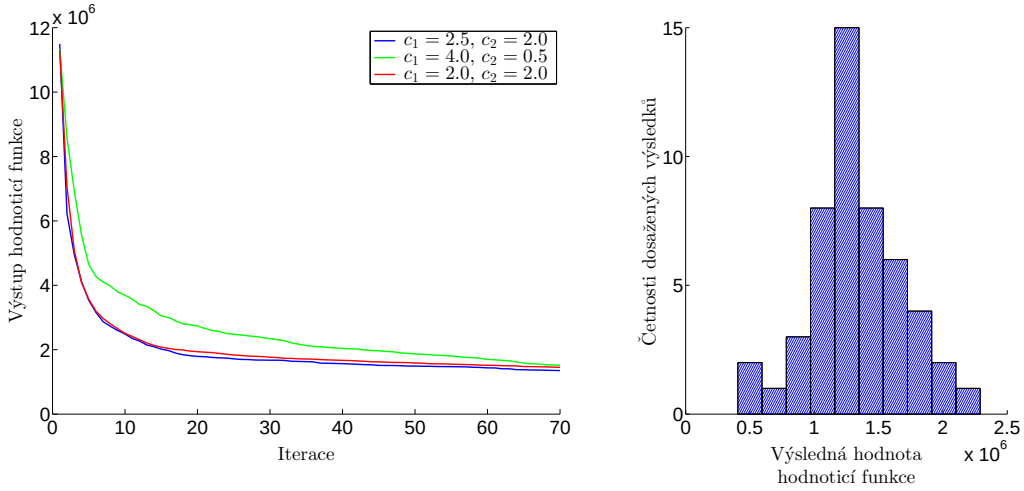
Ladění parametrů bude probíhat na základě průměrování výsledků následující konfigurace algoritmu:

počet opakování celého algoritmu	50
počet iterací PSO algoritmu	70
počet PSO částic (formací bezrozměrných kvadrokoptér)	20
počet kvadrokoptér v každé formaci	6
maximální povolená letová výška	515
minimální povolená letová výška	115

Omezení letové výšky jsou uvedena v jednotkách odpovídajících rozměrům mapy simulačního prostředí (viz např. obrázek 6b). Skutečné rozměry mapy v rovině $\{\mathbf{x}, \mathbf{y}\}$ jsou přibližně 175 $m \times 175$ m. Minimální letová výška udává hladinu, ve které snímky pořízené kamerou obsahují optimální objem dat, a maximální letovou výšku omezuje dostup kvadrokoptér.

5.2.1 Varianta s detekcí překážek

Samotná základní varianta PSO algoritmu není v úloze příliš využitelná kvůli tomu, že nerespektuje výskyt bezletových zón. Ladění parametrů c_1 a c_2 bude tedy probíhat rovnou ve variantě s detekcí překážek. Graf v obrázku 12a zachycuje vývoj hodnoticí funkce během jednotlivých iterací a graf v obrázku 12b znázorňuje kolikrát bylo dosaženo jakého výsledku při nejlepší kombinaci parametrů. Parametry c_1 a c_2 jsou voleny s krokem ± 0.5 kolem doporučeného nastavení $c_1 = 2$, $c_2 = 2$. Na obrázku 12a jsou zobrazeny průběhy hodnoticích funkcí pro doporučenou, nejlepší a nejhorší kombinaci konstant.



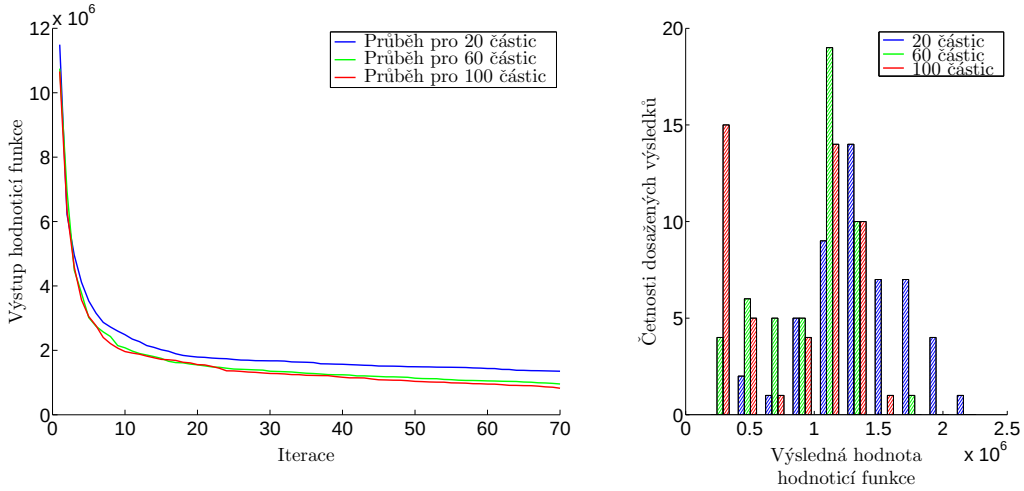
(a) vývoj hodnoticí funkce

(b) četnosti dosažených výsledků při nejlepší kombinaci parametrů

Obrázek 12: Hledání parametrů c_1 a c_2

Tyto parametry evidentně ovlivňují spíše rychlost konvergence algoritmu než výslednou hodnotu, neboť výsledné hodnoty se pro různé kombinace parametrů zásadně neliší.

Pro porovnání je na obrázku 13 zachycen průběh algoritmu pro třikrát a pětkrát zvětšený počet částic a parametry $c_1 = 2.5$, $c_2 = 2$. Je evidentní, že s rostoucím počtem jedinců algoritmu se zlepšují nalezená řešení. Daní za lepší nalezené řešení je doba běhu algoritmu, která se zvyšuje úměrně počtu částic.



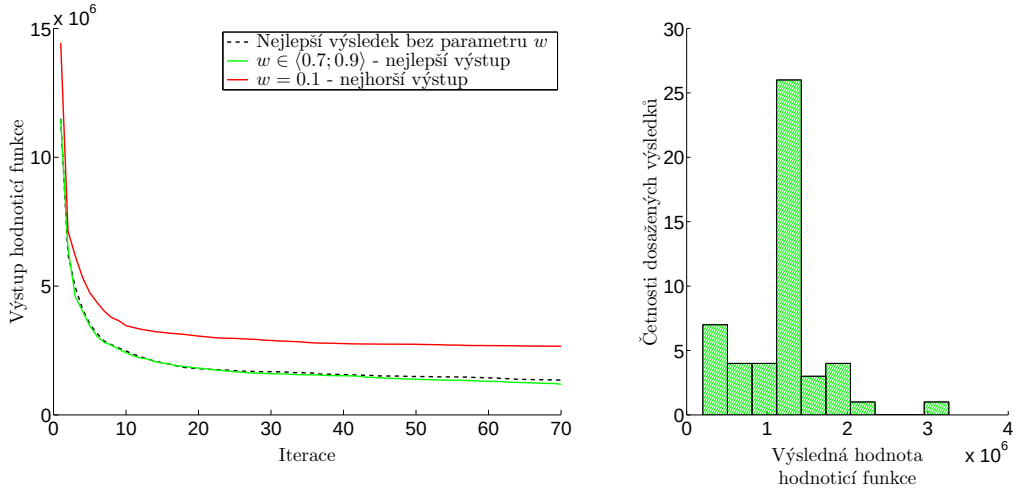
(a) vývoj hodnoticí funkce

(b) četnosti dosažených výsledků pro různé počty částic

Obrázek 13: Průběh algoritmu pro různé počty částic

5.2.2 Rozšíření PSO o použití setrvačnosti

Hodnoty setrvačnosti omezující vliv rychlosti z minulé iterace jsou pro testování voleny z intervalu $\langle 0.1; 1.0 \rangle$ s krokem 0.1. Dále jsou použity kombinace těchto hodnot pro testování lineárně se snižující hodnoty w s rostoucím číslem iterace. Výstup této varianty algoritmu znázorňuje obrázek 14.



(a) vývoj hodnoticí funkce

(b) četnosti dosažených výsledků při nejlepším nastavení parametru

Obrázek 14: Hledání parametru w

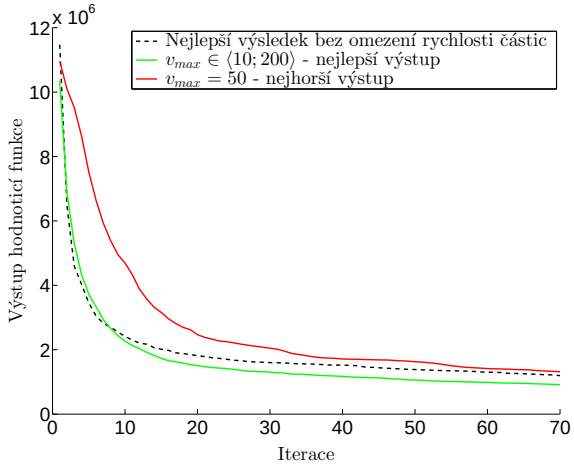
S ohledem na výsledky ladění parametru w lze konstatovat, že není dobré výrazně omezovat vliv rychlosti částice z minulé iterace a pro další testování volíme lineárně klesající hodnotu setrvačnosti na intervalu $\langle 0.7; 0.9 \rangle$.

5.2.3 Rozšíření PSO o omezení rychlosti částic

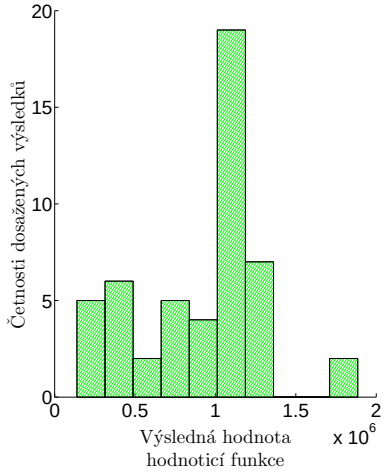
Výsledek testování PSO algoritmu při různých hodnotách maximální povolené rychlosti částic zachycují grafy na obrázku 15. Tyto hodnoty byly voleny v intervalu $\langle 50; 500 \rangle$ pro konstantní i klesající mezní rychlost. Podle výsledků bude v dalších simulacích maximální povolená rychlost částice omezena lineárně klesající hodnotou na intervalu $\langle 10; 200 \rangle$. S tímto omezením je dosaženo nejmenší průměrné hodnoty hodnoticí funkce.

5.2.4 Varianta respektující rozměry částice

Uvážíme-li rozměry mapy, minimální letovou výšku, výchozí minimální odstup $r_o = 70$ (v jednotkách mapy), zorný úhel kamery a vztah (31), dojdeme k závěru, že bude výhodné zvolit minimální odstup cílů na základě velikosti snímané plochy z minimální, resp. optimální letové výšky. Tak bude minimální odstup $r_o = 115$, tedy větší než velikost výchozího minimálního odstupu $r_o = 70$.

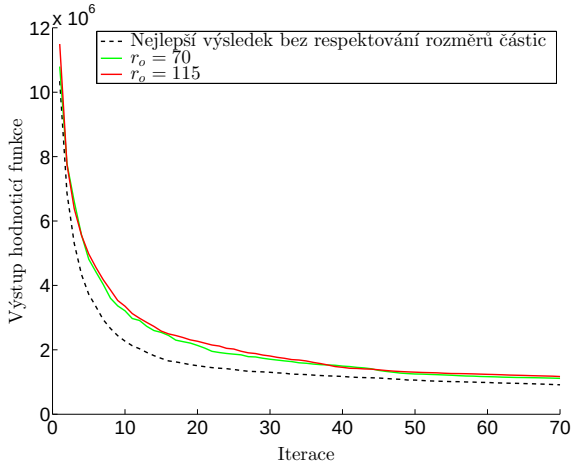


(a) vývoj hodnoticí funkce

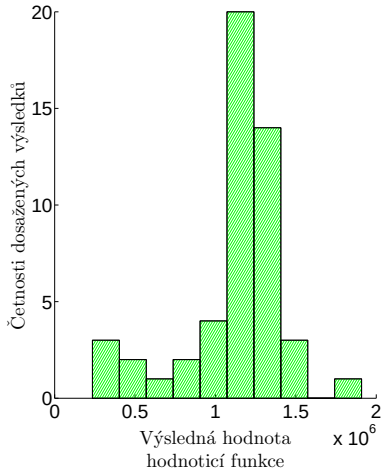


(b) četnosti dosažených výsledků při nejlepším nastavení parametru

Obrázek 15: Omezení rychlosti částic



(a) vývoj hodnoticí funkce

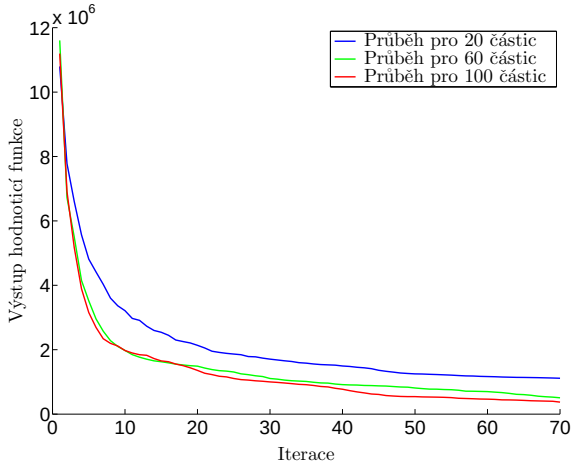


(b) četnosti dosažených výsledků při nejlepším nastavení parametru

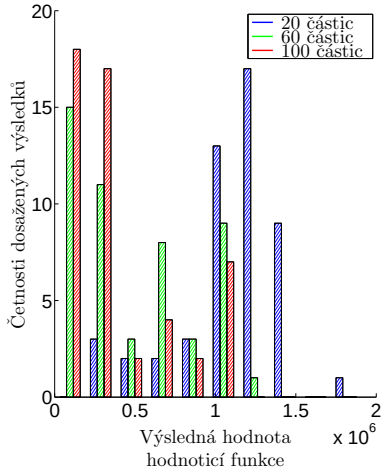
Obrázek 16: Dodržení minimálního odstupu

Z grafů na obrázku 16 je evidentní, že malý rozdíl výchozího odstupu a odstupu r_o založeného na velikosti snímané plochy nehraje významnou roli. Je zde ale patrné zhoršení průměrného výsledku díky částicím reálných rozměrů.

V grafu na obrázku 17a jsou znázorněny vývoje hodnoticích funkcí pro různé počty částic algoritmu, který nyní zahrnuje veškerá výše uvedená omezení. Opět je patrné, že při rostoucím počtu částic se zlepšují nalezená řešení.



(a) vývoj hodnoticí funkce



(b) četnosti dosažených výsledků pro různé počty částic

Obrázek 17: Průběh algoritmu s omezeními pro různé počty částic

Na základě provedených testování nastavíme laditelným parametrům pro další simulace následující hodnoty:

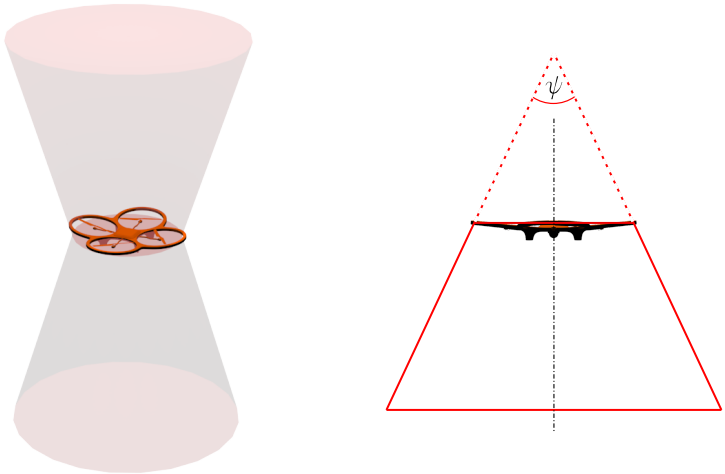
parametr c_1	2.5
parametr c_1	2.0
konstanta setrvačnosti w	lineárně klesající na intervalu $\langle 0.7; 0.9 \rangle$
omezení rychlosti částic v_{max}	lineárně klesající na intervalu $\langle 10; 200 \rangle$
minimální odstup cílů r_o	70

6 Omezení daná pohybem ve formaci

Při pohybu více robotů ve skupině musí být zajištěno, že každý jedinec bude předcházet kolizím s ostatními členy formace. Dalším omezením uvažovaným v této práci je požadavek relativní lokalizace členů formace. Ten definuje nutný počet sousedních robotů každého jedince, které se od něho nacházejí v menší než maximální povolené vzájemné vzdálenosti.

6.1 Kolize

Kolem použitých robotů je vytvořeno kolizní pásmo r_{min}, které nesmí sousední roboty narušit. Na základě kolizního pásma kvadrokoptér a ochranného pásma, na jehož délce je každý z robotů schopen z maximální rychlosti zastavit, stanovíme kolizní odstup r_{col}, který mezi sebou musí sousední roboty udržovat. V případě použití kvadrokoptér je třeba navíc zohlednit aerodynamické účinky vrtulí rotorů. Vrtule vytvářejí proud vzduchu, který negativně ovlivňuje ostatní členy formace. Tento proud vzduchu, znázorněný na obrázku 18, lze přibližně aproximovat tvarem přesýpacích hodin (dva komolé kužely zrcadlené podle horizontální osy tvořené rámem kvadrokoptéry), jak je uvedeno v bakalářské práci [13]. Vrcholový úhel komolého kuželu budeme pro další postup uvažovat $\psi = \frac{\pi}{6}$.



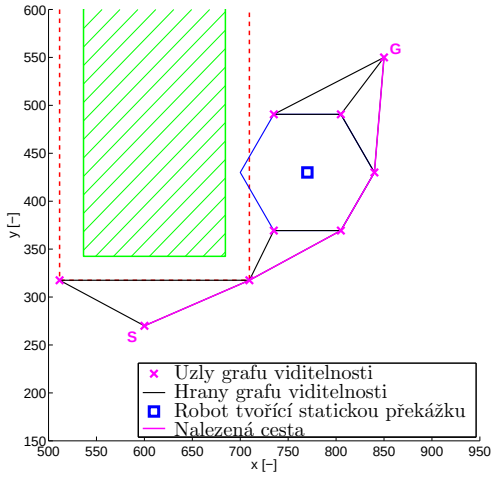
Obrázek 18: Znázornění proudů vzduchu (3D model [4])

Mechanismus předcházení kolizím musí zajistit, aby se dvě kvadrokoptéry nedostaly v rovině $\{\mathbf{x}, \mathbf{y}\}$ do bližší vzdálenosti než r_{min} a zároveň aby si vzájemně nenarušovaly prostor vymezený zónami proudění vzduchu. Také s ohledem na dále popsaný požadavek relativní lokalizace byl zvolen postup, kdy se všechny kvadrokoptéry pohybují v jedné letové hladině a aktuální výšku mění až poté, co dosáhnou pozic cílů v rovině $\{\mathbf{x}, \mathbf{y}\}$. Při závěrečném sestupu nebo stoupání k cíli

pokračuje robot tak dlouho, dokud nedosáhne cíle, nebo není indikována kolize s předpokládaným proudem vzduchu od vrtulí jiného stroje.

Při implementaci mechanismu předcházení kolizím se projevuje patrně největší nevýhoda grafu viditelnosti popsaného v podkapitole 4.3. Graf viditelnosti sice vede všechny kvadrokoptéry nejkratší možnou cestou k cíli, ale zároveň vytváří na trasách jednotlivých členů roje společné uzly, které musí všechny roboty navštívit. Zejména v těchto uzlech dochází při současném pohybu všech členů formace k častým kolizím, jejichž řešení je zvláště při účasti většího počtu členů velmi komplikované. Z toho důvodu je zvolen takový postup, že se v daném časovém okamžiku pohybuje pouze jedna z kvadrokoptér a ostatní tvoří statické překážky. Iterativní plánování drah všem kvadrokoptérám pomocí A* algoritmu po každém diskrétním časovém kroku by ale vedlo k nárůstu časové náročnosti programu. Proto tvoří všechny ostatní roboty na svých aktuálních pozicích statické překážky do té doby, než se i-tá kvadrokoptéra přemístí do svého iteračního cíle.

Kvadrokoptéry tvořící statické překážky jsou v rovině $\{\mathbf{x}, \mathbf{y}\}$ nahrazeny pravidelným šestiúhelníkem, jemuž vepsaná kružnice má poloměr nejméně r_{min}. Takovýto šestiúhelník se při plánování cest k cíli chová jako běžná bezletová zóna s ochranným pásmem, jak je znázorněno na obrázku 19.



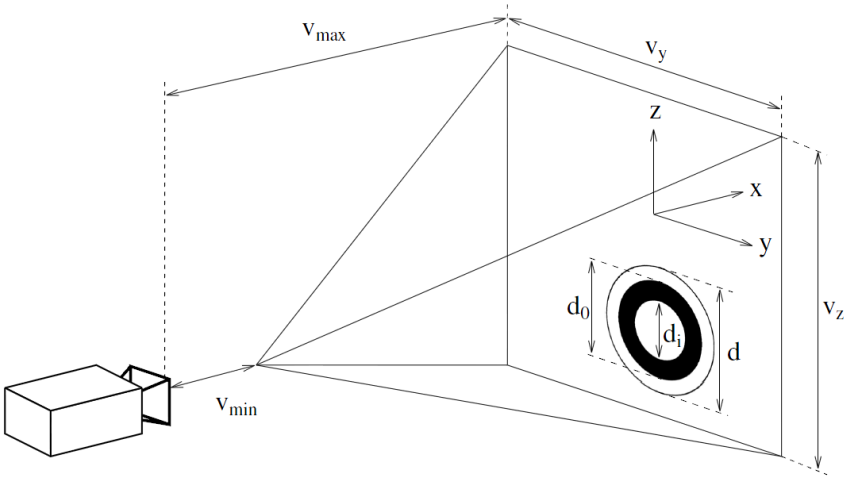
Obrázek 19: Řešení kolizí

6.2 Relativní lokalizace

Pohyb formace robotů může dále ovlivňovat požadavek relativní lokalizace, který vyžaduje udržení maximální vzájemné vzdálenosti v_{max} mezi členy. Všem členům formace je předem stanoven minimální počet sousedů, se kterými musí udržovat vzájemnou vzdálenost menší než v_{max}. Značení je vztaženo k obrázku 20.

Způsob realizace relativní vizuální lokalizace popsaný v literatuře [5] je založen na snímání černého a bílého kruhového symbolu na bílém pozadí (viz obrázek 20). Stejný symbol je umístěn na každém z členů formace. V této práci uvažujeme

osazení všech robotů více kamerami a větším počtem zmíněných symbolů tak, aby byl z každého směru viditelný alespoň jeden. Větší počet kamer společně s těmito symboly zajistí, že nezáleží na směru natočení robotu vůči ostatním členům formace. Na základě dat z kamery je možné získat informaci o vzájemné poloze robotů.

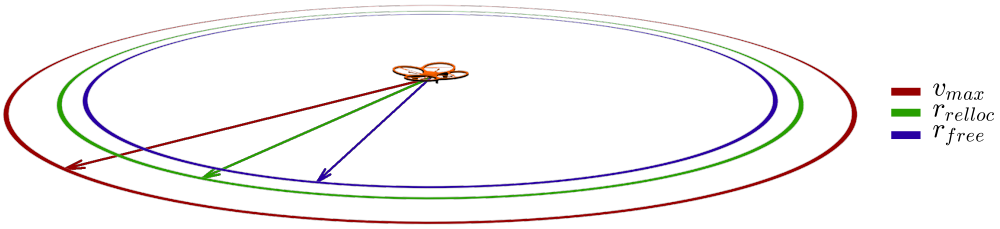


Obrázek 20: Princip relativní lokalizace [5]

Maximální povolená vzdálenost v_{max} mezi členy formace odpovídá maximální vzdálenosti, na kterou lze kamerou detekovat kruhový symbol. Tato vzdálenost vychází z ohniskové vzdálenosti použité kamery, rozlišení pořízeného snímku, velikosti symbolu d_0 a jeho natočení podél os $\mathbf{y}$ a $\mathbf{z}$ (dle obrázku). Obdobně je definována i minimální povolená vzdálenost v_{min}. Minimální povolenou vzdálenost v_{min} neuvažujeme, neboť je menší než velikost kolizního odstupu r_{col} zmíněného dříve.

Implementace omezení plynoucího z požadavku relativní vizuální lokalizace je v principu velmi jednoduchá. Jelikož kamera má omezený zorný úhel, je výhodné, aby se všechny kvadrokoptéry pohybovaly v jedné letové hladině. Tím odpadá nutnost zjišťovat v každém časovém kroku, zda se roboty stále nacházejí v zorných polích sousedních kamer. Stačí pouze opakovaně zjišťovat vzájemnou vzdálenost v rovině $\{\mathbf{x}, \mathbf{y}\}$ mezi sousedícími členy formace.

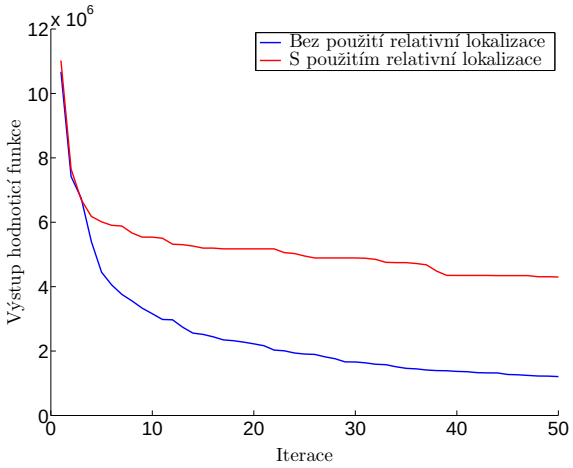
Předpokládejme, že relativní lokalizace vyžaduje, aby byl každý člen formace v neustálém vizuálním kontaktu s alespoň jedním dalším členem. Tedy, aby každá kvadrokoptéra měla souseda ve vzdálenosti menší než v_{max}. Kolem každého robotu je vytyčena kruhová zóna o poloměru r_{relloc} znázorněná na obrázku 21. Tento poloměr získáme odečtením vzdálenosti nutné k zastavení robotu od hodnoty v_{max}. Pokud je porušena podmínka dodržení vzdálenosti (překročena vzájemná vzdálenost r_{relloc}), robot, který svým pohybem tuto podmínku porušil, se vrátí na poslední známou pozici vyhovující podmínce. Pokračovat ve svém pohybu může teprve tehdy, až se vzdálenost sníží pod hodnotu danou poloměrem $r_{free} < r_{relloc}$. Tím je zabráněno kmitání kolem kružnice o poloměru r_{relloc}.



Obrázek 21: Omezení relativní lokalizace

V okamžiku, kdy všechny roboty dosáhnou svých iteračních cílů v rovině $\{\mathbf{x}, \mathbf{y}\}$, nebo jsou zablokovány podmínkou relativní lokalizace, jsou souřadnice $[x; y]$ jejich cílů určených PSO algoritmem (a tedy i jednotlivé PSO částice) upraveny podle aktuálních pozic. Požadované souřadnice z se snaží každá kvadrokoptéra dosáhnout změnou letové výšky. Kvadrokoptéra mění svou výšku, dokud není cíle dosaženo a dokud se nachází v zorném poli kamery sousedního stroje.

Vliv omezení relativní lokalizace na dříve získané výsledky ukazuje obrázek 22.



Obrázek 22: Vliv relativní lokalizace na průběh algoritmu

Jelikož požadavek relativní lokalizace zmenšuje aktuální operační prostor jednotlivých kvadrokoptér, zhorší se i průměrný dosažený výsledek odpovídající kvalitě pokrytí mapy a rychlost konvergence algoritmu. Vliv tohoto omezení klesá s rostoucím počtem členů formace, nebo se zvětšujícím se poloměrem v_{max}.

7 Experimenty

Funkčnost algoritmu pro hledání optimálního rozložení roje kvadrokoptér bude ověřena experimenty, které demonstrují vliv požadavku relativní lokalizace na nalezený výsledek. Jelikož výpočetní doba programu narůstá se složitostí mapy, počtem použitých částic PSO algoritmu, resp. virtuálních formací kvadrokoptér, a počtem nasazených robotů, budou statistické experimenty provedeny v jednodušších mapách, které pro nalezení relativně dobrého výsledku nevyžadují velký počet částic PSO algoritmu.

Hledání optimálního rozložení formace probíhá pro následující konfiguraci algoritmu (jednotky hodnot v tabulce odpovídají jednotkám mapy):

počet PSO částic (virtuálních formací kvadrokoptér)	15
počet kvadrokoptér v každé formaci	3
maximální povolená letová výška	800
minimální povolená letová výška	200
kolizní odstup r_{min}	80
maximální povolená vzájemná vzdálenost v_{max}	300

Dále je stanovena podmínka, že každý robot musí během pohybu k cílové pozici udržet lokalizační vzdálenost $< v_{max}$ s alespoň jedním dalším členem formace.

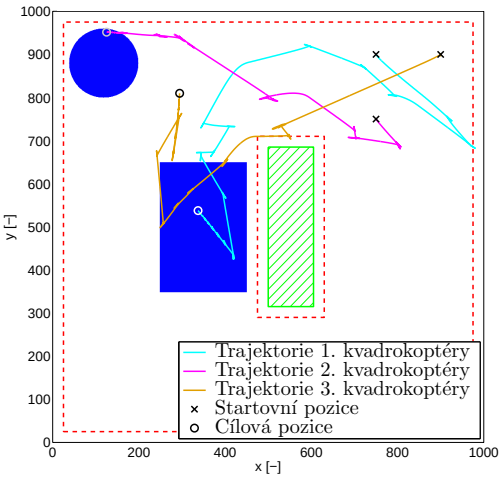
7.1 Jednoduchá mapa

Na obrázku 23 je znázorněn průběh optimalizace rozložení formace kvadrokoptér pro relativně jednoduchý případ s jednou bezletovou zónou. Mapu prostoru s vyznačenými lokalitami určenými k monitorování a s výsledným rozložením formace popisuje obrázek 23a. Na této mapě záleží kvalita nalezeného výsledku především na počátečním rozmístění všech částic PSO algoritmu (formací kvadrokoptér), které je náhodné.

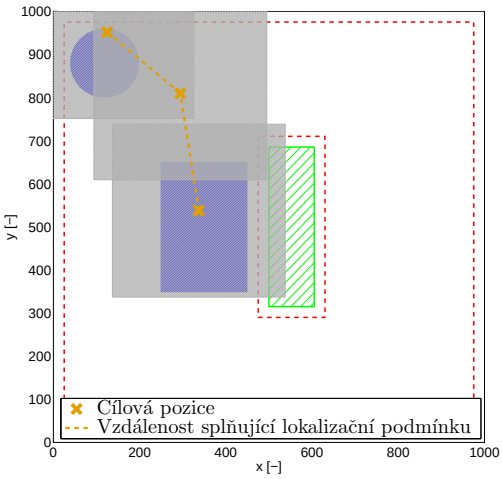
7.2 Speciální případ

Druhý experiment zachycený na obrázku 24 demonstruje vliv omezení relativní lokalizace. Samotný PSO algoritmus je schopen nalézt v zobrazené mapě se složitější bezletovou zónou optimální řešení, kdy jsou všechny oblasti určené k monitorování zcela pokryty. Velikost a tvar bezletové zóny ale kvadrokoptérám znemožňuje dosažení určených pozic při současném požadavku maximální povolené vzdálenosti.

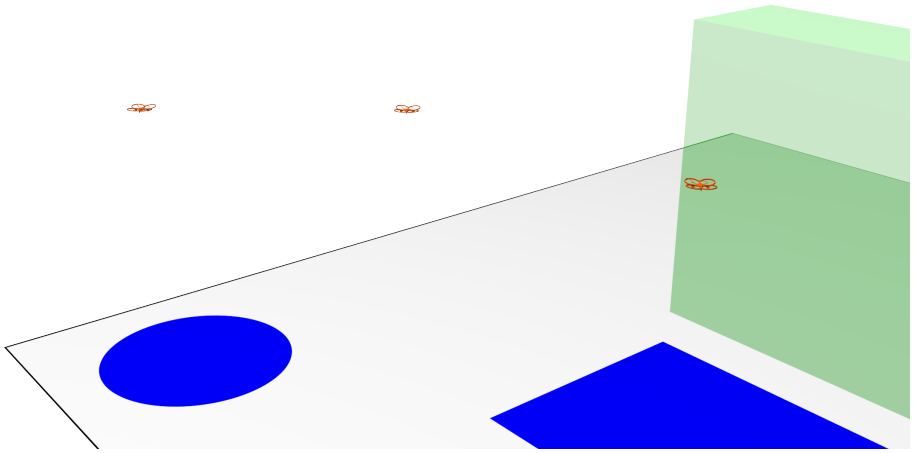
Hledání optimálního rozložení formace probíhá pro stejnou konfiguraci algoritmu jako v předchozím případě. Pouze je změněna hodnota maximální povolené vzájemné vzdálenosti na 200 jednotek mapy.



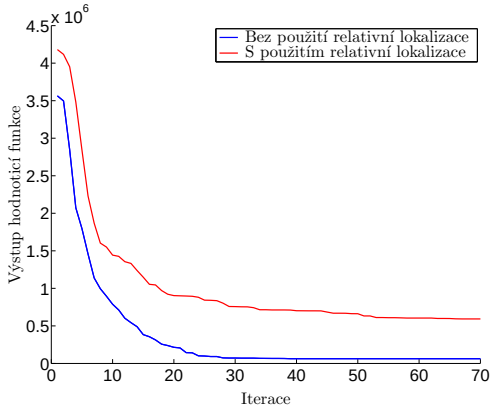
(a) naplánované trajektorie



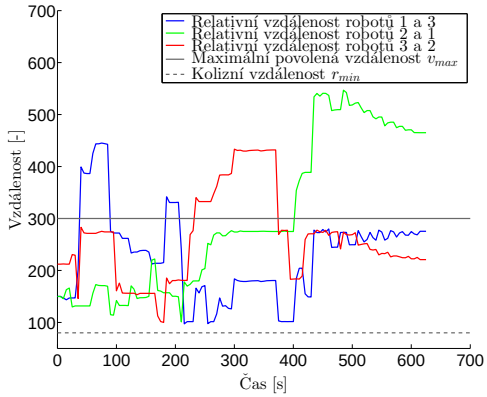
(b) výsledné pokrytí lokalit



(c) výsledné rozmístění kvadrokoptér

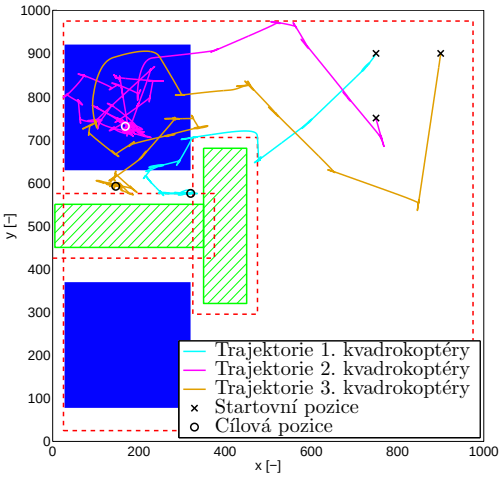


(d) průměrný vývoj hodnotící funkce pro 20 opakování optimalizačního algoritmu

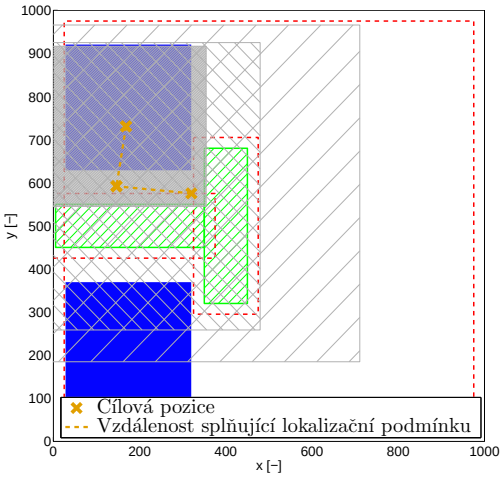


(e) relativní vzdálenosti robotů

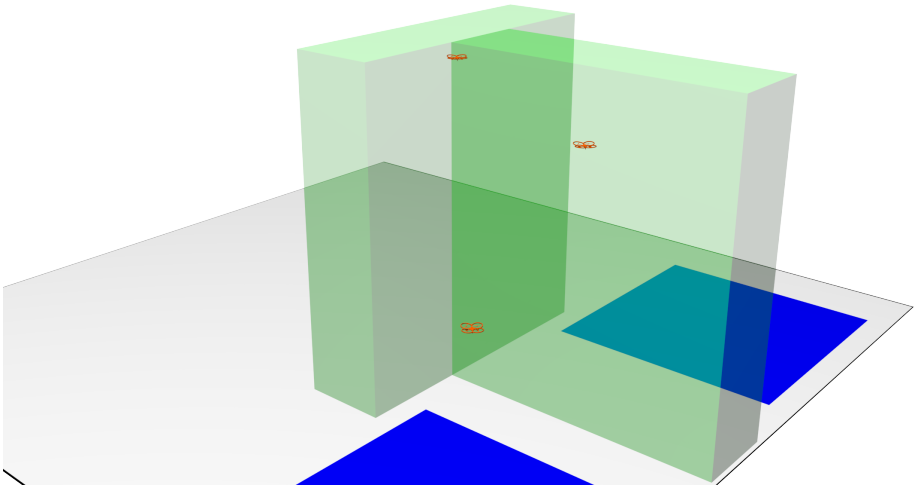
Obrázek 23: Experiment 1



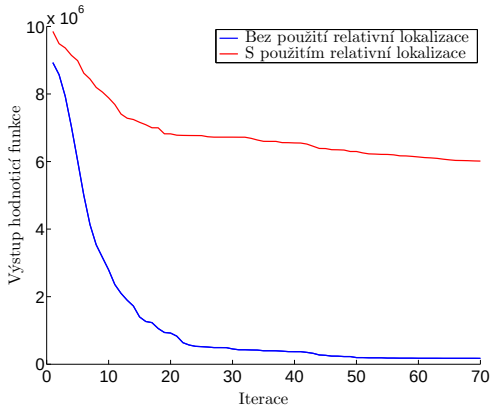
(a) naplánované trajektorie



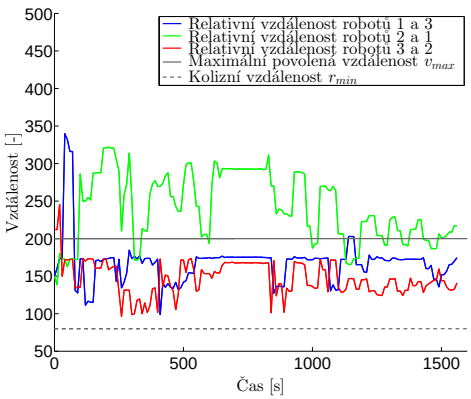
(b) výsledné pokrytí lokalit



(c) výsledné rozmístění kvadrokoptér



(d) průměrný vývoj hodnoticí funkce pro 20 opakování optimalizačního algoritmu



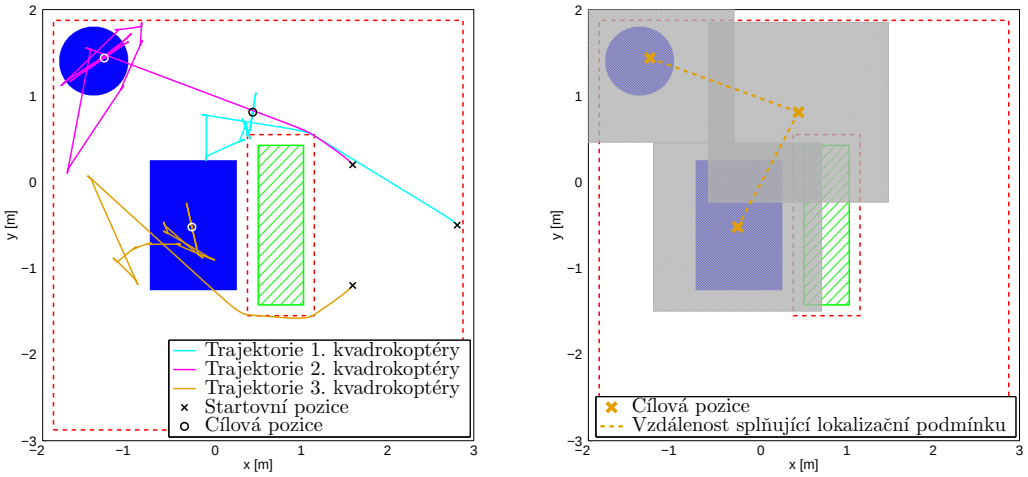
(e) relativní vzdálenosti robotů

Obrázek 24: Experiment 2

7.3 Reálné nasazení

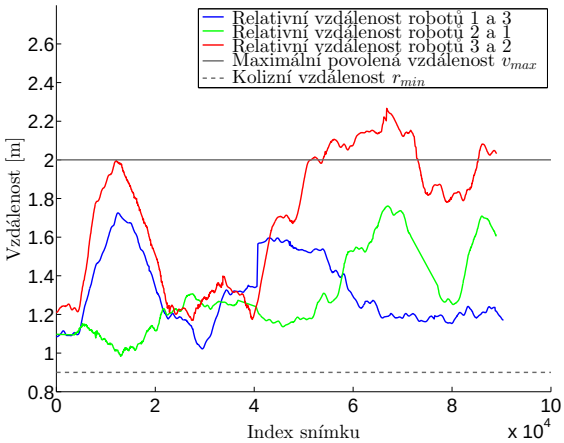
Ověření funkčnosti metody v praxi [20] proběhlo na mapě s jednoduchou bezletovou zónou. Konfigurace algoritmu byla nastavena na hodnoty odpovídající rozměrům místa nasazení. Průběh a výsledky experimentu popisuje obrázek 25.

K ověření funkčnosti mechanismu řešení kolizí a omezení relativní lokalizace jsou v průběhu experimentu vzájemné vzdálenosti mezi kvadrokoptérami počítány také na základě dat z globálního lokalizačního systému (VICON), který snímá celý pracovní prostor a umožňuje sledovat pohyb kvadrokoptér. Na obrázku 25c jsou znázorněny vzájemné vzdálenosti získané globálním kamerovým systémem během pohybu formace. Byla ověřena funkčnost implementovaných mechanismů, neboť vzdálenosti jednotlivých členů formace se pohybovaly v předepsaných mezích, což umožňuje stabilizovat roj kvadrokoptér na základě palubní relativní lokalizace.



(a) trajektorie snímané systémem VICON

(b) výsledné pokrytí lokalit



(c) relativní vzdálenosti kvadrokoptér podle systému VICON

Obrázek 25: Experiment s reálnými roboty

Z obrázku 25c je zřejmé, že během pohybu formace dojde k překročení maximální povolené vzdálenosti (a tím i k porušení lokalizační podmínky) mezi roboty 2 a 3. Tento fakt nemá vliv na další pohyb a lokalizaci robotů, neboť podmínkou je udržení lokalizační vzdálenosti alespoň s jedním dalším členem formace a zmíněné roboty stále udržují lokalizační vzdálenost s některým z dalších sousedů.

Obrázek 25b znázorňuje pokrytí oblastí určených k monitorování. K získání maximálního objemu dat z těchto oblastí jsou zřejmě dostačující snímky z kamer robotů 2 a 3. Robot 1 tak pouze tvoří prostředníka, pomocí kterého je během pohybu formace splněna podmínka relativní lokalizace, aby roboty 2 a 3 mohly zaujmout optimální pozice.

(a) experiment se třemi kvadrokoptérami

(b) snímky z palubních kamer pro relativní lokalizaci

Obrázek 26: Snímky experimentu s reálnými roboty

8 Závěr

Cílem této práce byl návrh, implementace a ověření funkčnosti algoritmu optimalizujícího tvar roje bezpilotních helikoptér pro potřeby robotického dohledu. Jako vstupní parametr byla algoritmu předána mapa oblasti s vyznačenými lokalitami, které je třeba monitorovat. Mapa také obsahuje informaci o případném výskytu bezletových zón (překážek) v pracovním prostoru. Úkolem algoritmu bylo nalézt takové rozmístění helikoptér, které maximalizuje objem dat získaný senzory použitými pro zadanou úlohu robotického dohledu. Nejen výsledné rozmístění, ale také trajektorie umožňující tohoto rozmístění dosáhnout musí splňovat kinematická omezení pohybu helikoptér. Také musí respektovat omezení plynoucí z pohybu ve formaci o více členech. Zvláště je kladen důraz na předcházení kolizím a udržení vzájemné lokalizační vzdálenosti.

První část práce je věnována seznámení se s problematikou a s prostředky, které budou použity k jejímu řešení. Konkrétně se jedná o popis použitých helikoptér a o způsob tvorby mapy pracovního prostoru pro praktické nasazení robotů. Dále byl popsán průběh optimalizace tvaru roje, metoda plánování pohybu v prostoru obsahujícím překážky a způsob řešení omezení plynoucích z pohybu ve formaci. Nakonec byla funkčnost použitých postupů ověřena několika experimenty.

Optimalizace tvaru roje helikoptér se provádí pomocí PSO (Particle Swarm Optimization) algoritmu. V práci je popsáno několik nově vytvořených modifikací pro použití algoritmu v této konkrétní úloze. Účelem těchto modifikací je respektovat případný výskyt bezletových zón (překážek) v pracovním prostoru a omezení daná nasazením reálných helikoptér, které jsou vzájemně stabilizované pomocí relativní vizuální lokalizace. Dále byly nalezeny hodnoty parametrů algoritmu vhodné pro zadanou úlohu.

V uvedených experimentech je ověřena funkčnost celého optimalizačního procesu. Mimo jiné je zde testována jeho schopnost vést helikoptéry k jejich cílům mimo bezletové zóny. Experimenty dále ukazují vliv omezení daného relativní lokalizací, neboli udržení maximální povolené vzájemné vzdálenosti mezi členy formace. Toto omezení výrazně zmenšuje prostor, ve kterém mohou jednotlivé helikoptéry v daný okamžik operovat, čímž se zhoršuje rychlost konvergence algoritmu. Je ale zajištěno, že proces optimalizace probíhá v prostoru řešení, která jsou realizovatelná skupinou relativně lokalizovaných helikoptér.

Experimentem s reálnými helikoptérami byla ověřena možnost praktického nasazení použitého optimalizačního algoritmu.

Reference

[1] HiSystems GmbH. http://www.mikrocontroller.com/, [Online; accessed November 4, 2013].

[2] Oscar Liang. http://blog.oscarliang.net/, [Online; accessed November 4, 2013].

[3] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric Tracking Control of a Quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[4] Pischixot. IR Quadrocopter 410. http://www.turbosquid.com/3d-models/free-ir-quadrocopter-410-3d-model/615814, [Online; accessed November 4, 2013].

[5] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *ICRA2013: Proceedings of 2013 IEEE International Conference on Robotics and Automation*, pages 985–990, 2013.

[6] FIRA. http://www.fira.net/, [Online; accessed November 30, 2013].

[7] RoboCup. http://www.robocup.org/, [Online; accessed November 30, 2013].

[8] Mohammad Al-khawaldah and Andreas Nüchter. Multi-Robot Exploration and Mapping with a rotating 3D Scanner. Technical report, Jacobs University Bremen gGmbH, 2012.

[9] J. R. Martínez-de-Dios, Luis Merino, Aníbal Ollero, Luis M. Ribeiro, and Xavier Viegas. Multi-UAV Experiments: Application to Forest Fires. In *Aníbal Ollero and Iván Maza (Eds.): Multiple Heterogeneous Unmanned Aerial Vehicles*, pages 207–228. Springer Berlin Heidelberg, 2007.

[10] Sonia Waharte and Niki Trigoni. Supporting Search and Rescue Operations with UAVs. In *2010 International Conference on Emerging Security Technologies (EST)*, pages 142–147, 2010.

[11] Volker Grabe, Heinrich H. Bülthoff, and Paolo Robuffo Giordano. Robust Optical-Flow Based Self-Motion Estimation for a Quadrotor UAV. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2012.

[12] James Kennedy and Russel Eberhart. Particle Swarm Optimization. In *Proceedings International Conference on Neural Networks IEEE*, pages 1942–1948, 1995.

[13] Vojtěch Pavlík. Algoritmy skupinové imteligence pro využití v multi-robotických úlohách, bachelor thesis. ČVUT Praha, 2012.

[14] Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. In *2011 IEEE International Conference on Robotics and Automation*, pages 763–770, 2011.

[15] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. *International Journal of Robotics Research*, 2012.

[16] Christopher Triola. Special Orthogonal Groups and Rotations. Submitted in partial fulfillment of the requirements for Honors in Mathematics at the University of Mary Washington, http://doctorh.umwblogs.org/files/2010/10/honors_triola.pdf, 2009. [Online; accessed November 30, 2013].

[17] DigitalGlobe, Geo-Basis-DE/BKG, Google Maps. Leutershausen, Německo, [Online; accessed November 9, 2013].

[18] Martin Saska. Plánování pro robotický fotbal, diploma thesis. ČVUT Praha, 2005.

[19] Michal Pěchouček and Milan Rollo. Informované metody prohledávání stavového prostoru, lecture, http://cw.felk.cvut.cz/doku.php/courses/a3b33kui. [Online; accessed November 11, 2013].

[20] Video record of the experiment. www.youtube.com/watch?feature=player_embedded&v=4S0vzYgFh8M/, [Online; accessed December 20, 2013].

Obsah přiloženého CD

Název adresáře	Popis obsahu
bp	bakalářská práce ve formátu .pdf
sources	zdrojové kódy programu
outputs	výstupní soubory programu
images	použité obrázky