SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Ing. Jaromír Stanko
Autoreferát dizertačnej práce

MODERNÉ METÓDY PLÁNOVANIA V ČIASTOČNE ZNÁMOM PROSTREDÍ PRE AUTONÓMNE BEZPILOTNÉ LIETAJÚCE PROSTRIEDKY

na získanie akademického titulu
„doktor“ („philosophiae doctor“, v skratke „PhD.“)

v doktorandskom študijnom programe:	Robotika a Kybernetika
v študijnom odbore:	9.2.7 kybernetika
Forma štúdia:	denná forma

Bratislava, 2024

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Jaromír Stanko
Autoreferát dizertačnej práce

MODERNÉ METÓDY PLÁNOVANIA V ČIASTOČNE ZNÁMOM PROSTREDÍ PRE AUTONÓMNE BEZPILOTNÉ LIETAJÚCE PROSTRIEDKY

na získanie akademického titulu
„doktor“ („philosophiae doctor“, v skratke „PhD.“)

v doktorandskom študijnom programe:	Robotika a Kybernetika
v študijnom odbore:	9.2.7 kybernetika
Forma štúdia:	denná forma

Bratislava, 2024

Dizertačná práca bola vypracovaná na:
Ústav robotiky a kybernetiky,
Fakulty elektrotechniky a informatiky,
Slovenskej technickej univerzity v Bratislave

Predkladateľ: Ing. Jaromír Stanko
Slovenská technická univerzita v Bratislave,
Fakulta elektrotechniky a informatiky,
Ústav robotiky a kybernetiky

Školiteľ: Ing. Jozef Rodina, PhD.
Slovenská technická univerzita v Bratislave,
Fakulta elektrotechniky a informatiky,
Ústav robotiky a kybernetiky

Oponent: Ing. Ivana Budinská, PhD.
Slovenská Akadémia Vied,
Ústav informatiky SAV

Oponent: doc. Ing. Ján Semjon, PhD.
Technická univerzita v Košiciach,
Strojnícka fakulta,
Katedra výrobnej techniky a robotiky

Autoreferát bol rozoslaný: .....

Obhajoba dizertačnej práce sa bude konať dňa 22.8.2024
o 10:30h na ÚRK FEI STU - Ilkovičova 3, 841 04 Bratislava, v miestnosti D424.

prof. Ing. Vladimír Kutiš, PhD.
dekan FEI STU v Bratislave

Obsah

Úvod

Bezpilotné lietajúce zariadenia (UAV) si vďaka rýchlemu vývoju informačných technológii a klesajúcim nákladom výrobu rozširujú svoje uplatnenie v rôznych oblastiach civilného využitia (poľnohospodárstvo, umelecká tvorba (video, fotografia), lesníctvo, záchranné zložky, geodézia ...). Donedávna sa úlohy vykonávane UAV zväčša odohrávali v exteriéry pričom nebol kladený dôraz na autonómiu ich vykonávania. Vychádzalo to z limitov dostupných technológií (dostupný výpočtový výkon voči potrebnej energii) ale aj povahy samotnej úlohy, kedy bolo resp. je postačujúce, ak je UAV ovládané operátorom. S príchodom nových technológii ako sú malé počítače s vysokým výpočtovým výkonom a zároveň s nízkou spotrebou energie, kompaktné Lidary, RGBD a SLAM kamery, atď. sa UAV otvorili možnosti ich využiteľnosti práve v tých oblastiach, kde je potrebná určitá miera autonómie počas vykonávania zadanej úlohy.

Časť úloh vykonávaných s využitím UAV sa môže odohrávať aj v interiéry, pričom pohyb v interiéry so sebou prináša dodatočné obmedzenia. Vnútorný priestor je obmedzený svojou veľkosťou, môže byť členitý a obsahovať úzke prechody. To limituje manévrovací priestor UAV pre obchádzania prekážky a taktiež zvyšuje nárok na rýchlosť vyhodnocovania potenciálnej kolízie a preplánovania potenciálne kolíznej cesty na bezkolíznu. Okrem iného v interiéry existuje väčšie riziko vzniku škody na UAV alebo jeho okolí (ľudia, materiál, iné zariadenia ...), nedostupnosti GNSS ...

Plánovanie cesty je dôležitou súčasťou riadiaceho systému autonómneho UAV. Cieľom takéhoto plánovania je vo všeobecnosti nájsť takú postupnosť platných konfigurácii (stavov) systému, ktorá dostane riadený systém z jeho počiatočnej konfigurácie do cieľovej konfigurácie bez kolízie riadeného systému s prostredím, v ktorom sa pohybuje. Vzorkovacie plánovacie algoritmy (sampling-based) sú jedným z existujúcich prístupov k plánovaniu. Využívajú vzorkovanie konfiguračného priestoru, ktorý je týmto procesom rýchlo preskúmavaný a hľadajú možné prepojenia medzi vzorkami (konfiguráciami). Rýchlo-prehľadávajúce náhodné stromy (RRT) sú podskupinou vzorkovacích plánovacích algoritmov. Dokážu efektívne prehľadávať mnohorozmerný konfiguračný priestor a zohľadňovať rôzne obmedzenia (prekážky, kinematické, dynamické, …). Zo získaných vzoriek konfiguračného priestoru vytvoria graf so stromovou štruktúrou. Vďaka ich všestrannosti a širokému rozšíreniu vzniklo množstvo modifikácii pôvodného RRT algoritmu, ktoré sa sústreďujú na zlepšenie plánovania vzhľadom na zvolené kritérium (kvalita výslednej cesty, rýchlosť nejdenia prvej cesty, rýchlosť konvergencie k

optimálnemu riešeniu, …), riešenie špecifickej skupiny úloh, výpočtovú náročnosť a pod. ... Zväčša sa využívajú na plánovanie globálnej cesty, ktorá je naplánovaná pred tým, ako sa robot začne pohybovať v prostredí.

Prostredie nemusí byť vždy vopred známe, resp. je známe len čiastočne v dôsledku neúplnej mapy prostredia, v čase meniaceho sa prostredia, prostredia s dynamickými prekážkami, ktorých stav nevieme predpovedať. Zmenou prostredia sa naplánovaná cesta môže rýchlo stať neplatnou a viesť ku kolízii. V takejto situácii je nutné cestu preplánovať. Začať plánovací proces od začiatku a tým zahodiť už existujúci graf nemusí byť ideálne. V niektorých prípadoch postačuje, ak je už existujúci graf vytvorený plánovacím algoritmom rekonštruovaný. Upravená cesta tak bude tvorená uzlami grafu, ktoré po rekonštrukcii nie sú ovplyvnené novou prekážkou. Existujú algoritmy vychádzajúce z RRT, ktoré sa venujú tejto problematike pričom využívajú rôzne prístupy k riešeniu tohto problému. Jedným z prínosom tejto práce je navrhnutie modifikácie RRT algoritmu, ktorý bude schopný rýchlo preplánovať cestu prerušenú prekážkou. Návrh algoritmu vychádza z už existujúceho algoritmu RRT^X, ktorý je upravený a zároveň rozšírený o funkcionality, ktoré dopomáhajú k rýchlemu preplánovaniu cesty.

Plánovací algoritmus si z časového hľadiska nemôže dovoliť brať priveľký ohľad na kvalitatívne kritéria preplánovanej cesty, ak má byť proces preplánovania ukončený čo najrýchlejšie. Pre optimalizáciu novej cesty je čas až po preplánovaní. Preto je ďalším prínosom práce rozšírenie navrhnutého plánovacieho algoritmu tak, že počas prechodu mobilného robota prostredím je aktuálna cesta priebežne vyhodnocovaná a optimalizovaná. Vďaka tomu môže byť čas vyhradený pre plánovanie prvotnej cesty skrátený a nová cesta, ktorá je výsledkom preplánovania, môže konvergovať k optimálnej ceste počas obchádzania prekážky.

Rýchlosť s akou je plánovací algoritmus schopný upraviť cestu, ktorá bola prerušená prekážkou, je kriticky dôležitým aspektom, ak chce byť takýto algoritmus využitý online počas prechodu mobilného robota prostredím. Ak by algoritmus nebol dostatočne rýchly, mobilný robot (uvažujúc UAV) by nemohol plynulo pokračovať v pohybe a musel by sa v určitom momente zastaviť a čakať na novú cestu. Okrem návrhu samotného algoritmu je preto dôležitá aj jeho implementácia a využitie dostupného výpočtového výkonu v najvyššej možnej miere. Paralelizácia je jedným z možných spôsobov dosiahnutia efektívneho využitia hardvéru, na ktorom je algoritmus spustený. Ďalším prínosom tejto práce je paralelizovanie navrhnutého plánovacieho algoritmu – upravenie vybraných častí algoritmu tak, aby ich bolo možné paralelizovať a zároveň upraviť implementáciu algoritmu tak, aby

procesor mohol paralelizovateľné časti algoritmu vykonávať paralelne vo viacerých vláknach.

Táto práca má za cieľ byť aj prakticky orientovaná. Preto si kladie za úlohu, aby bolo možné jej výsledok (navrhnutý algoritmus a jeho implementáciu) nasadiť na zariadení s obmedzeným výpočtovým výkonom osadeným na UAV a nie len na stolovom počítači, ktorý ho má v porovnaní s ním prebytok. Minipočítač osadený na dostupnom UAV má k dispozícii integrovaný grafický procesor s CUDA jadrami, ktorého využitie má potenciál na dodatočné paralelizovanie niektorých výpočtov vykonávaných počas procesu preplánovania a zníženie časovej náročnosti celého procesu. Preto je posledným spomenutým prínosom tejto práce nasadenie implementácie navrhnutého algoritmu na palubnom počítači osadenom na UAV, využitie GPU hardvérovej akcelerácie a vytvorenie ROS balíčka pre zjednodušenie nasadenia a distribúcie implementácie navrhovaného algoritmu.

Tézy dizertačnej práce

1. Návrh plánovacieho algoritmu založenom na rýchlo-prehľadávajúcich náhodných stromoch pre plánovanie cesty v čiastočne známom prostredí, ktorý bude schopný rýchlo upraviť cestu prerušenú neznámou prekážkou.
2. Priebežné vyhodnocovanie a optimalizácia naplánovanej cesty počas prechodu mobilného robota prostredím alebo obchádzania prekážky. Priebežné aktualizovanie cieľa plánovania počas prechodu mobilného robota prostredím.
3. Paralelizovanie plánovacieho algoritmu. Identifikácia častí algoritmu, ktoré je možné a zároveň vhodné vykonávať paralelne. Upraviť implementáciu algoritmu tak, aby procesor mohol paralelizovateľné časti algoritmu vykonávať paralelne vo viacerých vláknach.
4. Nasadenie navrhnutého algoritmu na palubnom počítači UAV, ktorý má obmedzený výpočtový výkon. Využitie hardvérovej akcelerácie dostupnej na palubnom počítači pre odľahčenie procesora a dodatočné zrýchlenie plánovacieho algoritmu.

1 Návrh algoritmu pre plánovanie cesty v čiastočne známom prostredí

Aplikácia autonómnych UAV v inšpekčných úlohách vnútorných priestorov ako sú výrobné haly alebo skladové priestory je jednou z hlavných motivácii tejto práce. Takéto prostredie môže byť z pohľadu pohybu v ňom výzvou pre UAV (členité prostredie, úzke prechody ...). Na druhú stranu je ale dobre definované vďaka dostupným mapovým podkladom a známom rozložení objektov v priestore, ktoré predstavujú potenciálnu prekážku pre UAV. Aj napriek tomu nie je takéto prostredie nemenné a z rôznych dôvodov sa môže stať, že sa v priestore nachádza objekt, ktorý nie je uvažovaný v dostupnej mape alebo známy objekt bol premiestnený. Cieľom tejto kapitoly je predstaviť navrhnutý plánovací algoritmus, ktorý je využiteľný pre danú aplikáciu. Rieši úlohu plánovania cesty pre UAV s využitím dostupnej globálnej mapy a zároveň v prípade hroziacej kolízie s vopred neznámou prekážkou dokáže počas prechodu UAV prostredím sledovanú cestu v krátkom čase preplánovať. UAV danú prekážku obíde po alternatívnej ceste a bude môcť bezkolízne pokračovať do cieľového bodu.

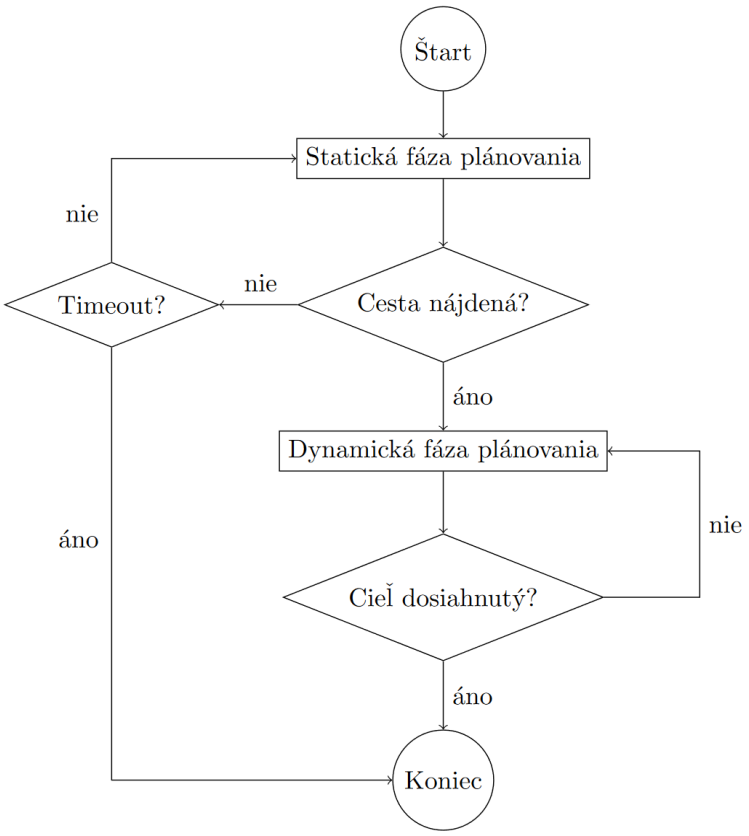
Navrhovaný a zároveň implementovaný algoritmus je modifikáciou rýchlo-prehľadávajúceho náhodného stromu a vychádza z algoritmu RRT^X [10]. Táto práca modifikuje pôvodný RRT^X algoritmus v dvoch smeroch. Prvým je rozšírenie o funkcionality inšpirované algoritmami z predošlej kapitoly a technikami, ktoré sú rozšírené nie len medzi vzorkovacími plánovacími algoritmami – informované vzorkovanie, priebežné optimalizovanie cesty počas prechodu UAV prostredím. Druhým smerom je úprava vhodných častí algoritmu tak, aby ich bolo možné paralelizovať. Dôraz je kladený na optimalizáciu výpočtového času najmä počas preplánovania cesty okolo prekážky.

Navrhnuté modifikácie sú hlavným prínosom tejto práce. Plánovací algoritmus navrhnutý touto prácou je rozšírený o obmedzenie maximálneho počtu uzlov, ktoré môže vytváraný graf so stromovou štruktúrou obsahovať a funkcionalitu, ktorá orezáva nadbytočné uzly. Modifikované je vzorkovanie konfiguračného priestoru počas plánovania celého procesu plánovania, kedy je so zvolenou pravdepodobnosťou vzorkovaný priestor v okolí najlepšej nájdenej cesty. Cieľom je urýchliť zlepšovanie aktuálne najlepšej nájdenej cesty, pokiaľ sa nenájde lepšia alternatíva cesta. Doplnená je možnosť zmeny cieľovej konfigurácie počas oboch fáz plánovania. Ďalšou modifikáciou je priebežná optimalizácia cesty počas prechodu robota prostredím overovaním možnosti prepojenia uzla stromu reprezentujúceho aktuálnu konfiguráciu robota so vzdialenejšími uzlami sledovanej cesty vo

vopred definovanom horizonte. Modifikovaný je prístup k preplánovaniu sledovanej cesty prerušenou vopred neznámou prekážkou, ktorý v navrhnutom plánovacom algoritme prebieha v dvoch krokoch. Navrhnutý algoritmus najprv vyskúša časovo nenáročné a rýchle prepojenie odstrihnutej časti sledovanej cesty s vytvoreným stromom a až potom pristúpi k výpočtovo náročnejšej rekonštrukcii časti stromu. Modifikovaný je samotný proces rekonštrukcie (prepojovania) stromu tak, aby sa rýchlo našlo nové prepojenie odstrihnutej časti cesty k stromu prepojovaním odstrihnutých uzlov stromu. Optimalizácii výpočtového času je venovaná samostatná kapitola, kde sú identifikované výpočtovo náročnejšie časti algoritmu.

1.1 Hlavná slučka plánovacieho algoritmu

Navrhnutý plánovací algoritmus rieši plánovanie globálnej cesty a jej preplánovanie respektíve opravu, ak bola zablokovaná prekážkou. Globálna cesta je plánovaná počas fázy, kedy sa UAV (alebo akýkoľvek iný mobilný robot) nepohybuje v priestore a čaká na plánovací algoritmus a výsledok jeho plánovania – statická fáza. Počas prechodu UAV prostredím po naplánovanej globálnej je táto cesta ďalej optimalizovaná a v prípade hroziacej kolízie je cesta upravená, preplánovaná – dynamická fáza. Hlavná slučka navrhnutého algoritmu je definovaná týmito dvomi fázami plánovania (obr. 1.1). Najprv prebehne statická fáza plánovania, ktorej cieľom je nájdenie čo najlepšej cesty z počiatočnej do cieľovej konfigurácie. Po nej nasleduje dynamická fáza, ktorej hlavným cieľom je zabezpečiť, aby malo UAV po celý čas k dispozícii plán bezkolíznej cesty. Popri tom priebežne optimalizuje sledovanú cestu a udržiava vytváranú stromovú štruktúru. Celý proces plánovania je ukončený dosiahnutím cieľovej konfigurácie (úspešný koniec) alebo nenájdením cesty v definovanom časovom limite počas statickej fázy (zlyhanie). Priebehu plánovania (statickej aj dynamickej fázy) je možné zmeniť cieľovú konfiguráciu robota.



Obrázok 1.1 Vývojový diagram hlavnej slučky plánovacieho algoritmu (principiálny náhľad).

1.2 Statická fáza plánovania

Počas statickej fázy plánovania algoritmus náhodným vzorkovaním prehľadáva priestor. Ak je nájdené prvotné riešenie, teda cesta z počiatočnej do cieľovej konfigurácie, algoritmus pokračuje v prehľadávaní priestoru a hľadaní lepšej cesty až do uplynutia časového limitu t_{limit}. Na obr. 1.2 je vyobrazený priebeh statickej fázy plánovania. Pred začiatkom plánovania je množina uzlov vytváraného stromu V inicializovaná uzlom predstavujúcim cieľovú konfiguráciu robota $v_{cieľ}$, ktorý je koreňom stromu (krok 1). Taktiež je inicializovaná množina uzlov reprezentujúcich najlepšiu nájdenú cestu τ (krok 2). Následne sa začne hľadanie cesty z $v_{cieľ}$ do $v_{štart}$. Krok 4 reprezentuje ukončovaciu podmienku plánovania – časový limit. V prvom kroku samotného plánovania (krok 5) sa algoritmus pokúsi orezať koncové uzly stromu tak, aby počet uzlov vo V neprekročil ich maximálny počet. Množstvo súčasne orezaných uzlov je definované parametrom n. V nasledujúcom kroku (krok 6) algoritmus rozširuje strom pridaním nového uzla $v_{nový}$ získaného náhodným vzorkovaním. Po pridaní $v_{nový}$ do stromu je zavolaná rutina prepojovania uzla $v_{nový}$ s jeho susednými uzlami (podobne ako je to v RRT*) a zároveň sa začne kaskádová aktualizácia ohodnotenia

susedov, pre ktorých sa $v_{nový}$ stane ich novým rodičovským uzlom a ich potomkov (tak ako v RRT^X). V poslednom kroku slučky statickej fázy plánovania je skontrolované, či sa pridaním $v_{nový}$ do V alebo postupným prepojovaní a šírením informácie nezmenilo pripojenie $v_{štart}$ k stromu alebo jeho ohodnotenie a tým sa našla nová najlepšia cesta τ (krok 8). Po uplynutí časového limitu sa overí, či bola nájdená nejaká cesta, teda τ nie je prázdna množina. Ak bola cesta nájdená, algoritmus pokračuje do druhej, dynamickej fázy plánovania. V opačnom prípade algoritmus vyhlási neúspech nadradenému systému, ukončí plánovanie a čaká na novú úlohu.

Algorithm 1 Statická fáza plánovania

1: $V \leftarrow \{v_{\text{cieľ}}\}$
2: $\tau \leftarrow \{\emptyset\}$
3:
4: **while** $t < t_{\text{limit}}$ **do**
5: OrezaťStrom(V, n)
6: $v_{\text{nový}} \leftarrow$ RozšíriťStrom(V)
7: Prepájanie&ŠírenieInformácie$(v_{\text{nový}})$
8: $\tau \leftarrow$ AktualizáciaCesty$(v_{\text{štart}})$
9: **end while**
10:
11: **if** NemáRodiča$(v_{\text{štart}})$ **then**
12: **KONIEC**
13: **else**
14: **DYNAMICKÁ FÁZA**
15: **end if**

Obrázok 1.2: Algoritmus statickej fázy plánovania.

1.3 Dynamická fáza plánovania

Dynamická fáza plánovania nasleduje po statickej za predpokladu, že sa podarilo nájsť bezkolízne prepojenie $v_{štart}$ s $v_{cieľ}$. Počas dynamickej fázy sa predpokladá, že robot môže vykonávať pohyb v priestore a sleduje naplánovanú cestu. Na obr. 1.3 je zobrazený algoritmus hlavnej slučky dynamickej fázy. Algoritmus dynamickej fázy je ukončený dosiahnutím cieľového stavu robotom (krok 1). Alternatívnym spôsobom ukončenia je zastavenie vykonávania algoritmu nadradeným systémom, ktorý volá plánovací algoritmus, z dôvodu prekročenia časového limitu pre preplánovanie. V prvom kroku každej iterácie sa aktualizuje $v_{štart}$ na aktuálnu polohu robota. Pokiaľ bola aktualizovaná mapa prostredia

Algorithm 2 Dynamická fáza plánovania

```
1: while v_štart ≠ v_cieľ do
2:     v_štart ← AktualizáciaNaPolohuRobota()
3:     if NováMapa() then
4:         KontrolaÚsekuCesty(τ, d)
5:     end if
6:     OrezaťStrom(V, n)
7:     v_nový ← RozširiťStrom()
8:     Prepájanie&ŠírenieInformácie(v_nový)
9:     τ ← AktualizáciaCesty(v_štart)
10: end while
```

Obrázok 1.3: Dynamická fáza plánovania.

(krok 3), existuje šanca prerušenia cesty τ novou prekážkou. Preto je skontrolovaný úsek aktuálnej cesty τ do vzdialenosti d od $v_{štart}$ (krok 4). V prípade hroziacej kolízie je v rámci tohto kroku cesta τ upravená tak, aby neobsahovala uzly stromu, ktoré sú v kolízii s prekážkou. Algoritmus ďalej pokračuje podobne ako počas statickej fázy. V kroku 6 sa udržuje počet uzlov stromu tak, aby neprekročil maximálny počet uzlov, ktoré môže strom obsahovať. V kroku 6 prebehne algoritmus rozširovania stromu. V nasledujúcom kroku prebehne kaskádové prepájanie uzlov a šírením informácie. V poslednom kroku je aktualizovaná najlepšia nájdená cesta.

Na obr. 1.4 sa nachádza algoritmus preplánovania cesty a na obr. 1.5 je vyobrazený prvý krok preplánovania. V procese preplánovania cesty sú uzly stromu rozdelené do troch podmnožín – uzly v kolízii s prekážkou, uzly odstrihnuté od stromu, uzly neovplyvnené prekážkou. Ako prvé sú z množiny $N_{v_{int}}$ identifikované kolízne uzly (obr. 1.5b červená) a uzly, ktoré stratili prepojenie so svojim rodičom vplyvom prekážky (obr. 1.5b, zelená). Uzly v kolízii s prekážkou tvoria podmnožinu $V_{kolízne} \subset V$ a uzly odstrihnuté od stromu tvoria podmnožinu $V_{odstrihnuté} \subset V$ (obr. 1.4, krok 4). Červený štvoruholník z obr. 1.5b reprezentuje neznámu prekážku. Uzly 5 a 6 sú označené ako kolízne uzly a patria do $V_{kolízne}$, pretože sa nachádzajú vo vnútri prekážky. Uzly 13 a 9 nie sú v kolízii s prekážkou, ale rodič uzla 9 je v kolízii a prepojenie uzla 13 s jeho rodičom (uzol 2) je prerušené prekážkou. Preto sú označené ako odstrihnuté (osirotené) uzly a patria do $V_{odstrihnuté}$. Uzol $v_{štart}$ sa nachádza od v_{int} vo vzdialenosti väčšej ako $r_{kolizia}$ no prerušením cesty sú od stromu odstrihnuté všetky uzly τ nachádzajúce sa pred prekážkou z pohľadu $v_{štart}$. Z tohto dôvodu je $v_{štart}$ na obr. 1.5b taktiež označený ako odstrihnutý uzol aj keď zatiaľ nepatrí do $V_{odstrihnuté}$. Popri

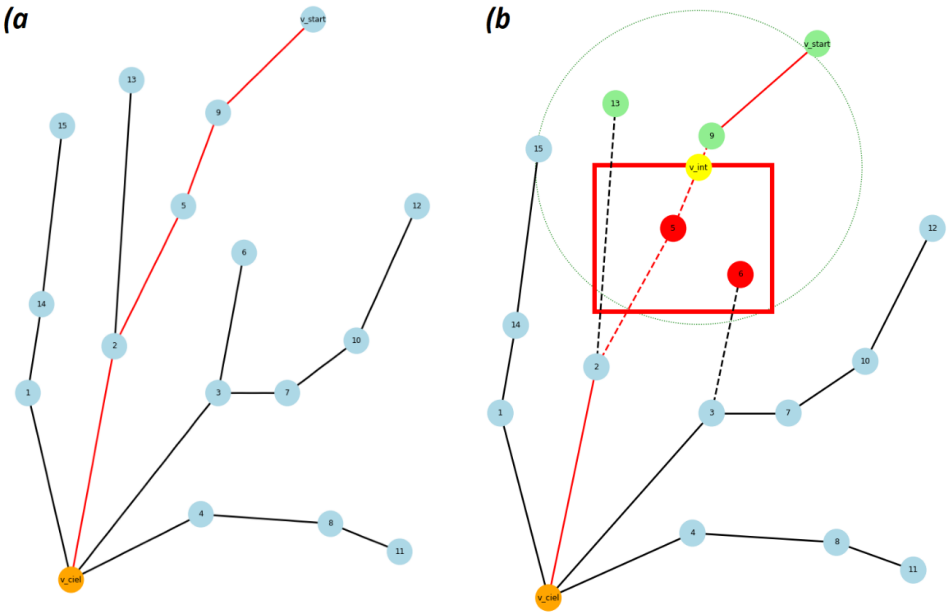
tom sú pre uzly z $N_{v_{int}}$, ktoré nie sú ovplyvnené prekážkou (svetlomodrá) skontrolované prepojenia s ich susedmi na bezkolíznosť. Hlavným cieľom tohto kroku je aktualizovať stav prepojeniam neovplyvneného uzla z $N_{v_{int}}$ s jeho susedmi, ktoré už boli overené na bezkolíznosť počas pridávania uzla z $N_{v_{int}}$ do stromu alebo počas kaskádového prepojovania a šírenia informácie. Podľa obr. 1.5b, pre uzol 15 budú overené už pretým kontrolované prechody medzi ním a jeho susedmi.

Algorithm 3 PreplánovanieCesty

1: $V_{kolizne} \leftarrow \{\emptyset\}$
2: $V_{odstrihnute} \leftarrow \{\emptyset\}$
3: $Q \leftarrow \{\emptyset\}$
4: $V_{kolizne}, V_{odstrihnute} \leftarrow$ UzlyOvplyvnenéPrekážkou($N_{v_{int}}$)
5: $V_{odstrihnute} \leftarrow V_{odstrihnute} \cup$ PotencialneOdstrihnutéUzly($V_{odstrihnute}$)
6: $V_{odstrihnute} \leftarrow V_{odstrihnute} \cup$ Potomkovia($V_{kolizne}$)
7: VymažPotomkov($V_{kolizne}$)
8: VymažRodiča($V_{kolizne}$)
9: VymažRodiča($V_{odstrihnute}$)
10: $V_{odstrihnute} \leftarrow V_{odstrihnute} \cup$ ŠírenieInformácieOdstrihnutia($V_{odstrihnute}$)
11: $Q \leftarrow$ ValidníSusedia($V_{odstrihnute}$)
12: $\tau_{new} \leftarrow$ RýchlePrepojenie()
13: **if** $\tau_{new} \neq \{\emptyset\}$ **then**
14: $\tau \leftarrow \tau_{new}$
15: **KONIEC**
16: **end if**
17: $\tau_{new} \leftarrow$ VynútenéPrepájanie&ŠírenieInformácie()
18: **if** $\tau_{new} \neq \{\emptyset\}$ **then**
19: $\tau \leftarrow \tau_{new}$
20: **KONIEC**
21: **else**
22: **ERROR** – Cesta nenájdená
23: **end if**

Obrázok 1.4: Algoritmus preplánovania - nájdenia nového prepojenia $v_{štart}$ s $v_{cieľ}$ a aktualizovanie uzlov stromu.

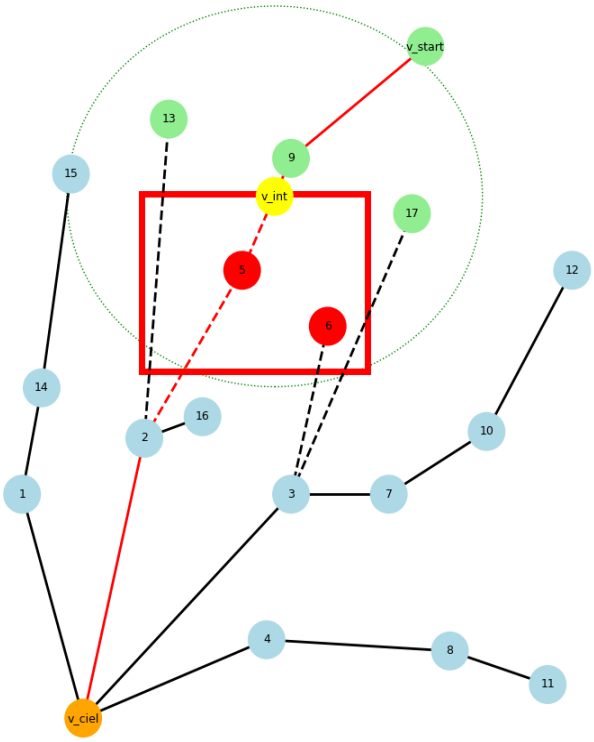
Prerušenie cesty prekážkou predstavuje závažnú zmenu pre vytváraný strom. Pre zachovanie jeho integrity je nutné aktualizovať ostatné časti stromu, ktoré sú ovplyvnené prekážkou, ale sú od miesta kolízie vzdialenejšie ako $r_{kolízia}$. Pred šírením informácie o prekážke sú navyše skontrolované uzly nachádzajúce sa mimo $r_{kolizia}$, ktoré majú spoločného rodiča s uzlami z $V_{odstrihnuté}$ (obr. 1.4, krok 5). Uzly, ktoré sú súrodencami uzlov z $V_{odstrihnuté}$ sú potenciálne ovplyvnené prekážkou, keďže sa nachádzajú v jej blízkom okolí a ich rodič stratil prepojenie s niektorými z ich súrodencami, ktorý sa už nachádzajú v $V_{vodstrihnuté}$.



Obrázok 1.5: Priebeh preplánovania cesty okolo vopred neznámej prekážky: a – pôvodná podoba stromu pred kontrolou úseku cesty, b – identifikovanie vplyvu prekážky na sledovanú cestu a stromovú štruktúru.

V príklade na obr. 1.6 sú do stromu z obr. 1.5 doplnené uzly 16 a 17, vďaka ktorým prebehne kontrola súrodencov uzlov z $V_{odstrihnuté}$. Prechod medzi uzlami 2 a 16 je skontrolovaný kvôli uzlu 13 a prechod medzi uzlami 3 a 7 je skontrolovaný kvôli uzlu 17. Ak by niektoré z týchto prepojení bolo prerušené, tak daný súrodenec (uzol 16, 7) by bol pridaný do $V_{odstrihnuté}$. Podmnožina $V_{odstrihnuté}$ je ďalej rozšírená o potomkov uzlov z $V_{kolízne}$, ktoré sa v $V_{odstrihnuté}$ ešte nenachádzajú (obr. 1.4, krok 6).

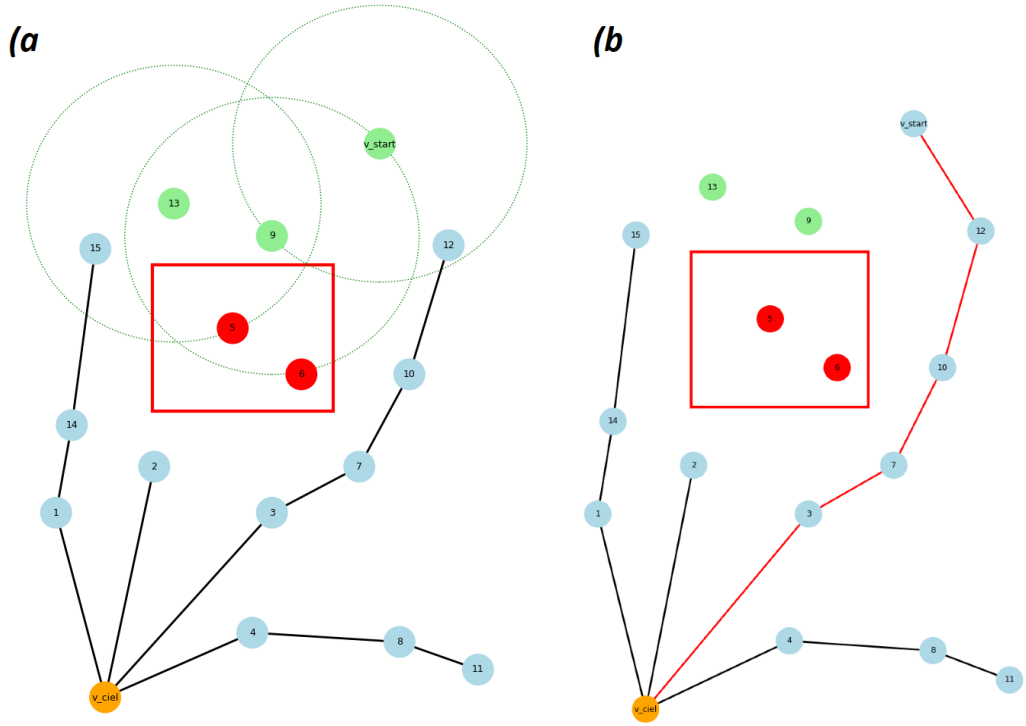
Na základe informácii získaných v predošlých krokoch sú aktualizované prepojenia uzlov nachádzajúcich sa v $V_{kolízne}$ a $V_{odstrihnuté}$ (obr. 1.4, kroky 7 - 9). Uzlom z $V_{odstrihnuté}$ sú odstránené ich prepojenia k rodičom. Uzlom z $V_{kolízne}$ sú okrem prepojení s ich rodičmi odstránené aj prepojenie s ich potomkami. Vďaka tomu sú uzly z $V_{kolízne}$ a $V_{odstrihnuté}$ oddelené od neovplyvnených uzlov z V. V ďalšom kroku nastane cez uzly z $V_{odstrihnuté}$ šírenie informácie o odstrihnutí od stromu všetkým ich potomkom (obr. 1.4, krok 10). Uzlom, ktoré dostali informáciu o odstrihnutí od stromu sú odstránené prepojenia s ich rodičmi a potomkami (obr. 1.7a), aktualizované ohodnotenia a sú pridané do $V_{odstrihnuté}$.



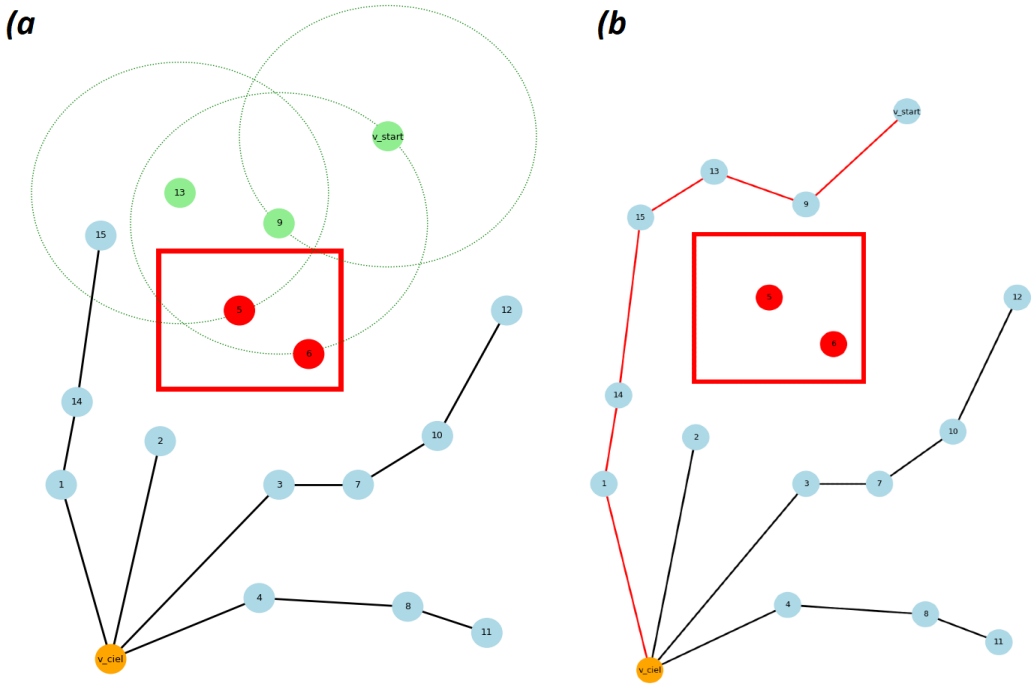
Obrázok 1.6: Kontrola súrodencov uzlov z $V_{odstrihnuté}$.

V ďalšom kroku sú uzly susedné s uzlami z $V_{odstrihnuté}$, ktoré nie sú ovplyvnené prekážkou a majú pripojenie k stromu pridané do prioritného radu Q určeného pre prepojovanie (obr. 1.4, krok 11). Tento krok je možné pozorovať na obr. 1.7a, kde sú nájdené uzly nachádzajúce sa vo vzdialenosti r od uzlov 13, 9 a $v_{štart}$. Tomuto kritériu vyhovuje uzol 15, ktorý je pridaný do Q. Q je využitý v algoritme pre vynútené prepojovanie a šírenie informácie.

Po spracovaní odstrihnutých uzlov stromu nasleduje samotný proces preplánovania. Keďže všetky odstrihnuté uzly stromu sú už známe, plánovací algoritmus najprv vyskúša, či je možné niektorý z odstrihnutých uzlov sledovanej cesty τ pripojiť k stromu cez jeho suseda. Na obr. 1.7b je zobrazený výsledok úspešného rýchleho prepojenia uzla odstrihnutej časti cesty s niektorým z uzlov stromu – prepojenie $v_{štart}$ s uzlom 12. Na obr. 1.8b je zobrazený náhľad na nájdenie novej cesty okolo prekážky vynúteným prepojovaním a šírením informácie, pretože v r okolí uzlov odstrihnutej časti cesty sa nenachádza žiaden uzol stromu.



Obrázok 1.7: Priame prepojenie odstrihnutej časti cesty so stromom. a – Odstránenie prepojení kolíznych a odstrihnutých uzlov a hľadanie validných susedov v okolí odstrihnutých uzlov. b – Priame prepojenie uzla odstrihnutej časti cesty.



Obrázok 1.8: Preplánovania cesty okolo neznámej prekážky vynúteným prepojovaním. a – Odstránenie prepojení kolíznych a odstrihnutých uzlov a hľadanie validných susedov v okolí odstrihnutých uzlov. b – Nájdenie alternatívnej cesty vynúteným prepojovaním.

2 Paralelizovanie plánovacieho algoritmu

Okrem návrhu plánovacieho algoritmu je dôležitá aj jeho implementácia a možnosť prispôsobenia samotného návrhu za účelom čo najefektívnejšieho využitia hardvéru, na ktorom je algoritmus vykonávaný. Vnorené systémy ako sú napr. palubné počítače UAV sa za posledné roky výrazne posunuli v pred z hľadiska ich výkonu, no stále je ich výkon obmedzený v porovnaní so stolovými počítačmi. Palubný počítač UAV vykonáva množstvo iných, častokrát dôležitejších úloh než je plánovanie. Toto je jednou z motivácii pre čo najrýchlejšie vykonanie plánovania cesty a procesu preplánovania. Okrem vhodného návrhu algoritmu sa čas jeho vykonávania dokáže významne urýchliť paralelizovaním jeho výpočtovo náročnejších častí. Vďaka moderným vnoreným systémom je možné využiť hardvérové akcelerátory ako je GPU, ktoré majú k dispozícii. Paralelizovanie tak nie je obmedzené len na CPU. Výhodou využitia hardvérových akcelerátorov je možné celkové zrýchlenie výpočtov a odľahčenie CPU.

V prvom rade je ale nutné identifikovať časti algoritmu, ktoré je možné paralelizovať. Zároveň musia byť dostatočne výpočtovo náročné, teda dodatočný čas potrebný na vytváranie vlákien alebo presun dát do/z GPU je zanedbateľný v porovnaní časom ušetreným paralelizovaním. Časti algoritmu, kde sa vykonáva množstvo kontrol kolízii sú vhodným kandidátom pre paralelizovanie. Ďalším vhodným kandidátom sú tie časti, kde sa manipuluje s väčším množstvom uzlov. Algoritmy alebo ich časti, ktoré boli identifikované ako vhodné pre paralelizovanie a zároveň zrýchlené paralelizovaním:

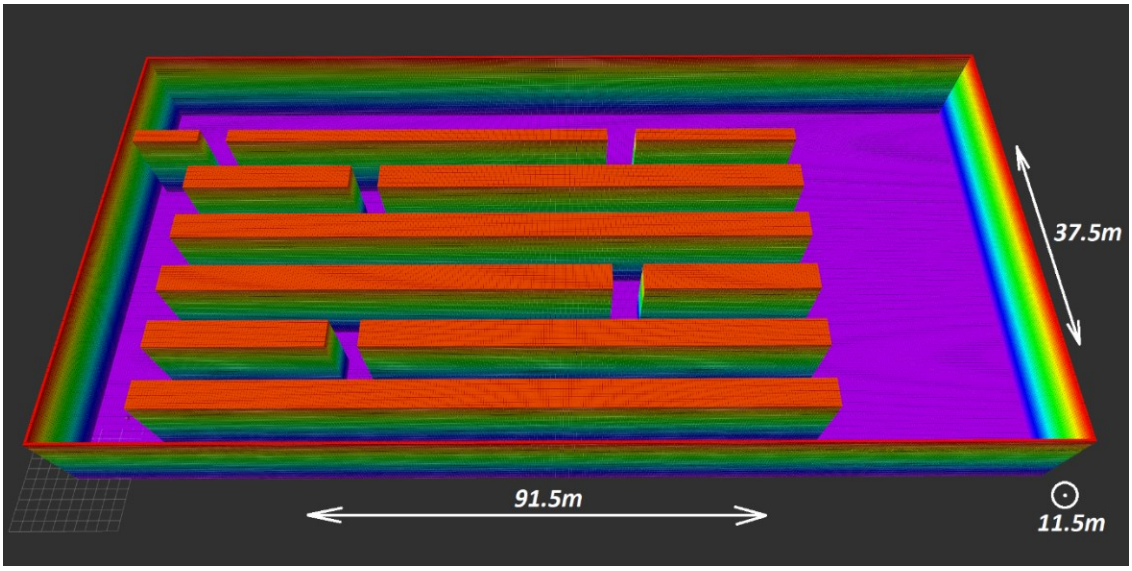
- algoritmus aktualizácie cieľového uzla stromu – uzla polohy robota,
- algoritmus kontroly uzlov v okolí prekážky (CPU + GPU),
- algoritmus kontroly potenciálne odstrihnutých uzlov od stromu,
- algoritmu šírenia informácie uzlom o odstrihnutí od stromu,
- algoritmu inicializácie množiny validných susedov odstrihnutých uzlov stromu,
- algoritmus rýchleho prepojenia odstrihnutej časti cesty k stromu,
- algoritmus vynúteného prepájania uzlov a šírenia informácie,
- algoritmus zmeny cieľa plánovania – koreňa stromu (CPU + GPU).

3 Experimentálne vyhodnotenie plánovacieho algoritmu

Táto kapitola sa venuje vyhodnoteniu výsledkov získaných počas simulačných experimentov. Experimenty boli vykonané na palubnom počítači UAV (Jetson Xavier NX, 6 CPU jadier a 384 CUDA jadier), aby bola overená rýchlosť algoritmu na zariadení s obmedzeným výpočtovým výkonom. V rámci tejto je kladený dôraz na rýchlosť algoritmu resp. preplánovania prerušenej cesty. Implementácia navrhnutého plánovacieho algoritmu je vytvorená s využitím OMPL [13].

3.1 Simulačné prostredie

Simulácie prebiehajú v prostredí, ktoré reprezentuje fiktívny skladový priestor (obr. 3.1). Simulované UAV sa môže voľne pohybovať v priestore v osiach x, y, z. Veľkosť mapy simulovaného skladového priestoru je 91,5m/37,5m/11,5m (dĺžka/šírka/výška). Mapu priestoru reprezentuje oktálový strom vytvorený prostredníctvom knižnice OctoMap [6] a PCL [15]. Najmenší rozmer bunky oktálového stromu je 0.2m. Vytvorený oktálový strom je využitý počas detegovania kolízie geometrického modelu UAV s prostredím.



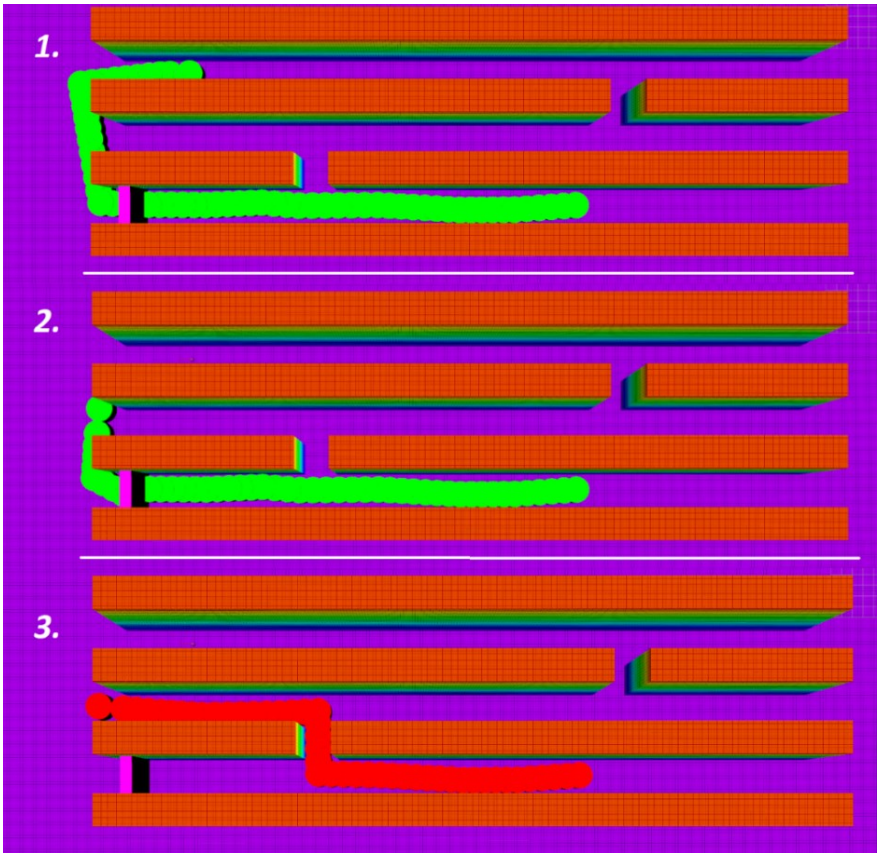
Obrázok 3.1: Oktálový strom ako mapa simulovaného skladového priestoru.

3.2 Vyhodnotenie paralelizovania plánovacieho algoritmu

V nasledujúcej podkapitole sú vyhodnotené výsledky paralelizovania častí plánovacieho algoritmu, ktoré sú diskutované v kapitole 4. Cieľom paralelizovania je znížiť výpočtový čas vybraných častí algoritmu a dosiahnuť tak preplánovanie prerušenej cesty v „reálnom

čase“. Využívané UAV dokáže aktualizovať mapu prostredia s frekvenciou 5Hz – 10Hz. Z toho vyplýva, že celkový proces preplánovania nesmie presiahnuť 200ms. . Cieľom je ale dosiahnuť čo najkratší čas preplánovania, aby algoritmus dokázal využiť čo najvyššiu frekvenciu aktualizovania mapy.

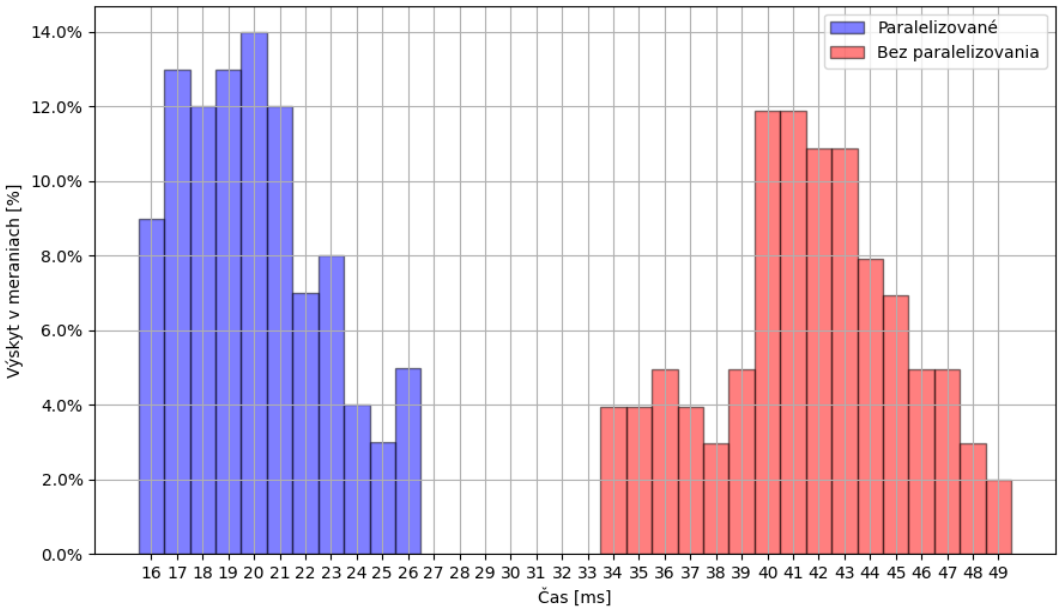
Jednotlivé experimenty pre paralelizované a neparalelizované verzie častí algoritmu sú realizované v troch konfiguráciách, pre ktoré sa mení maximálny počet uzlov v strome (4000, 8000, 16000 uzlov). Pokiaľ nie je špecifikované inak, v každom experimente bolo vykonaných 100 meraní – behov simulácie, z ktorých boli zhromaždené dáta pre vyhodnotenie. Prehľadávaná vzdialenosť v okolí miesta kolízie cesty s prekážkou je r = 7,5m. Na obr. 3.2 je znázornený náhľad priebehu testovania preplánovania cesty plánovacím algoritmom ak vopred neznáma prekážka prerušila cestu naplánovanú počas statickej fázy plánovania. V prvom kroku UAV začne sledovať naplánovanú cestu (zelená čiara). Počas prechodu UAV prostredím je do mapy pridaná nová prekážka (cyklámenový kváder) blokujúca uličku. Momentom priblíženia sa UAV k prekážke na vzdialenosť menšiu ako je maximálna vzdialenosť detegovania prekážky je spustený proces preplánovania. V kroku 3 je nájdená alternatívna cesta, ktorou sa UAV môže vydať (červená čiara).



Obrázok 3.2: Náhľad na priebeh preplánovania cesty prerušenej prekážkou.

3.2.1 Vynútené prepojovanie uzlov

Na obr. 3.3 je vyobrazený histogram časov vykonávania procesu vynúteného prepojovania kedy je maximálny počet uzlov stromu N = 8000. Zvýšením počtu uzlov v strome je pozorovateľné celkové predĺženie času potrebného na prepojovanie. Taktiež sa prehlbuje časový rozdiel medzi paralelizovanou a neparalelizovanou verziou algoritmu.



Obrázok 3.3: Histogram času vykonania algoritmu vynúteného prepojovania (N = 8000, r = 7,5m)

Tab. 1: Výsledok paralelizovania algoritmu vynúteného prepojovania (N = 4000, r = 7.5m)

	min [ms]	max [ms]	priemer [ms]
Neparalelizovaná	19	35	29.88
Paralelizovaná	7	20	12.57
Zrýchlenie	-	-	2,37x

Tab. 2: Výsledok paralelizovania algoritmu vynúteného prepojovania (N = 8000, r = 7,5m)

	min [ms]	max [ms]	priemer [ms]
Neparalelizovaná	34	49	41.52
Paralelizovaná	16	26	19.99
Zrýchlenie	-	-	2,07x

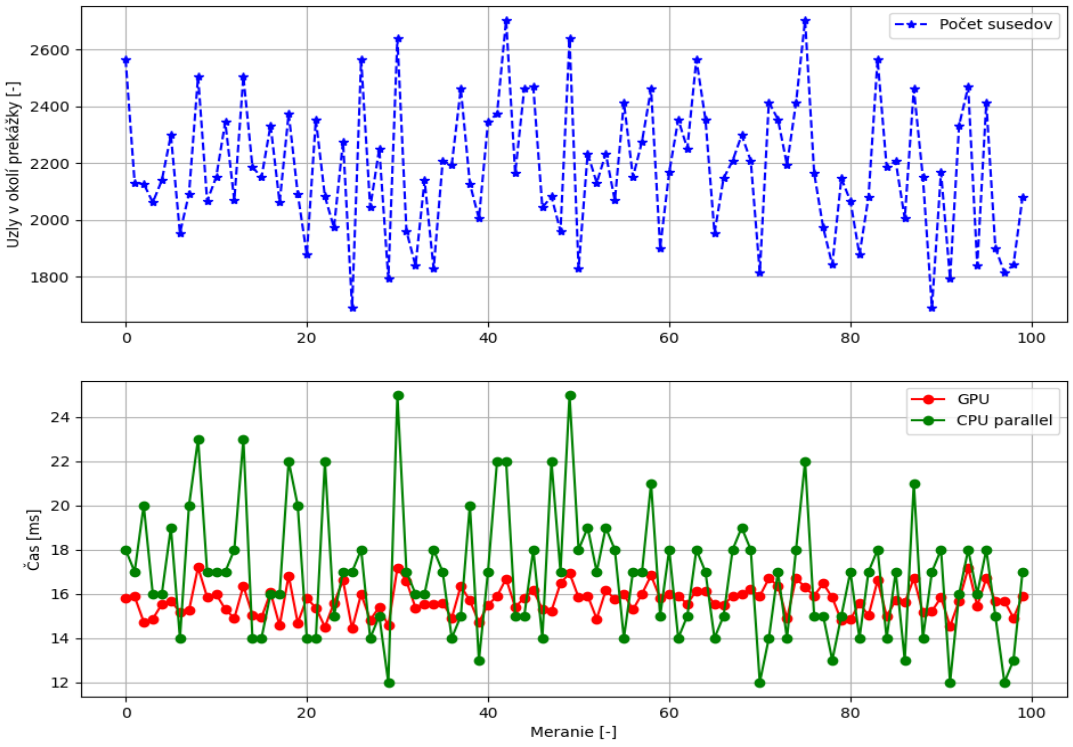
Tab. 3: Výsledok paralelizovania algoritmu vynúteného prepojovania (N = 16000, r = 7,5m)

	min [ms]	max [ms]	priemer [ms]
Neparalelizovaná	53	71	58,41
Paralelizovaná	21	34	27,95
Zrýchlenie	-	-	2,09x

V rámci tabuliek tab. 1 až tab. 3 sú zhrnuté výsledky vykonaných experimentov pre všetky konfigurácie maximálneho počtu uzlov v strome. V každej tabuľke je uvedený maximálny, minimálny a priemerný čas vykonania algoritmu. V poslednom riadku tabuľky je zhrnuté zrýchlenie priemerného času vykonania paralelizovanej verzie algoritmu v porovnaní s neparalelizovanou.

3.2.2 Identifikácia uzlov ovplyvnených prekážkou

Na obrázku obr. 3.4 je vyobrazené porovnanie dĺžky trvania identifikovania ovplyvnených uzlov v okolí prekážky pre prípad paralelizovania algoritmu na CPU a GPU.



Obrázok 3.4: Porovnanie času potrebného pre kontrolu uzlov v okolí prekážky algoritmom paralelizovaným na CPU a na GPU (N = 16000, r = 7,5m).

Zrýchlenie algoritmu paralelizovaním na GPU sa prejavuje až s narastajúcim počtom uzlov v okolí prekážky, kedy skrátenie času stráveného detegovaním kolízie kompenzuje čas strávený nahrávaním dát do GPU, prípravou dát pred nahratím do GPU a spracovaním dát stiahnutých z GPU. V rámci tabuliek tab. 4 až tab. 6 sú zhrnuté výsledky vykonaných experimentov pre všetky konfigurácie maximálneho počtu uzlov v strome. V každej tabuľke je uvedený maximálny, minimálny a priemerný čas vykonania algoritmu. V poslednom riadku tabuľky je zhrnuté zrýchlenie priemerného času vykonania paralelizovanej verzie algoritmu v porovnaní s neparalelizovanou. Tabuľka tab. 6 je doplnená o výsledky paralelizovania s využitím GPU a zrýchlenie GPU akcelerovanej verzie v porovnaní CPU paralelizovanou verziou.

Tab. 4: Výsledok paralelizovania algoritmu pre kontrolu uzlov v okolí prekážky (N = 4000, r = 7,5m)

	min [ms]	max [ms]	priemer [ms]
Neparalelizovaná	6	15	8,28
Paralelizovaná	1	7	3,54
Zrýchlenie	-	-	2,33x

Tab. 5: Výsledok paralelizovania algoritmu pre kontrolu uzlov v okolí prekážky (N = 8000, r = 7,5m)

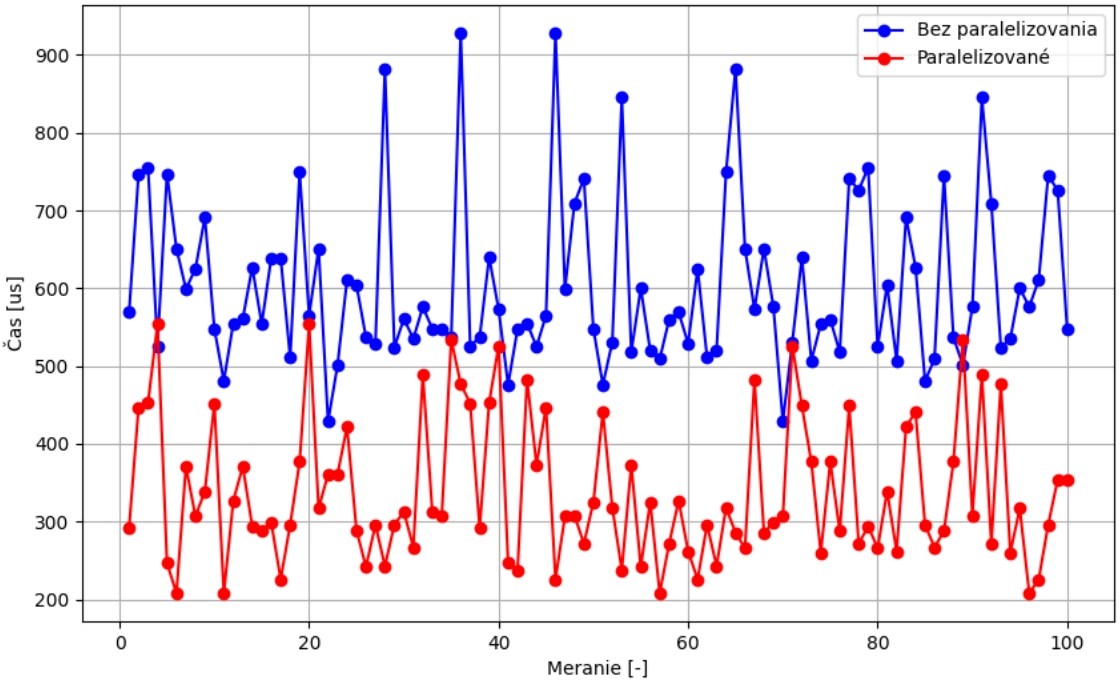
	min [ms]	max [ms]	priemer [ms]
Neparalelizovaná	14	24	18,2
Paralelizovaná	5	14	7,54
Zrýchlenie	-	-	2,41x

Tab. 6: Výsledok paralelizovania algoritmu pre kontrolu uzlov v okolí prekážky (N = 16000, r = 7,5m).

	min [ms]	max [ms]	priemer [ms]	zrýchlenie [-]
Neparalelizovaná	29,00	62,00	36,12	
Paralelizovaná (CPU)	12,00	25,00	16,86	2,14x
Paralelizovaná (GPU)	14,09	17,29	15.78	1.48x

3.2.3 Šírenie informácie uzlom o odstrihnutí od stromu

Na obrázku obr. 3.5 sú vyobrazené časy jednotlivých meraní získaných počas experimentov, kedy bol maximálny počet uzlov v strome obmedzený na N = 8000 uzlov.



Obrázok 3.5: Porovnanie dĺžky trvania šírenia informácie o kolízii (N = 16000).

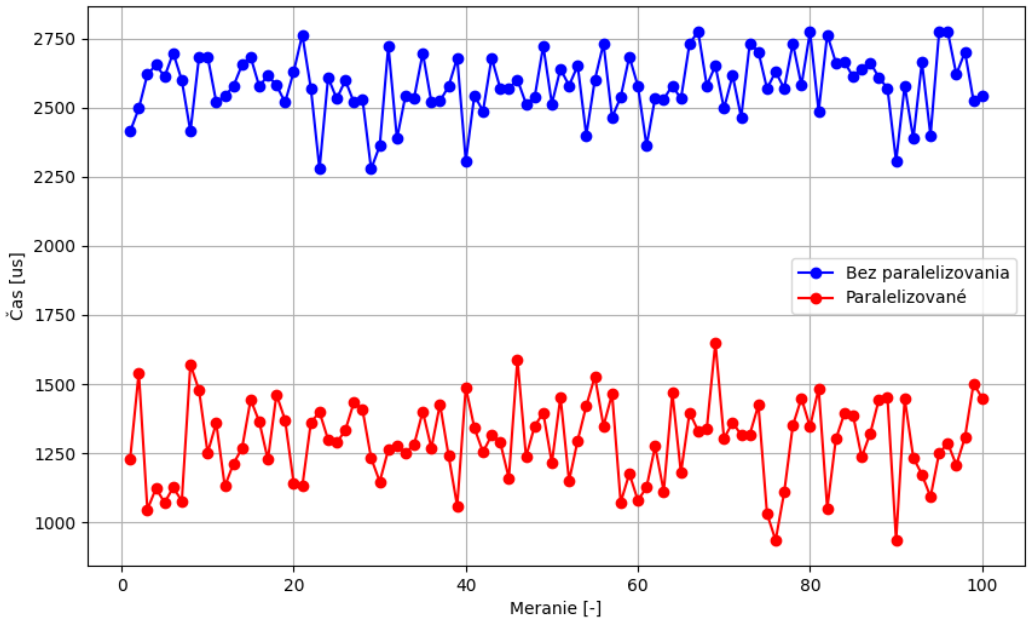
V tabuľke tab. 7 sú zhrnuté výsledky vykonaných experimentov. Proces šírenia informácie je výpočtovo menej náročný proces. Zrýchlenie algoritmu v prípade počtu uzlov 4000 nie je výrazné v porovnaní s neparalelizovanou verziou. Výhoda paralelizovania sa začína prejavovať až so zvyšujúcim sa počtom uzlov v strome.

Tab. 7: Výsledné zhrnutie výsledkov paralelizovania procesu šírenia informácie o kolízii odstrihnutým častiam stromu.

	počet uzlov [-]	min [us]	max [us]	priemer [us]	zrýchlenie [-]
Neparalelizovaná	4000	97	476	170	-
Paralelizovaná	4000	50	270	151,02	1,13x
Neparalelizovaná	8000	247	773	365,38	-
Paralelizovaná	8000	123	410	222,68	1,64x
Neparalelizovaná	16000	429	928	603,88	-
Paralelizovaná	16000	208	554	337,5	1,78x

3.2.4 Kontrola potenciálne odstrihnutých uzlov

Na obrázku obr. 3.6 sú vykreslené priebehy dĺžky trvania behov algoritmu vykonaných počas experimentov, kedy bol maximálny počet uzlov v strome obmedzený na N = 16000 uzlov.



Obrázok 3.6: Porovnanie dĺžky kontroly potenciálne odstrihnutých uzlov (N = 16000).

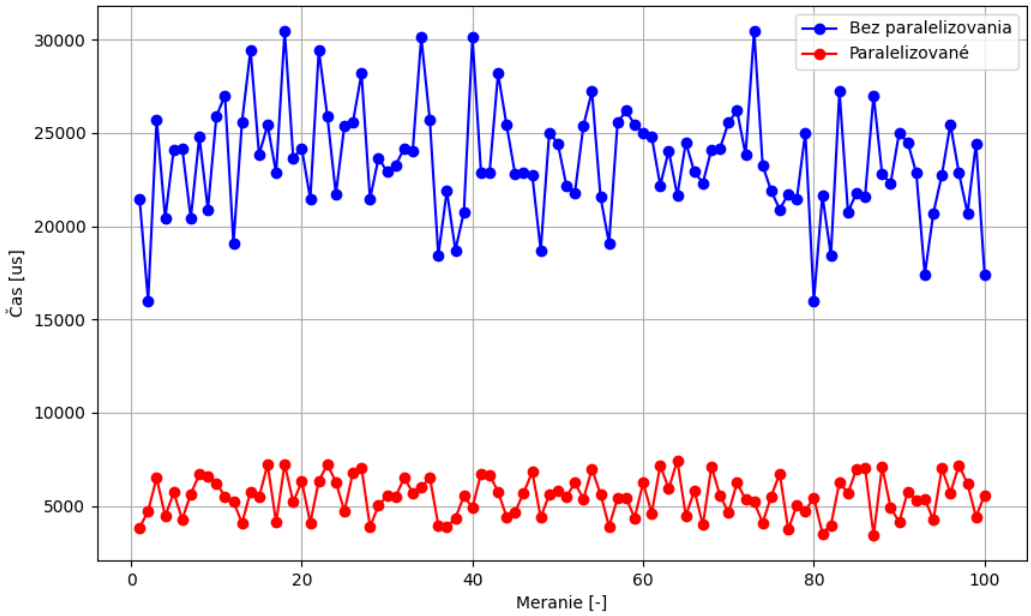
V tabuľke tab. 8 sú zhrnuté maximálne, minimálne a priemerné časy trvania paralelizovanej a neparalelizovanej verzie algoritmu zo všetkých vykonaných experimentov a zrýchlenia priemernej dĺžky trvania algoritmu.

Tab. 8: Výsledné zhrnutie výsledkov paralelizovania algoritmu hľadania potenciálne odstrihnutých uzlov.

	počet uzlov [-]	min [us]	max [us]	priemer [us]	zrýchlenie [-]
Neparalelizovaná	4000	742	949	822,54	-
Paralelizovaná	4000	532	845	747,42	1,10x
Neparalelizovaná	8000	1499	1797	1601,58	-
Paralelizovaná	8000	898	1212	1050,04	1,52x
Neparalelizovaná	16000	2278	2775	2580,3	-
Paralelizovaná	16000	935	1649	1293,79	1,99x

3.2.5 Inicializácia prioritného radu pre vynútené prepojovanie

Na obr. 3.7 sú vykreslené priebehy dĺžky trvania behov algoritmu vykonaných počas experimentov, kedy bol maximálny počet uzlov v strome obmedzený na N = 16000 uzlov.



Obrázok 3.7: Porovnanie dĺžky trvania inicializácie prioritnej rady (N = 16000).

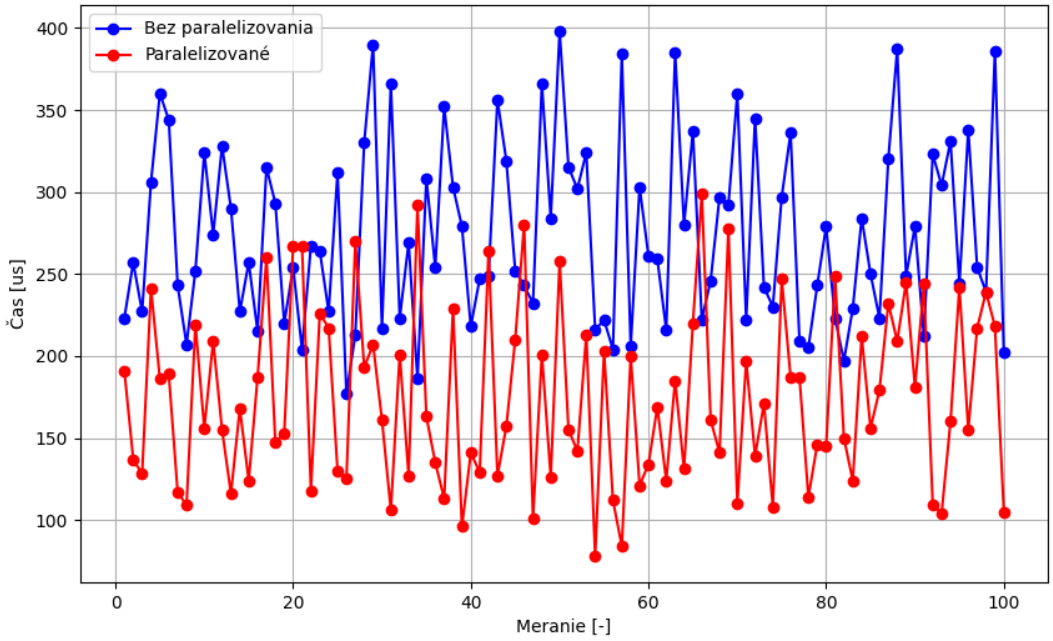
V tabuľke tab. 9 sú zhrnuté maximálne, minimálne a priemerné časy trvania paralelizovanej a neparalelizovanej verzie algoritmu zo všetkých vykonaných experimentov. Algoritmus inicializácie prioritného radu spracováva veľké množstvo uzlov. V tomto prípade je možné pozorovať viac ako dvojnásobné zrýchlenie algoritmu jeho paralelizovaním aj v prípade nižšieho množstva uzlov v strome.

Tab. 9: Zhrnutie výsledkov paralelizovania algoritmu inicializácie prioritného radu validnými susedmi odstrihnutých uzlov stromu.

	počet uzlov [-]	min [us]	max [us]	priemer [us]	zrýchlenie [-]
Neparalelizovaná	4000	765	3606	1473, 66	-
Paralelizovaná	4000	290	1276	593,44	2,48x
Neparalelizovaná	8000	4377	11783	7977,08	-
Paralelizovaná	8000	1307	3822	2330,88	3,42x
Neparalelizovaná	16000	16003	30468	23485,22	-
Paralelizovaná	16000	3432	7441	5510,2	4,26x

3.2.6 Rýchle prepojenie

Na obrázku obr. 3.8 sú zobrazené časy trvania rýchleho prepojenia, kedy je maximálny počet uzlov v strome obmedzený na N = 16000 uzlov.



Obrázok 3.8: Porovnanie dĺžky trvania algoritmu rýchleho prepojenia odstrihnutej časti cesty k stromu (N = 16000).

V tabuľke tab. 10 sú zhrnuté maximálne, minimálne, priemerné časy trvania algoritmu a priemerné zrýchlenie algoritmu paralelizovaním. V tomto prípade je možné pozorovať negatívny vplyv nákladov na vytváranie vlákien na čas vykonania algoritmu v prípade nižšieho počtu uzlov v strome (N = 4000).

Tab. 10: Zhrnutie výsledkov paralelizovania algoritmu rýchleho pripojenia odrezaného úseku cesty k stromu.

	počet uzlov [-]	min [us]	max [us]	priemer [us]	zrýchlenie [-]
Neparalelizovaná	4000	34	203	110, 84	-
Paralelizovaná	4000	49	188	113, 14	0,97x
Neparalelizovaná	8000	58	260	142,08	-
Paralelizovaná	8000	43	188	122	1,16x
Neparalelizovaná	16000	177	398	274,34	-
Paralelizovaná	16000	78	299	174,6	1,57x

3.2.7 Celkový čas preplánovania cesty

V rámci tabuľky tab. 11 sú prezentované výsledné časy trvania celého procesu preplánovania cesty v prípade, kedy sa cestu nepodarilo preplánovať rýchlim prepojením odrezanej časti cesty so stromom. V tabuľke tab. 12 je zhrnutie výsledných časov plánovania, kedy bolo rýchle prepojenie úspešné. V tabuľke sú uvedené maximálne, minimálne a priemerné časy preplánovania cesty spolu s priemerným zrýchlením procesu preplánovania paralelizovaním.

Tab. 11: Zhrnutie výsledkov paralelizovania na celkový čas preplánovania s vynúteným prepojovaním.

	počet uzlov [-]	min [ms]	max [ms]	priemer [ms]	zrýchlenie [-]
Neparalelizovaná	4000	31	56	39,43	-
Paralelizovaná	4000	13	28	19,68	2,01x
Neparalelizovaná	8000	61	87	72,9	-
Paralelizovaná	8000	30	47	37,92	1,92x
Neparalelizovaná	16000	109	152	130,76	-
Paralelizovaná	16000	50	74	62,27	2,09x

Tab. 12: Zhrnutie výsledkov paralelizovania na celkový čas preplánovania.

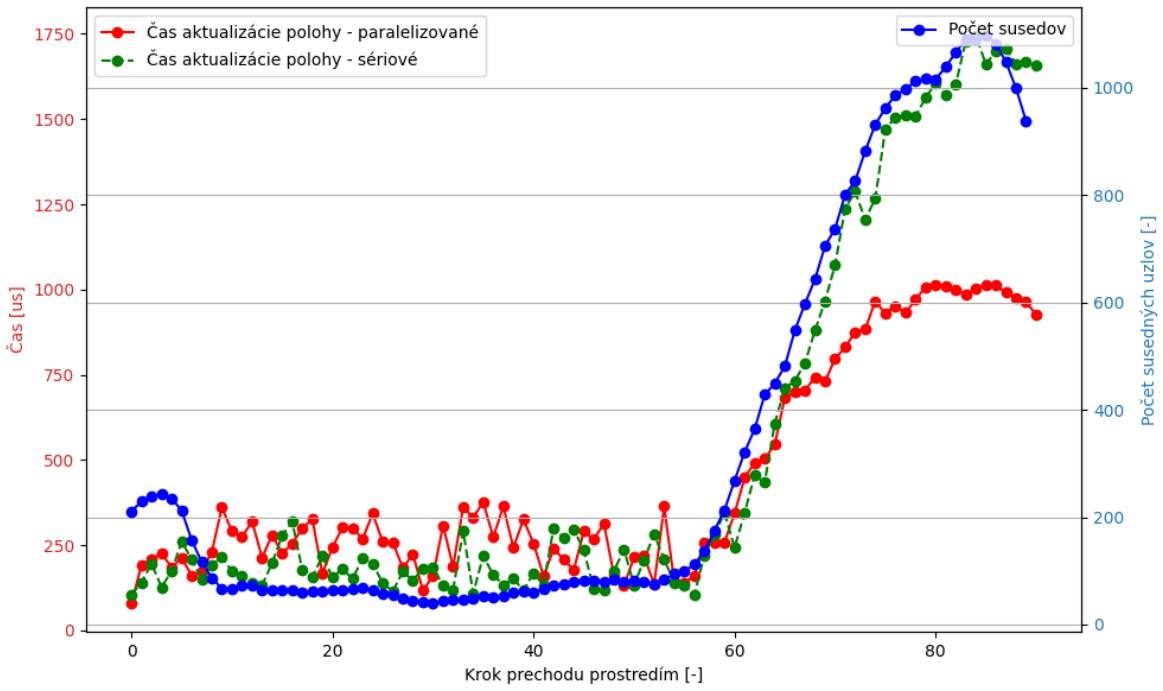
	počet uzlov [-]	min [ms]	max [ms]	priemer [ms]	zrýchlenie [-]
Neparalelizovaná	4000	9	22	12, 82	-
Paralelizovaná	4000	5	11	7, 32	1,75x
Neparalelizovaná	8000	25	40	31,61	-
Paralelizovaná	8000	12	24	16,82	1,88x
Neparalelizovaná	16000	55	87	69,92	-
Paralelizovaná	16000	28	40	34,47	2,02x

3.2.8 Pohyb robota – zmena cieľového uzla stromu

Na obrázku obr. 3.9 je vyobrazený priebeh dĺžky trvania aktualizácie cieľového stavu stromu vzhľadom na aktuálnu polohu robota pri maximálnom množstve uzlov stromu (N = 4000).

Os „x“ predstavuje krok UAV prostredím počas sledovania cesty, teda aktualizovanie polohy UAV v strome. Na osi „y“ sú vyobrazené časy aktualizovania cieľového stavu stromu paralelizovanou (červená) a sériovou verziou algoritmu (zelená) a množstvo susedných uzlov nového uzla reprezentujúceho aktuálnu polohu UAV (modrá).

V rámci tabuľky tab. 13 sú zhrnuté výsledné zrýchlenia algoritmu vplyvom paralelizovania – minimálne, maximálne a priemerné zrýchlenie sériovej verzie algoritmu.



Obrázok 3.9: Priebeh doby trvania aktualizovania polohy robota v rámci stromovej štruktúry plánovacieho algoritmu (N = 4000).

Tab. 13: Zrýchlenie algoritmu aktualizácie aktuálnej polohy robota v stromovej štruktúre v krokoch s vyšším množstvom susedných uzlov nového uzla.

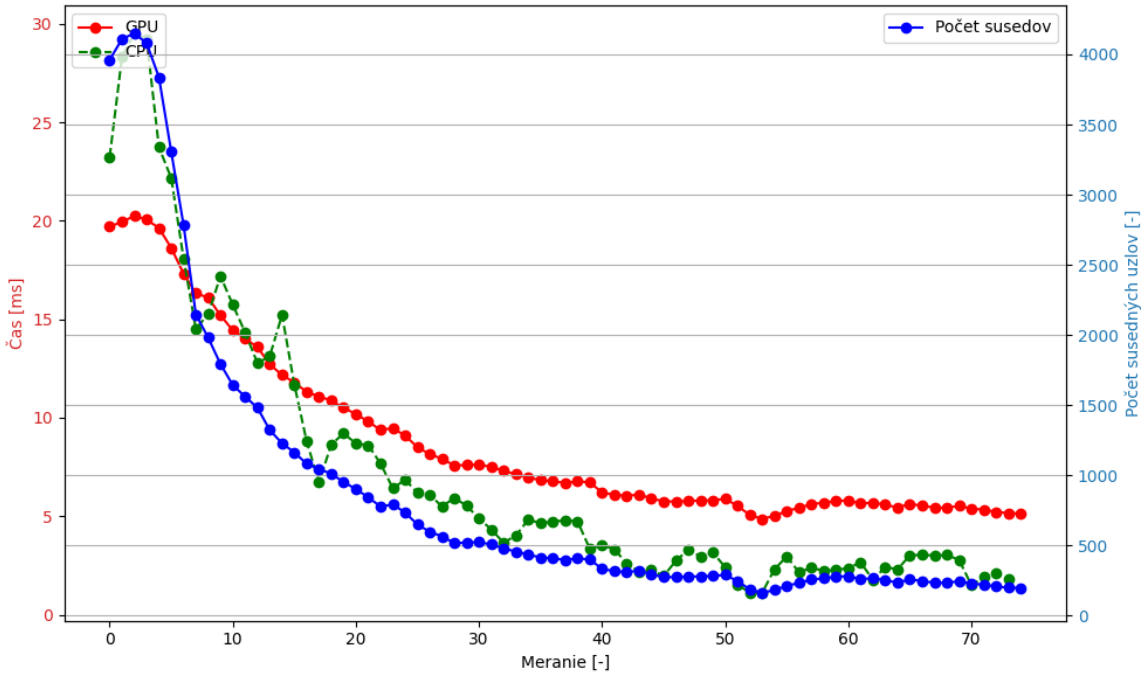
	zrýchlenie [-]		
počet uzlov	min	max	priemer
4000	1,04x	1,79x	1,50x
8000	1,01x	1,98x	1,61x
16000	1,06x	2,00x	1,70x

3.2.9 Zmena cieľa plánovania – pohyb koreňa stromu

Jednotlivé experimenty boli vykonané tak, že po naplánovaní cesty UAV nezačalo prechod prostredím ale koreň stromu sa začal po naplánovanej ceste posúvať smerom k aktuálnej

pozícii UAV. Vykreslené priebehy predstavujú časy potrebné na posun koreňa stromu (červená, zelená čiara) a počet susedných uzlov nového koreňa (modrá čiara). Na obr. 3.10 je vyobrazené porovnanie paralelizovania algoritmu s využitím CPU (zelená) a GPU (červená).

V rámci tabuľky tab. 14 sú zhrnuté výsledné zrýchlenia algoritmu vplyvom paralelizovania – minimálne, maximálne a priemerné zrýchlenie sériovej verzie algoritmu. V poslednom riadku tabuľky sú uvedené zrýchlenia algoritmu dosiahnuté využitím GPU, kedy maximálny počet uzlov v strome N = 16000. Uvedené je maximálne zrýchlenie dosiahnuté využitím GPU v porovnaní s CPU paralelizovanou verziou.



Obrázok 3.10: Priebeh času potrebného pre posun koreňa stromu algoritmom paralelizovaným na CPU a GPU. (N = 16000)

Tab. 14: Zrýchlenie algoritmu aktualizácie aktuálnej polohy robota v stromovej štruktúre v krokoch s vyšším množstvom susedných uzlov nového uzla.

	zrýchlenie [-]		
počet uzlov	min	max	priemer
4000	1,04x	1,79x	1,50x
8000	1,01x	1,98x	1,61x
16000	1,06x	2,00x	1,70x

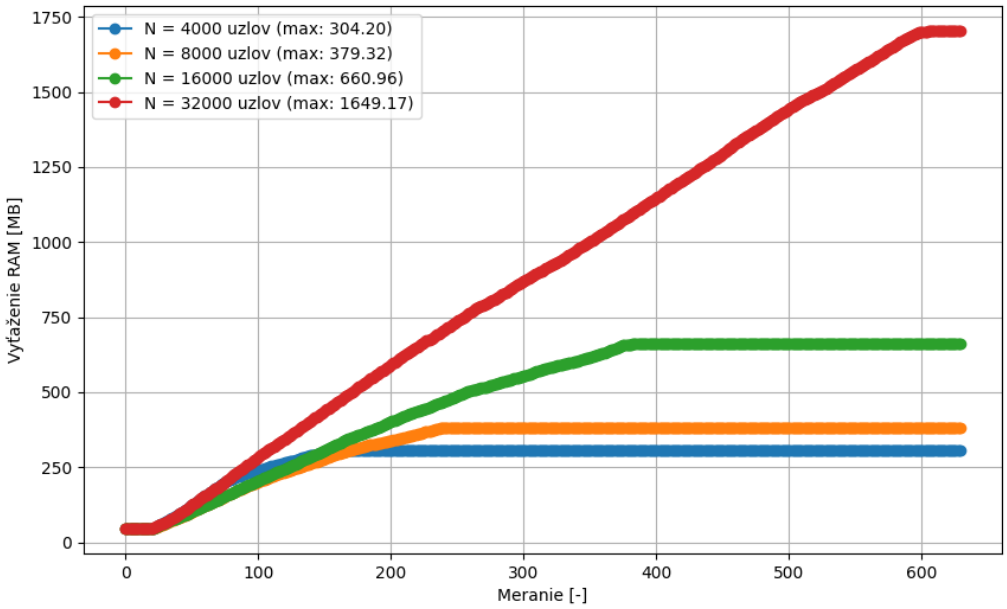
3.3 Priebežná optimalizácia cesty počas prechodu robota prostredím

V tabuľke tab. 15 sú zhrnuté výsledky vykonaných experimentov: priemerná dĺžka pôvodnej cesty (d1) a prejdenej cesty (d2), minimálne, maximálne a priemerné zlepšenie (skrátenie) dĺžky cesty. V prípade experimentov s maximálnym počtom uzlov N = 8000 nie je výrazný rozdiel aj napriek tomu, že po uplynutí času (2,5s) statickej fázy plánovania strom ešte neobsahoval maximálny počet uzlov. Maximálny počet uzlov bol dosiahnutý počas prechodu prostredím.

Tab. 15: Zlepšenie dĺžky cesty počas prechodu robota prostredím v porovnaní s pôvodne plánovanou cestou

	d1[m]	d2[m]	N	t [s]	min [%]	max [%]	priemer[%]
Cesta 1	83,63	80,68	8000	2,5	0,21	11,66	3,50
Cesta 2	83,97	80,76	8000	5	0,07	9,76	3,78
Cesta 3	94,67	86,35	4000	5	0,06	35,73	8,06

3.4 Využitie pamäte RAM – obmedzenie počtu uzlov



Obrázok 3.11: Porovnanie využitia pamäte RAM procesom plánovača.

Na obrázku obr. 3.11 je vyobrazené využitie pamäte RAM procesom plánovača počas statickej fázy plánovania a prechodu prostredím pri rôznych obmedzeniach

maximálneho počtu uzlov stromu. Po dosiahnutí maximálneho počtu uzlov stromu sa množstvo využitej pamäte procesom plánovača mení zanedbateľne.

Záver

V súčasnosti existuje množstvo variácii RRT algoritmu, ktoré sú využité na riešenie širokej škály plánovacích úloh. Online plánovaniu a plánovaniu v čiastočne známom prostredí sa z nich venuje menšia skupina. Jednotlivé algoritmy sú v rámci publikácii testované na výkonnejších zariadeniach (stolových počítačoch), ktoré si menší autonómny mobilný robot ako je napr. UAV nemôže dovoliť nosiť na palube. Kvôli tomu je náročne rozhodnúť, či je algoritmus možné nasadiť na palubný počítači autonómneho UAV. Hlavnou motiváciou tejto práce je nasadenie autonómneho UAV, ktoré vykonáva inšpekčné úlohy v skladovom priestore. Skladový priestor je dobre štruktúrované prostredie, v ktorom sa ale môže vyskytnúť vopred neznáma prekážka kolidujúca s cestou prechodu prostredím, ktorú má pre danú inšpekčnú úlohy UAV vopred naplánovanú. Úlohou tejto práce je, aby autonómne UAV vykonávajúce inšpekčnú úlohu bolo schopné upraviť už vopred naplánovanú cestu v prípade hroziacej kolízie s vopred neznámou prekážkou.

Jedným z hlavných cieľov tejto práce je návrh plánovacieho algoritmu, ktorý dokáže riešiť takto vzniknuté kolízne situácie. Navrhnutý algoritmus musí pre autonómne UAV naplánovať cestu z bodu A do bodu B, kontrolovať priebeh prechodu UAV prostredím a v prípade hroziacej kolízie sledovanú cestu upraviť. Algoritmus navrhnutý a implementovaný v rámci tejto práce vychádza z online plánovacieho algoritmu RRT^X, ktorého rozšírenie a modifikácia je jedným z hlavných prínosov tejto práce. Dôraz je kladený na dynamickú fázu plánovania, kedy počas prechodu UAV prostredím môže nastať kolízna situácia. Navrhnutý plánovací algoritmus v prípade prerušenia cesty vopred neznámou prekážkou nezačne proces plánovania od začiatku ale pokúsi sa prerušenú cestu rekonštruovať. Modifikáciou procesu rekonštrukcie cesty, sa navrhnutý plánovací algoritmus v prvom kroku pokúsi rýchlo pripojiť odstrihnutú časť cesty k stromu. Ak vopred neznáma prekážka neblokuje cestu v časti úzkeho prechodu alebo neodreže všetky uzly stromu v r okolí uzlov odstrihnutej časti cesty, je možné odstrihnutý úsek cesty cez niektorý z jeho uzlov rýchlo pripojiť k stromu. Takto sa algoritmus môže vyhnúť druhému kroku preplánovania, ktorý je v porovnaní s rutinou rýchleho prepojenia výpočtovo náročnejšia.

V prípade neúspechu rutiny rýchleho pripojenia odstrihnutej časti cesty k stromu je vykonaný druhý implementovaný krok preplánovania – vynútené prepojovanie uzlov. Tento

proces spočíva v postupnom prepojovaní odstrihnutých uzlov stromu cez ich validné susedné uzly, ktoré nestratil pripojenie k stromu. Proces prepojovania je smerovaný od validných susedov k uzlu predstavujúceho aktuálnu polohu UAV a je ukončený opätovným pripojením niektorého uzla odstrihnutej časti cesty k stromu.

Popri implementovaných modifikáciách procesu preplánovania cesty je navrhnutý plánovací algoritmus rozšírený o priebežné optimalizovanie sledovanej cesty a možnosť upravovať cieľ danej plánovacej úlohy. Priebežná optimalizácia cesty sa vykonáva počas aktualizácie polohy UAV v stromovej štruktúre plánovacieho algoritmu. Vplyv priebežnej optimalizácie na výslednú dĺžku prejdenej cesty v porovnaní s dĺžkou pôvodne plánovanej cesty je zhrnutý v kapitole 3.3.

Navrhnutý plánovací algoritmus je možne označiť ako online algoritmus nie len vďaka jeho schopnosti upraviť sledovanú cestu vplyvom nečakanej zmeny prostredia ale aj schopnosti prispôsobiť sa zmene cieľa plánovania. Stromová štruktúra vytváraná navrhnutým plánovacím algoritmom sa rozrastá smerom z cieľového stavu UAV k jeho počiatočnému. Vďaka takémuto usporiadaniu je jednoduché manipulovať s uzlom stromu, ktorý predstavuje aktuálnu polohu UAV. To neplatí v prípade manipulácie s koreňom stromu, keďže táto operácia ma vplyv na celú stromovú štruktúru. Ako sa ukázalo v rámci experimentov v kapitole 3.2.9, proces manipulácie s koreňom stromu je v dôsledku množstva kontrol kolízii so susednými uzlami výpočtovo náročný. Navrhnutý plánovací algoritmus je preto doplnený o implementáciu funkcie, ktorá efektívne manipuluje s koreňom stromu a umožňuje aktualizovať cieľ plánovania. Toto je jedným z praktických prínosov plánovacieho algoritmu, kedy v prípade potreby jednoduchej úpravy cieľa už naplánovanej a sledovanej inšpekčnej cesty nie je nutné zastaviť UAV začať nové plánovanie.

Autonómne UAV vykonáva množstvo výpočtových operácii na palubnom počítači pričom jeho výkon je v porovnaní so stolovým počítačom obmedzený. Aby navrhnutý plánovací algoritmus dokázal rýchlo reagovať na zmeny v prostredí alebo zadaní plánovacej úlohy, musí efektívne využívať hardvér, na ktorom sa plánovanie vykonáva. Jedným z hlavných prínosnou tejto práce je efektívne využitie hardvéru osadeného na UAV paralelizovaní plánovacieho algoritmu. V rámci kapitoly 2 sú pomenované časti plánovacieho algoritmu, ktoré je možné a zároveň vhodné paralelizovať. Okrem toho sú v rámci tejto práce predstavené nutné úpravy častí plánovacieho algoritmu tak, aby ich bolo možné efektívne paralelizovať distribuovaním výpočtov na viaceré vlákna CPU. Vďaka paralelizovaniu vykonaných výpočtov je výrazne skrátený čas potrebný na preplánovanie

cesty alebo aktualizovanie cieľa plánovania. Vplyv paralelizovania na vybrané časti plánovacieho algoritmu ako aj na celkový čas preplánovania cesty a zmeny cieľa plánovania sú zhrnuté v rámci kapitoly 3.2.

Popri výkone CPU je obmedzené aj množstvo operačnej pamäte RAM palubného počítača. Preto je potrebné zohľadniť množstvo pamäte RAM využitej plánovacím algoritmom. Ďalším prínosom tejto práce je modifikácia plánovacieho algoritmu, ktorou je obmedzenie maximálneho množstva uzlov vytváraného stromu, čím je výrazne obmedzené maximálne množstvo využitej pamäte RAM procesom plánovacieho algoritmu. Po dosiahnutí maximálneho počtu uzlov je odstrihnutá a vymazaná časť okrajových uzlov stromu (uzly bez potomkov) tak, aby sa vytváraná stromová štruktúra mohlo priebežne rozrastať a neustále prehľadávať priestor. Proces rozrastania a orezávania stromu prebieha počas celej doby plánovania. Množstvo využitej pamäte RAM procesom plánovania v závislosti od maximálneho množstva uzlov stromu je zhrnuté v rámci meraní kapitoly 3.4.

Dostupný palubný počítač UAV (NVIDIA Jetson Xavier NX) má okrem CPU s viacerými vláknami k dispozícii grafický procesor pre hardvérové akcelerovanie výpočtov. Podobne ako v prípade paralelizovania plánovacieho algoritmu na CPU sú v rámci kapitoly 2 označené časti plánovacieho algoritmu vhodné pre hardvérovú akceleráciu výpočtov na GPU. Ďalším prínosom tejto práce je identifikovanie a dodatočné zrýchlenie vybraných výpočtovo náročných častí plánovacieho algoritmu s využitím hardvérovej akcelerácie. Vplyv hardvérovej akcelerácie na čas vykonávania akcelerovaných častí algoritmu je vyhodnotený v rámci kapitol 3.2.2 a 3.2.9.

Snahou tejto práce je orientovanie sa na jej praktike využitie. Preto je posledným prínosom tejto práce samotná implementácia, nasadenie navrhnutého plánovacieho algoritmu a vykonanie všetkých experimentov z kapitoly 3 na hardvéry palubného počítača UAV. Vďaka tomu je na základe dosiahnutých výsledkov možné posúdiť praktickú využiteľnosť algoritmu pri jeho nasadení na autonómnom UAV. Implementácia plánovacieho algoritmus je z praktických dôvodov zaobalená do ROS balíčka vďaka čomu je možná jednoduchá interakcia s plánovacím algoritmom. To do značnej mieri uľahčuje získavanie dát počas plánovania, vizualizovanie, simulovanie, konfigurovanie plánovača a do budúcnosti má potenciál jednoduchšej integrácie do existujúceho systému využívajúceho ROS.

Literatúra

[1] Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

[2] LaValle, S. M. (2006). Planning Algorithms. Cambridge University Press.

[3] Noreen, I., Khan, A., Habib, Z. (2016). Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions. *International Journal of Advanced Computer Science and Applications*. 7. 10.14569/IJACSA.2016.071114.

[4] Karaman, S., Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning, Int J Rob Res, vol. 30, pp. 846-894.

[5] Naderi, K., Rajamäki, J., & Hämäläinen, P. (2015). *RT-RRT*: a real-time path planning algorithm based on RRT*.* 113–118. https://doi.org/10.1145/2822013.2822036

[6] Wurm, K., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W. (2010). OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. 2.

[7] Yuncheng, L., Zhucun X., Gui-Song X., Liangpei, Z. (2018). A survey on vision-based UAV navigation, Geo-spatial Information Science, 21:1, 21-32

[8] Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., Teller, S. (2011). Anytime Motion Planning using the RRT*. *2011 IEEE International Conference on Robotics and Automation*.

[9] Gammell, J., Srinivasa, S., Barfoot, T. (2014). Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2997–3004.

[10] Otte, M., & Frazzoli, E. (2015). RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, *35*. https://doi.org/10.1177/0278364915594679

[11] Chandler, B., & Goodrich, M. A. (2017). Online RRT* and online FMT*: Rapid replanning with dynamic cost. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6313–6318. https://doi.org/10.1109/IROS.2017.8206535

[12] Adiyatov, O., & Varol, H. A. (2017). A novel RRT*-based algorithm for motion planning in Dynamic environments. *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 1416–1421. https://doi.org/10.1109/ICMA.2017.8016024

[13] Şucan, I. A., Moll, M., Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4), 72–82. https://doi.org/10.1109/MRA.2012.2205651

[14] Pan, J., Chitta, S., Manocha, D. (2012). FCL: A general purpose library for collision and proximity queries. *2012 IEEE International Conference on Robotics and Automation*, 3859–3866. https://doi.org/10.1109/ICRA.2012.6225337

[15] Rusu, R. B., & Cousins, S. (2011, May 9). 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*.

[16] Chandra, R., Dagum, L., Kohr, D., Menon, R., Maydan, D., & McDonald, J. (2001). *Parallel programming in OpenMP*. Morgan kaufmann.

Zoznam publikačnej činnosti autora

V2 Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka

V2_01 BABINEC, Andrej - RODINA, Jozef - MRÁZ, Eduard - STANKO, Jaromír. Using RTAB-Map as a 3D mapping tool for drones. In *ELITECH´20 [elektronický zdroj] : 22nd Conference of doctoral students. Bratislava, Slovakia. May 27, 2020*. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2020, [5] s. ISBN 978-80-227-5001-1. Kategória publikácie do 2021: AFD

V2_02 STANKO, Jaromír - HUBINSKÝ, Peter - RODINA, Jozef. Intelligent BLDC motor controller with embedded PLC like module. In *ELITECH´19 [elektronický zdroj] : 21st Conference of doctoral students. Bratislava, Slovakia. May 29, 2019*. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2019, CD-ROM, [6] s. ISBN 978-80-227-4915-2. Kategória publikácie do 2021: AFD

V2_03 STANKO, Jaromír - RODINA, Jozef - HUBINSKÝ, Peter. Comparison of approaches for UAV dynamics consideration in sampling based path planning methods. In *ISMCR 2020 : 23rd International Symposium on Measurement and Control in Robotics. Budapest, Hungary. October 15-17, 2020*. Piscataway : IEEE, 2020, [5] s. ISBN 978-0-7381-4269-

2. V databáze: IEEE: 9263768 ; DOI: 10.1109/ISMCR51255.2020.9263768 ; SCOPUS: 2-s2.0-85099541640 ; WOS: 000833324500033. Kategória publikácie do 2021: AFC

V2_04 STANKO, Jaromír - ŠTEC, Filip - PALKOVIČ, Lukáš - RODINA, Jozef - RAU, Dávid. Towards automatic inventory checking using an autonomous unmanned aerial vehicle. In *ETFA 2022 : 27th International Conference on Emerging Technologies and Factory Automation. Stuttgart, Germany. September 6-9, 2022*. Danvers : IEEE, 2022, [8] s. ISBN 978-1-6654-9996-5. V databáze: DOI: 10.1109/ETFA52439.2022.9921460 ; WOS: 000934103900038 ; SCOPUS: 2-s2.0-85141410119 ; IEEE: 9921460. Typ výstupu: príspevok z podujatia; Výstup: zahraničný; Kategória publikácie do 2021: AFC

V3 Vedecký výstup publikačnej činnosti z časopisu

V3_01 STANKO, Jaromír - RODINA, Jozef - HUBINSKÝ, Peter. Inteligentný menič so vstavanými PLC pre BVDC motory. In *ATP Journal plus : Výskum v kybernetike na FEI STU v Bratislave*. č. 2 (2019), s. 41-44. ISSN 1336-5010. Kategória publikácie do 2021: ADF

V3_02 STANKO, Jaromír - ŠTEC, Filip - RODINA, Jozef. Process automation of warehouse inspection using an autonomous unmanned aerial vehicle. In *MM Science Journal*. October (2022), s. 5864-5869. ISSN 1803-1269(P) (2022: 0.7 - IF, 0.239 - SJR, Q3 - SJR Best Q). V databáze: DOI: 10.17973/MMSJ.2022_10_2022063 ; WOS: 000860510800001 ; SCOPUS:2-s2.0-85139174041.
Typ výstupu: článok; Výstup: zahraničný; Kategória publikácie do 2021: ADM

Riešené vedecko-výskumné projekty

Riadenie autonómneho lietajúceho prostriedku v neznámom vnútornom prostredí. (Mladý výskumník)

Algoritmus kolektívnej inteligencie: Interdisciplinárne štúdium swarmového správania netopierov. (APVV-17-0116)

Kolaboratívny robot pre použitie v laboratóriu. (APVV-17-0214)

Navigácia autonómneho vozidla v interiéri s použitím umelej inteligencie. (Program na podporu excelentných tímov mladých výskumníkov)

Navigačný stack pre autonómne drony v priemyselnom prostredí. (APVV-21-0352)

Robustná lokalizácia pre drony v priemysle 4.0. (VEGA 1/0599/20)

Výskum a vývoj využiteľnosti autonómnych lietajúcich prostriedkov v boji proti pandémii spôsobenej COVID-19. (UAVLIFE, OP VaI - 313011ATR9)

Štatistika: kategória publikačnej činnosti od 2022

V2	Vedecký výstup publikačnej činnosti ako časť editovanej knihy alebo zborníka	4
V3	Vedecký výstup publikačnej činnosti z časopisu	2
Súčet		**6**

Štatistika: kategória ohlasov od 2022

1 Citácia v publikácii registrovaná v citačných indexoch				**1**
	Zahraničné		1	
Súčet				**1**