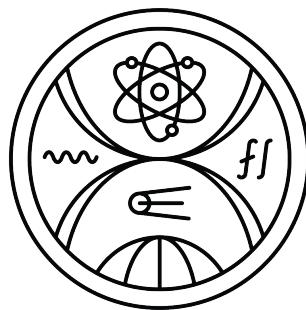


Univerzita Komenského v Bratislave  
Fakulta matematiky, fyziky a informatiky

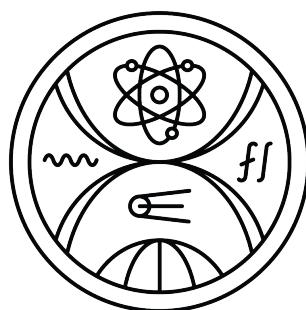


# ADAPTÍVNE RIADENIE KROKU ŠTVORNOHÉHO ROBOTA

Diplomová práca



Univerzita Komenského v Bratislave  
Fakulta matematiky, fyziky a informatiky



# ADAPTÍVNE RIADENIE KROKU ŠTVORNOHÉHO ROBOTA

Diplomová práca

Study program: aplikovaná informatika  
Branch of study: informatika  
Department: Katedra aplikovanej informatiky  
Supervisor: prof. RNDr. Roman Ďuríkovič, PhD.  
Consultant: Mgr. Rastislav Marko





## ZADANIE ZÁVEREČNEJ PRÁCE

- Meno a priezvisko študenta:** Bc. Zuzana Mačicová  
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
- Študijný odbor:** informatika  
**Typ záverečnej práce:** diplomová  
**Jazyk záverečnej práce:** slovenský  
**Sekundárny jazyk:** anglický
- Názov:** Adaptívne riadenie kroku štvornohého robota  
*Adaptive step control of a biped robot*
- Anotácia:** Adaptívne riadenie je v súčasnosti jedným z najpopulárnejších prístupov pre ovládanie kroku štvornohých robotov. V porovnaní s konvenčnejšími metódami dokáže generovať vysoko dynamické pohyby a zároveň zabezpečiť stabilitu a robustnosť kroku. Napriek neustále zvyšujúcemu sa výkonu mikropočítačov a vývoju stále efektívnejších optimalizačných riešičkov je však problém kráčania stále komplexný a jeho úspešné formulácie sa spoliehajú na mnohé zjednodušenia a zanedbania skutočných vlastností riadeného systému. Práve efektívne formulácie zohľadňujúce čo možno najviac relevantných vplyvov na krok sú v súčasnosti predmetom výskumu.
- Ciel:** Cieľom práce je analyzovať a navrhnúť model adaptívneho riadenia pre štvornohého robota operujúceho v zložitom prostredí. Táto téma bude vypracovaná v spolupráci s firmou Panza Robotics s.r.o.
1. Naštudujte problematiku riadenia kroku kráčajúcich robotov.
  2. Analyzujte možnosti matematických formulácií problému optimálneho riadenia kroku.
  3. Navrhnite a implementujte adaptívne riadenie pre štvornohého robota.
  4. Overte navrhnuté riešenie pri rôznych podmienkach v simulačnom prostredí.
- Literatúra:** Ding, Yanran & Pandala, Abhishek & Park, Hae-Won. (2019). Real-time Model Predictive Control for Versatile Dynamic Motions in Quadrupedal Robots. 10.1109/ICRA.2019.8793669.  
Grandia, R., Taylor, A. J., Ames, A. D., & Hutter, M. (2021, May). Multi-layered safety for legged robots via control barrier functions and model predictive control. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 8352-8358). IEEE.  
Carlo, Jared & Wensing, Patrick & Katz, Benjamin & Bledt, Gerardo & Kim, Sangbae. (2018). Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. 1-9.



Univerzita Komenského v Bratislavе  
Fakulta matematiky, fyziky a informatiky

---

10.1109/IROS.2018.8594448.

**Vedúci:** prof. RNDr. Roman Ďuríkovič, PhD.

**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky

**Vedúci katedry:** prof. Ing. Igor Farkaš, Dr.

**Dátum zadania:** 29.11.2022

**Dátum schválenia:** 29.11.2022

prof. RNDr. Roman Ďuríkovič, PhD.

garant študijného programu

.....  
študent

.....  
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracovala samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa a konzultanta.

Bratislava, 2024

Bc. Zuzana Mačicová



# Pod'akovanie

TODO

# Abstrakt

TODO

**Kľúčové slová:** TODO

# Abstract

TODO

**Keywords:** TODO

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Prehľad problematiky</b>	<b>2</b>
1.1 Riadenie u iných robotov . . . . .	5
1.1.1 ANYmal . . . . .	6
1.1.2 Cheetah . . . . .	7
1.1.3 Solo . . . . .	8
1.1.4 CHAMP . . . . .	9
1.1.5 Quad SDK . . . . .	10
1.2 Diskusia . . . . .	11
<b>2 Implementácia</b>	<b>12</b>
2.1 Prehľad technológií . . . . .	12
2.2 Popis Artabana . . . . .	14
2.3 Inverzná dynamika robota . . . . .	14
<b>3 Výsledky</b>	<b>16</b>
3.1 Priame pozičné ovládanie . . . . .	16
3.2 Quad SDK bez inverznej dynamiky . . . . .	17
3.3 Quad SDK s inverznou dynamikou predných nôh . . . . .	17

# Zoznam obrázkov

1.1	Príklad riadiaceho procesu robota. Najprv sa spracuje signál z kamier a zoberie sa príkaz od používateľa, kam má robot ísť. Potom sa vytvorí prvotný plán v High-level control, ktorý sa potom pošle Low-level control, ktorý komunikuje priamo s hardvérom robota. . . . .	3
1.2	Typ kroku štvornohých zvierat - chôdza krokom. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Nohy v kontakte sú vyznačené zelenou farbou, bielou je vyznačená noha, ktorá sa podložky nedotýka. Značenie nôh je nasledovné: 1 - ľavá predná, 2 - pravá predná, 3 - ľavá zadná, 4 - pravá zadná. . . . .	4
1.3	Typ kroku štvornohých zvierat - klus. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na obr.1.2. . . .	4
1.4	Typ kroku štvornohých zvierat - cval. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na obr.1.2. . .	4
1.5	Schéma MPC. Na obrázku vidíme ako v danom časovom horizonte sa mení kontrolný signál na základe nameraného stavu a predikovaného stavu [2]. . . . .	5
1.6	ANYmal [8]. . . . .	6
1.7	Riadiaci proces ANYmal. Schéma procesu riadenia robota ANYmal [21].	6
1.8	MIT Cheetah 3 [7]. . . . .	7
1.9	Schéma riadenia robota Cheetah [17]. . . . .	8
1.10	Robot Solo [15]. Robot Solo má schopnosť dostať sa do rôznych konfigurácií so svojimi nohami, vďaka tomu, že má veľký rozsah kĺbov. . . .	8
1.11	Proces riadenia robota Solo [25]. . . . .	9
1.12	Proces riadenia robota vo frameworku CHAMP[19]. . . . .	9
1.13	Štruktúra Quad SDK. Hierarchická štruktúra Quad SDK, ktorá má 3 hlavné časti: Global Planner, Local Planner a Robot Driver [23]. . . .	10
2.1	Štruktúra ROS balíčku [16]. . . . .	12
3.1	Zobrazanie zadnej nohy a jej trajektórie. . . . .	16

3.2	Porovnanie plánovanej trajektórie nohy so skutočnou. Oranžovou a sivou vidíme skutočnú a plánovanú trajektóriu ľavej prednej nohy v simulácii. Čierrou je znázornená trajektória tela. . . . .	17
3.3	Porovnanie plánovanej trajektórie nohy so skutočnou. Oranžovou a sivou vidíme skutočnú a plánovanú trajektóriu ľavej prednej nohy v simulácii. Čierrou je znázornená trajektória tela. . . . .	18

# Zoznam tabuliek



# Úvod

# Kapitola 1

## Prehľad problematiky

Oblast' robotiky v posledných rokoch zažíva veľký rozmach. Či už v oblasti priemyselnej výroby, zdravotníctva, pri prieskume oblastí nebezpečných pre človeka alebo v domácnosti. Veľa z týchto robotov vykonáva monotónne úlohy, avšak pri kráčajúcich robotoch sú ich úlohy menej opakujúce sa a je nevyhnutnosťou aby sa robot vedel prispôsobiť rôznym situáciám, v ktorých sa môže ocitnúť. Tu vzniká potreba adaptívneho riadenia takýchto robotov.

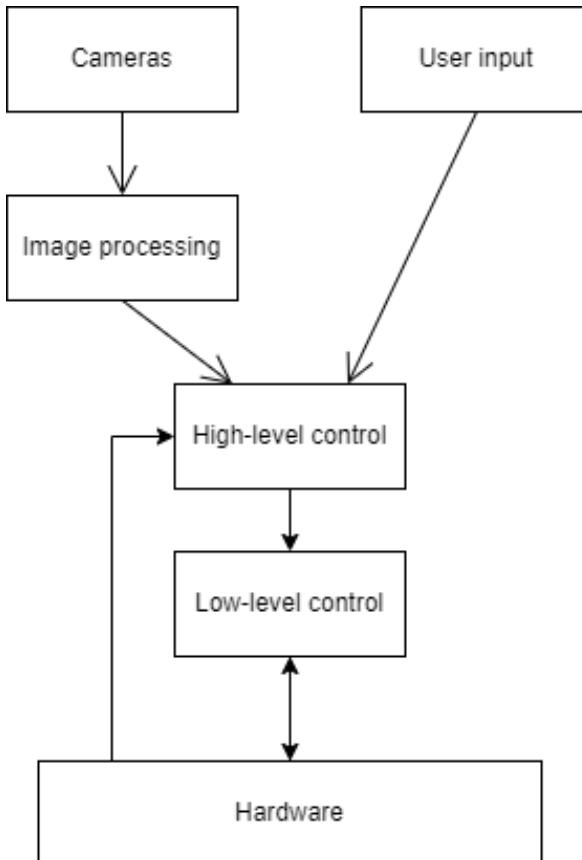
Pri chodiacich robotoch vidíme hlavne viacero štvornohých robotov, ktorých výhodou je napríklad väčšia stabilita oproti dvojnohým robotom. Tieto roboty umožňujú pohyb po členitom teréne, či už sú to schody vo výrobnej hale, nerovný terén v lese alebo prekážka ktorú je potrebné prekročiť na inak rovnej ploche. Pre roboty na pásoch či kolesách je takáto prekážka neprekonateľná aj keď na rovnom povrchu majú veľa výhod ako nenáročné riadenie a jednoduchšiu konštrukciu.

Riadenie štvornohých robotov zahrňa veľa aspektov. Najprv musí robot poznať svoje okolie. To zahŕňa spracovanie signálu z kamery alebo kamier, kedy si vytvorí vnútornú reprezentáciu svojho okolia [21] [8] [14]. Uzazujze sa, že roboty, ktorých ovládací proces nezahŕňa okolité prostredie, nezvládajú chôdzu v teréne kde sú napríklad diery v podložke [25]. Ďalej musí poznať svoj stav - kde sa nachádza v rámci prostredia, ako je natočený a kde sa nachádzajú jeho nohy. To typicky vieme vypočítať pomocou priamej kinematiky z natočenia motorov, ktoré získame zo senzorov. Niektoré roboty majú aj senzor v spodnej časti nohy, ktorá sa dotýka podložky, či je noha v kontakte alebo nie [15] [13]. Taktiež musí robot vedieť ktorým smerom má ísť. Tento údaj môže pochádzať z už dopredu naplánovanej cesty alebo priamo ako príkaz od používateľa, ktorý robota ovláda.

Na základe všetkých týchto dát sa počíta aký pohyb má robot vykonávať, napríklad či má ísť dopredu, otáčať sa a akou rýchlosťou. Ďalej sa vypočítajú trajektórie nôh, prípadne sily, ktorými majú jednotlivé nohy pôsobiť. Z nich sa väčšinou vypočítajú krútiace momenty na motoroch a tie sa pošlú do robota. Celý proces riadenia však

musí počítať aj s ďalšími obmedzeniami ako sú napríklad rozsahy nôh a to aby reakčné sily koncových častí nôh sa nachádzali v kuželi trenia a predišlo sa šmýkaniu nôh robota [7].

Tento proces, príklad vidíme na obr.1.1, sa opakuje veľa krát za sekundu a priebežne sa upravuje. Je nevyhnuté aby bol dostatočne rýchly aby vedel reagovať na zmeny a zároveň efektívny aby mohol bežať priamo na počítači v tele robota.



Obr. 1.1: Príklad riadiaceho procesu robota. Najprv sa spracuje signál z kamier a zoberie sa príkaz od používateľa, kam má robot ísť. Potom sa vytvorí prvotný plán v High-level control, ktorý sa potom pošle Low-level control, ktorý komunikuje priamo s hardvérom robota.

Takéto adaptívne riadenie je nevyhnutnosťou pri robotoch s nohami, keďže ich okolie sa môže neustále meniť. Napríklad sa na naplánovanej ceste zjaví prekážka alebo je miesto kam robot stúpil nestabilné.

Ďalším problémom pri riadení robotov sú rozdiely medzi modelom v simulácií a skutočným hardvérom. Hardvérové súčiastky môžu mať nepresné rozmery, inú hmotnosť, opotrebením sa môžu deformovať, prípadne nie sú dostatočne pevné a ohýbajú sa. V systéme môžu vznikať rôzne trenia, s ktorými sme nepočítali, robot môže mať nedostatočný príkon a iné problémy. Toto spôsobuje potrebu validácie riadenia aj na hardvéri a často môžeme dostať iné výsledky ako v simulácií.

Tieto rozdiely sú zodpovedné aj za náročný a dlhý proces ladenia a hľadania chýb,

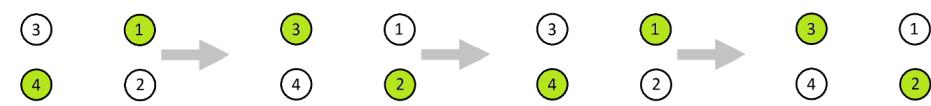
kedže je náročné odizolovať, kde sa daná chyba nachádza. Taktiež do výslednej chôdze vstupuje veľa parametrov, ako dĺžka a výška kroku, výška tela pri chôdzi a rýchlosť chôdze, ktoré je potrebné optimálne nastaviť. Väčšinou tieto parametre vieme nastaviť iba empiricky, skúšaním rôznych kombinácií.

Dôležitý je aj typ chôdze. Štvornohé robony majú inšpiráciu v prírode vo štvornohých zvieratách, to znamená, že aj typ chôdze je im podobný. Typ chôdze vie ovplyvniť plynulosť kroku robota, ale aj stabilitu či rýchlosť. Poznáme niekoľko typov chôdze. Medzi najznámejšie patrí napríklad chôdza krokom, klus a cval.

Pri chôdzi krokom, vid. obr. 1.2, má robot vždy aspoň 3 nohy na podložke, tento typ chôdze je najstabilnejší ale aj väčšinou pomalší oproti zvyšným. Pri kluse, vid. obr. 1.3, sa striedajú vždy dve a dve protiľahlé nohy v kontakte s podložkou. Medzi vystriedaním nôh môže byť ešte fáza letu, kedy nie je žiadna noha v kontakte s podložkou alebo fáza, kedy robota stabilizujeme, kedy sú 3 alebo 4 nohy v kontakte s podložkou. Tento typ chôdze je veľmi populárny pri štvornohých robotoch. Cval je v prírode typicky najrýchlejší typ chôdze. Tesne po sebe sa striedajú predné dve a zadné nohy dve nohy, vid. obr. 1.4. Pri testovaní kroku na robotovi sme sa venovali hlavne chôdzi a klusu.



Obr. 1.2: Typ kroku štvornohých zvierat - chôdza krokom. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Nohy v kontakte sú vyznačené zelenou farbou, bielou je vyznačená noha, ktorá sa podložky nedotýka. Značenie nôh je nasledovné: 1 - ľavá predná, 2 - pravá predná, 3 - ľavá zadná, 4 - pravá zadná.



Obr. 1.3: Typ kroku štvornohých zvierat - klus. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na obr.1.2.



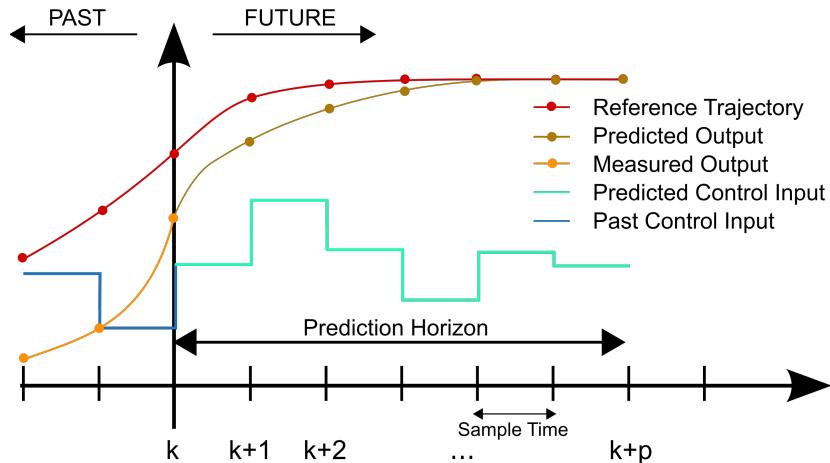
Obr. 1.4: Typ kroku štvornohých zvierat - cval. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na obr.1.2.

## 1.1 Riadenie u iných robotov

Existuje viacero štvornohých robotov, ktoré sú schopné stabilnej chôdze. Niektoré sú vyvíjané univerzitami a sú verejne dostupné spôsoby a metódy akými sú riadené, veľa robotov je však vyvíjaných súkromnými firmami, ktoré svoj výskum v tejto oblasti nepublikujú.

Pravdepodobne najznámejším štvornohým robotom je Spot od Boston Dynamics [6]. Z videí je vidno, že tento robot má dobrú stabilitu a vyladenú chôdzu a zvláda aj zložitý terén ako napríklad schody, avšak nie sú verejne dostupné podrobnejšie informácie o tom, ako funguje jeho riadenie.

Princípy riadenia niektorých iných štvornohých robotov, ako napríklad ANYmal a Cheetah, však verejne sú. Základom pre väčšinu je Model Predictive Control (MPC) [7], [10], [17]. Vo všeobecnosti MPC je metóda optimálneho riadenia, kedy sa počíta optimálna akcia, ktorá minimalizuje cost function za splnenia vopred zadefinovaných podmienok pre konečný časový horizont danej veľkosti. Zároveň však prijíma spätnú väzbu zo systému a plán prispôsobuje [4]. Schému MPC môžeme vidieť na obrázku 1.5.

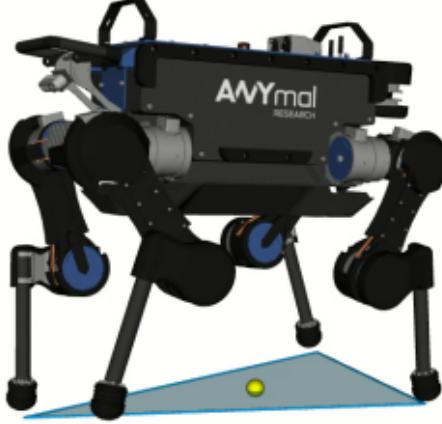


Obr. 1.5: Schéma MPC. Na obrázku vidíme ako v danom časovom horizonte sa mení kontrolný signál na základe nameraného stavu a predikovaného stavu [2].

V robotike sa MPC využíva hlavne na plánovanie cesty, ktorá je pre robota schodná. Výstupom môžu byť priamo krútiace momenty motorov, trajektórie nôh, trajektória tela, sily akými majú jednotlivé nohy pôsobiť alebo kombinácia z už spomenutých. Do MPC väčšinou vstupuje vhodná reprezentácia prostredia, v ktorom sa robot pohybuje, cieľ robota alebo smer akým má ísť a jeho aktuálny stav. Pod stavom robota väčšinou rozumieme polohu a natočenie tela, natočenie motorov, prípadne rýchlosť motorov. Vstupom môže byť aj informácia zo senzorov v koncovkej časti nohy, či je noha robota na podložke alebo nie, takýto senzor však nie každý robot má.

Ďalej si rozoberieme niekoľko robotov a spôsoby ich riadenia, keďže každý robot má iné senzory a stavbu tela, existujú aj rôzne prístupy k ich riadeniu.

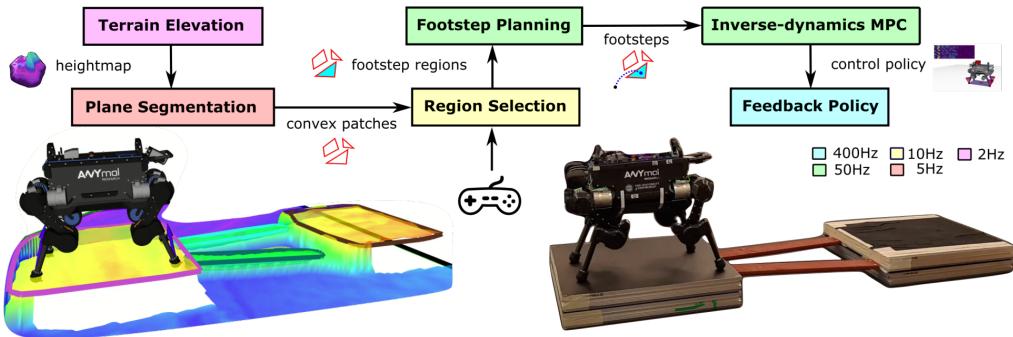
### 1.1.1 ANYmal



Obr. 1.6: ANYmal [8].

Tento štvornohý robot je vyrábaný firmou ANYbotics, obr.1.6. Víe sa pohybovať aj v zložitejšom teréne ako sú schody alebo terén, kde sú diery v podložke a rôzne iné nerovnosti a robot nemôže stúpiť na ľubovoľné miesto na zemi [21].

Na obrázku 1.7 je znázornené jeho riadenie.



Obr. 1.7: Riadiaci proces ANYmal. Schéma procesu riadenia robota ANYmal [21].

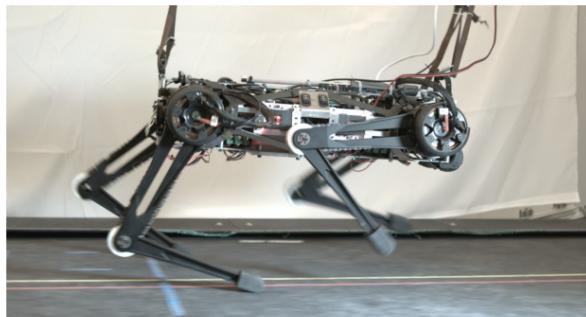
Najprv sa spracuje vstup z kamier, prípadne iných senzorov, v tele robota. Vytvorí sa mapa terénu a z nej sa vypočítajú konvexné plochy, na ktoré môže robot bezpečne stúpiť. Bezpečne stúpiť pre robota znamená, že daná plocha nie je príliš naklonená a zároveň je dostatočne veľká pre koncovú časť nohy. Tieto dátu sa pošlú ďalej do MPC, spolu s pokynom od používateľa ktorým smerom má robot ísiť a jeho aktuálnym stavom (uhly motorov, natočenie tela,...). MPC vypočíta optimálnu trajektóriu ťažiska a trajektórie nôh, ktoré nie sú v kolízii so žiadnym okolitým objektom [21].

MPC využíva knižnicu CROCODDYL (Contact RObot COntrol by Differential DYnamic Library). CROCODDYL je knižnica pre optimálne riadenie robotov, ktorú je možné použiť nielen pre štvornohé roboty [20]. Slúži pre efektívne riadenie robotov

pri viacerých bodoch kontaktu robota s prostredím, napríklad pri štvornohých robotoch je viacero nôh v kontakte s podložkou. Výhodou tejto knižnice je jej verejná dostupnosť a dokumentácia.

Riadiaci softvér ANYmal-a je kompatibilný s ROS, avšak nie je zverejnený v plnom rozsahu v jednotnom framework-u. Sú zverejné iba časti celého riadenia v podobe niektorých algoritmov<sup>1 2 3</sup>.

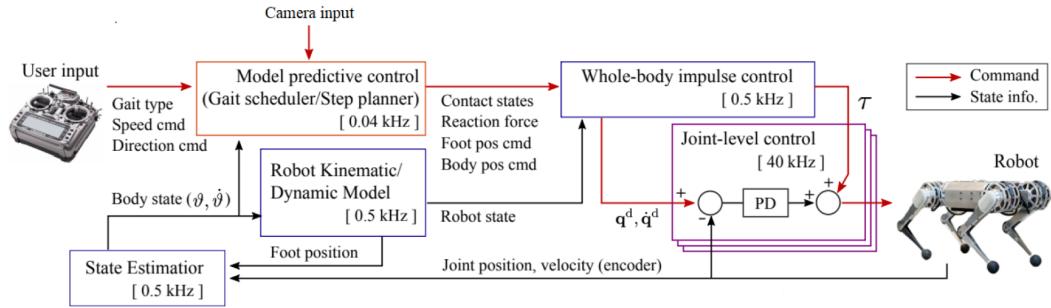
### 1.1.2 Cheetah



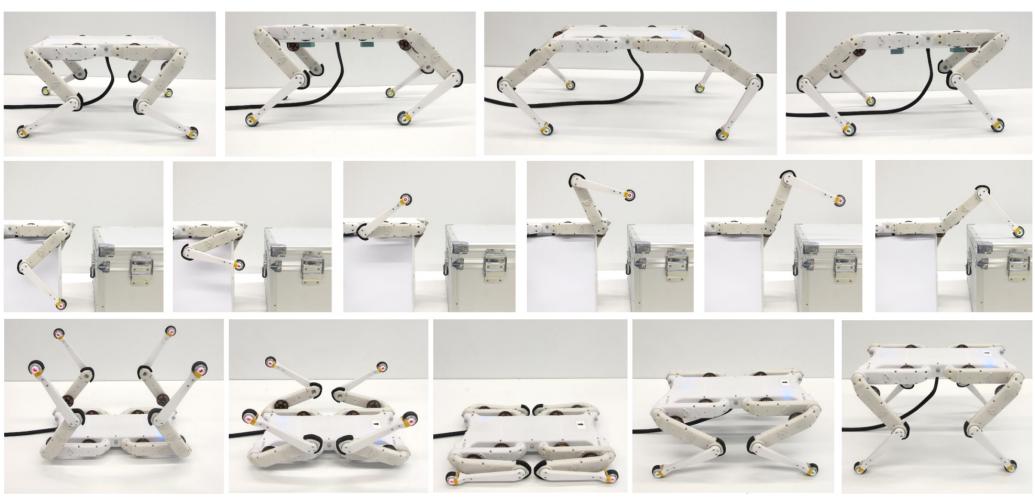
Obr. 1.8: MIT Cheetah 3 [7].

Cheetah 3 je robot vyvíjaný na Massachusetts Institute of Technology (MIT). Ako prvý štvornohý robot zvládol salto dozadu [5]. Má konštrukciu tela typickú pre veľa štvornohých robotov, nohy sa skladajú z 2 segmentov a každá noha má 3 motory. Väčšina jeho váhy je koncentrovaná v tele, keďže má jednoduché a ľahké nohy. Je ovládaný pomocou krútiaceho momentu. Pri riadení tohto robota sa využíva MPC, ktoré určí reakčné sily od podložky pre jednotlivé nohy, vid. obr. 1.9. Do MPC vstupuje zjednodušený model robota, požadovaná rýchlosť, pozícia a natočenie robota. Chýba však spracovanie prostredia, keďže v [7] sa nespomínajú žiadne senzory, ktoré by sníimali okolie robota.

Všetok software je verejne prístupný<sup>4</sup>. Využívajú Linux a je písaný v C++ [9]. Nepoužívajú však Robotic Operating System (ROS), čo je veľká nevýhoda keďže pri našom robotovi už máme väčšinu kódu organizovaná v ROS balíčkoch a vyžadovalo by to veľké dodatočné úpravy.



Obr. 1.9: Schéma riadenia robota Cheetah [17].



Obr. 1.10: Robot Solo [15]. Robot Solo má schopnosť dostať sa do rôznych konfigurácií so svojimi nohami, vďaka tomu, že má veľký rozsah kľbov.

### 1.1.3 Solo

Robot Solo, obr. 1.10, je výsledkom projektu Open Dynamic Robot Initiative, ktorého cieľom bolo vytvoriť neveľmi drahého a jednoduchého štvornohého robota na zstrojenie. Jeho telo sa skladá zo súčiastok, ktoré boli vytlačené na 3D tlačiarni alebo sú bežne dostupné, aj vďaka tomu váži len 2.2 kg. Patrí k robotom, ktoré sú riadené krútiacim momentom. V koncovej časti nôh má senzor, či je noha v kontakte s podložkou [15].

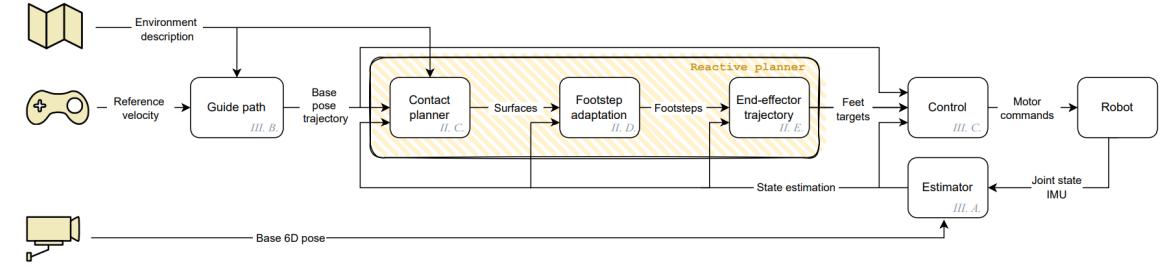
Na obrázku 1.11 vidíme proces riadenia robota Solo. Najprv sa z údajov o prostredí vytvoria konvexné plochy a naplánujú sa kontakty s podložkou, teda kde je vhodné pre robota stúpiť. Tento plán počíta aj s aktuálnym stavom robota a požadovaným smerom robota ako príkazom od operátora. Potom sa vypočítajú trajektórie jednotlivých nôh, reprezentované ako spliny, s ohľadom na vyhnutie sa kolíziám. Časť Control na základe trajektórií jednotlivých nôh pomocou inverznej kinematiky vypočíta požadované natočenie jednotlivých kľbov. Low-level control nakoniec pomocou PD-controllera

<sup>1</sup>[https://github.com/leggedrobotics/ocs2/tree/main/ocs2\\_sqp](https://github.com/leggedrobotics/ocs2/tree/main/ocs2_sqp)

<sup>2</sup>[https://github.com/leggedrobotics/elevation\\_mapping\\_cupy/tree/main](https://github.com/leggedrobotics/elevation_mapping_cupy/tree/main)

<sup>3</sup>[https://github.com/ANYbotics/grid\\_map/tree/master/grid\\_map\\_sdf](https://github.com/ANYbotics/grid_map/tree/master/grid_map_sdf)

<sup>4</sup><https://github.com/mit-biomimetics/Cheetah-Software>

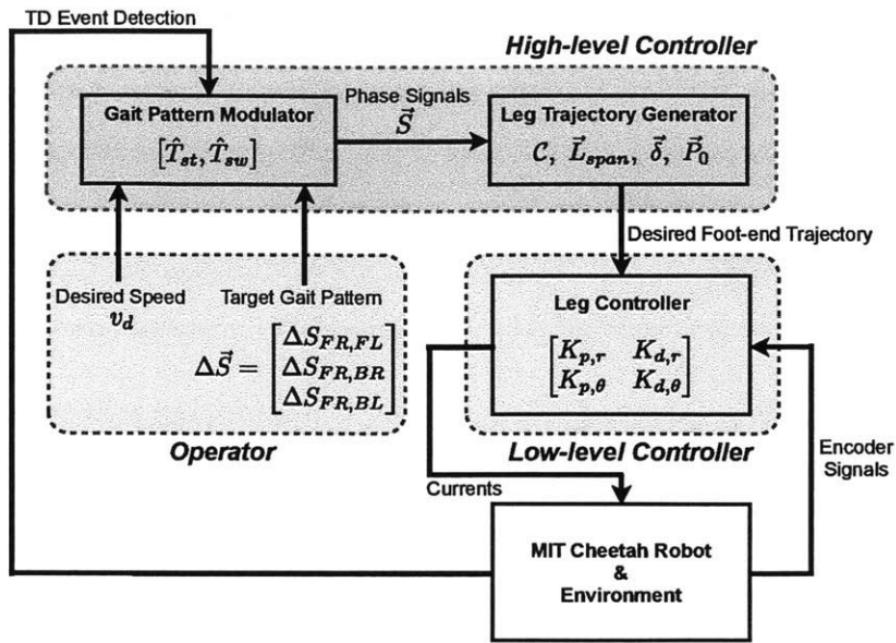


Obr. 1.11: Proces riadenia robota Solo [25].

a krútiacich momentov ovláda jednotlivé motory. Z hardvéru dostaváme spätnú odozvu o stave robota vo forme natočenia jednotlivých motorov a natočenia a pozície robota [25].

#### 1.1.4 CHAMP

CHAMP je vererejne dostupný framework pre riadenie štvornohých robotov<sup>5</sup>. Využíva ROS, je písaný v C++ a podporuje beh simulácie v Gazebe.



Obr. 1.12: Proces riadenia robota vo framework CHAMP[19].

Ako vidíme na obrázku 1.12, riadenie robota má dva hlavné komponenty a to High-level Controller a Leg Controller. High-level controller je zodpovedný za generovanie trajektórií pre jednotlivé nohy na základe požadovanej rýchlosťi robota od operátora a typu chôdzky. Low-level Controller posielá kontrolné signály motorom na základe trajektórií nôh. Z hardvéru sa ako odozva posielajú naspäť do High-level Controller-a

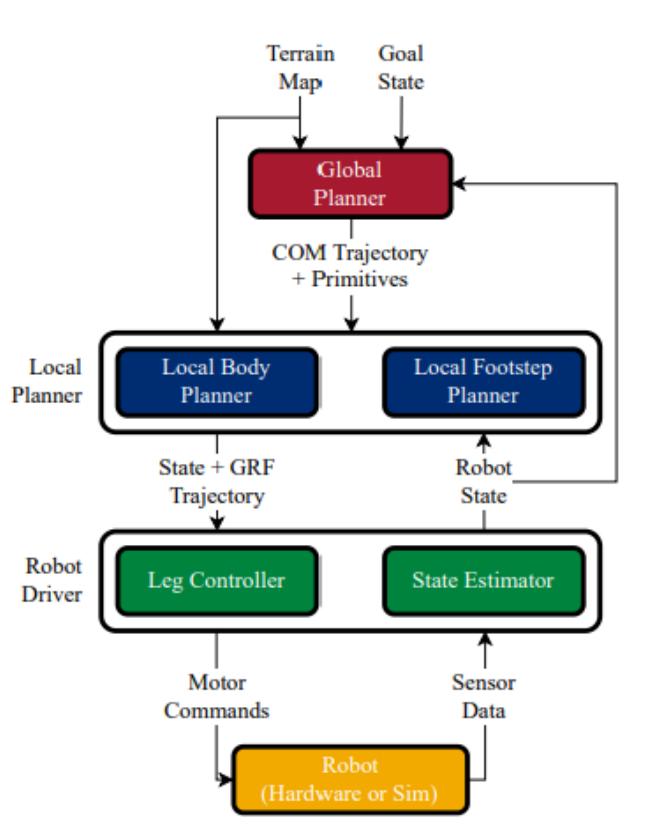
<sup>5</sup><https://github.com/chvmp/champ>

informácie o kontakte nohy s podložkou a do Leg-Controllera zas údaje o stave motorov robota. Chýba však spracovanie prostredia, v ktorom sa robot pohybuje.

### 1.1.5 Quad SDK

Je verejne dostupný framework<sup>6</sup>, vytvorený špeciálne pre štvornohé roboty. Využíva ROS a je písaný predovšetkým v C++ a Pythone. Nie je urobený na mieru pre špecifického robota, ale všeobecne pre štvornohé roboty. Podporuje ovládanie a beh na hardvéri ale aj simuláciu v softvéri Gazebo.

Na obr. 1.13 vidíme štruktúru Quad SDK. Časť Global Planner je zodpovedná za výpočet trajektórií tela robota bez kolízií tak, aby sa robot z aktuálneho stavu dostał do želaného. Vstupom je 2.5D reprezentácia okolitého terénu a cieľový stav robota. Cieľovým stavom môže byť aj príkaz od operátora z ovládača. Taktiež doň vstupuje aktuálny stav robota a parameter toho, aký je želaný krok robota [23].



Obr. 1.13: Štruktúra Quad SDK. Hierarchická štruktúra Quad SDK, ktorá má 3 hlavné časti: Global Planner, Local Planner a Robot Driver [23].

Local Planner na základe okolitého terénu, výstupu z Global Planner a stavu robota vypočíta kedy a kde majú byť nohy v kontakte s podložkou a akou silou majú pôsobiť nohy na podložku. Má dve časti a to Local Body Planner a Local Footstep Planner.

<sup>6</sup><https://github.com/robomechanics/quad-sdk>

Local Body Planner naplánuje najlepšie reakčné sily nôh od podložky, tak aby robot sledoval požadovanú trajektóriu tela. Local Footstep Planner plánuje trajektórie jednotlivých nôh.

Tretia časť - Robot Driver je rozhranie medzi plánom a harvérom, prípadne simuláciou. Leg Controller je zodpovedný za výpočet správnych krútiacich momentov na motoroch na základe reakčných síl a trajektórií nôh. State Estimator zbiera informácie o robotovi zo senzorov za účelom zstrojenia čo najlepšieho obrazu o stave robota. To zahŕňa pozíciu a natočenia tela robota, ale aj uhol natočenia jednotlivých motorov. V prípade, že máme spustenú iba simuláciu tieto informácie poskytuje Gazebo.

Quad SDK okrem častí opísaných vyššie obsahuje pomocné triedy, kde je napríklad kinematika a dynamika robota a jeho celkový model. Na to využíva knižnicu Rigid Body Dynamics Library (RBDL) [12]. Táto knižnica si načíta model robota zo súboru typu Unified Robot Description Format (URDF). Tento typ súboru je štandardným formátom pre fyzický opis robota.

Táto knižnica bola testovaná nielen na robotoch v simuliácii ale aj na hardvéri na rôznych robotoch ako napríklad Ghost Robotics Spirit alebo Unitree Robotics A1.

## 1.2 Diskusia

Riadenie štvornohých robotov je netriviálny proces, ktorý musí zohľadňovať veľa aspektov naraz. Pri adaptívnom riadení je dôležité mať spätnú väzbu z robota, ale ukazuje sa ako nevyhnutnosť poznať prostredie, v ktorom sa robot nachádza. To vyžaduje hardvér, ktorý s týmito potrebami ráta a má potrebné senzory. Od softvéru požadujeme aby vedel dátá zo senzorov spracovať v reálnom čase a taktiež v reálnom čase sa na ich základe rozhodnúť ako optimálne riadiť robota. Z vyššie popísaných možností riadenia najlepšie tieto potreby spĺňa Quad SDK. Tento framework ráta s dátami so senzorov, beží v reálnom čase a je možné ho adaptovať na rôznych robotov. Ráta však s tým, že väčšina robotov ma podobné nohy, ktoré sa skladajú z horného a spodného segmentu. Náš robot má o niečo komplikovanejšiu konštrukciu nôh, preto žiadne z vyššie spomínaných riešení nie je možné využiť bez väčších úprav. Quad SDK taktiež využíva podobné technológie aké sme pri vývoja robota už použili a všetok kód je verejne dostupný, čo napríklad pri ANYmalovi nie je. Framework CHAMP, tak ako aj pri robotovi Cheetah, neráta pri výpočte optimálneho riadenia robota s prostredím, v ktorom sa robot nachádza. Robot Solo má verejne dostupný kód a aj sníma svoje okolie, oproti Artabanovi je však veľmi ľahký a na každej nohe má o 1 motor menej. Na základe už vyššie spomenutého sme sa rozhodli pre knižnicu Quad SDK pre riadenie robota Artabana.

# Kapitola 2

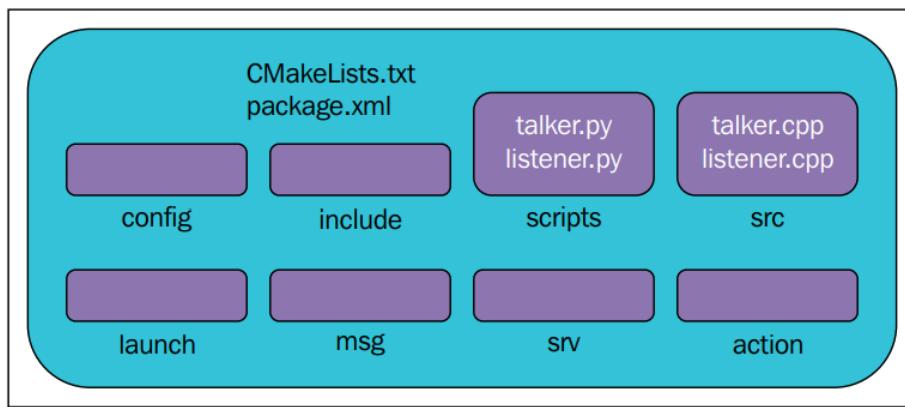
## Implementácia

### 2.1 Prehľad technológií

Pre riadenia robota sme sa rozhodli využiť framework Quad SDK. Tento framework je schopný adaptívneho riadenia štvornohých robotov, je verejne dostupný aj s dokumentáciou<sup>1</sup>. Využíva nasledovné technológie: ROS, URDF, SDF, C++, Python, Gazebo. Väčšinu z nich sme pri riadení robota používali aj doteraz, čo je veľká výhoda.

#### Robotic Operating System (ROS)

Robotic Operating System (ROS) je verejne dostupný systém. Pozostáva z knižníc a nástrojov, ktoré majú pomôcť vývojárom vyvíjať štruktúrovaný a udržateľný kód pre robotov [3].



Obr. 2.1: Štruktúra ROS balíčku [16].

Funguje ako množina procesov (nodes), ktoré je možné jednotlivo púštať, a komunikujú medzi sebou posielaním a prijímaním správ. Toto umožňuje zmeniť časť kódu bez zásahu alebo obmedzení na inej. Jednotlivé procesy je možné zoskupovať do balíčkov

<sup>1</sup><https://github.com/robomechanics/quad-sdk/wiki/1.-Getting-Started-with-Quad-SDK>

(packages) [24]. Typická štruktúra takého balíčku je na obr. 2.1. Tieto balíčky väčšinou potom predstavujú jednotlivé repositories na Githubu pri spolupráci viacerých vývojárov.

ROS taktiež obsahuje viacero bežne používaných knižníc a funkcionálít pri vývoji robotov, napríklad low-level control. Podporuje viacero programovacích jazykov, okrem iných aj C++ a Python.

### Universal Robot Description Format (URDF)

Universal Robot Description Format (URDF) je XML (eXtensible Markup Language) formát súboru, slúžiaci na opis robota v ROS-e. Základnými kameňmi tohto formátu sú segment (link) a kĺb (joint) [18].

Kĺb spája vždy dva segmenty. Poznáme viacero typov kĺbov a to napríklad otáčavý, s limitmi alebo bez limitov, posuvný a pevný - pevne spája dva segmenty. Dôležitým parametrom každého kĺbu je, kde sa vrámcí segmentov, ktoré spája, nachádza. Taktiež každý kĺb má popísanú svoju os otáčania ako 3d vektor. Nižšie vidíme príklad kĺbu v URDF:

```
<joint name="joint1" type="continuous">
  <parent link="link1"/>
  <child link="link2"/>
  <axis xyz="1 0 0"/>
  <origin xyz="0 0 0.5"/>
</joint>
```

Kĺb joint1 je otáčavý kĺb bez limitov a spája segment link1 a link2. Otáča sa okolo osi x.

Segment popisuje hmotnosť jednotlivých častí robota a ich inerčnú maticu. Nižšie vidíme príklad definície segmentu:

```
<link name="link1">
  <inertial>
    <mass value="5"/>
    <inertia
      ixx="0.3"
      ixy="0.0"
      ixz="0.0"
      iyy="0.4"
      iyz="0.0"
      izz="0.2"/>
  </inertial>
```

</link>

URDF je schopné popísť kinematický strom, avšak nie je schopné popísť kinematickú slučku. Tá sa dá uzatvoriť dodatočne v niektorých softvéroch.

Vo formáte URDF máme popis celého Artaban so všetkými jeho časťami - segmentmi a kĺbmi.

### Simulation Description Format (SDF)

SDF je ďalší je XML formát súboru pre popis robota. Využíva ho najmä Gazebo. Je podobný URDF, tiež jeho základnom je segment (link) a kĺb (joint), ale umožňuje vytvárať aj kinematické slučky v systéme [18]. URDF je do SDF možné konvertovať automaticky.

### Gazebo

Gazebo je fyzikálny 3D simulátor, určený primárne pre vývoj robotov. Umožňuje simuláciu robota aj s prostredím, v ktorom sa pohybuje. Vďaka tomu je možné si overiť funkčnosť kódu a vyladiť niektoré chyby ešte predtým ako sa otestuje na robotovi.

## 2.2 Popis Artabana

V tejto časti popíšeme nášho robota Artabana. Dôležité je pre nás ako má skonštruktované nohy, ale aj jeho kinematika. V procese jeho riadenia taktiež potrebujeme poznať Jakobiány jednotlivých nôh a inverznú dynamiku.

## 2.3 Inverzná dynamika robota

Poznáme priamu (FD) a inverznú dynamiku (ID) tuhých telies. Priama dynamika predstavuje výpočet požadovaných rýchlosťí daného systému na základe modelu daného systému vid. 2.1. Inverzná dynamika je výpočet síl alebo krútiacich momentov, ktoré musia vzniknúť v systéme aby sa vytvorilo požadované zrýchlenie [11], vid. 2.2.

$$\ddot{q} = FD(model, q, \dot{q}, \tau) \quad (2.1)$$

$$\tau = ID(model, q, \dot{q}, \ddot{q}) \quad (2.2)$$

$q$  predstavuje pozíciu robota (pozícia ťažiska robota vo svete, natočenie jednotlivých motorov),  $\dot{q}$  predstavuje rýchlosťi,  $\ddot{q}$  zrýchlenia a  $\tau$  krútiace momenty.  $model$  je model robota ktorý zahŕňa nielen rozmery jednotlivých častí robota a popis jednotlivých kĺbov ale aj hmotnosti všetkých častí.

Pri výpočte inverznej dynamiky štvornohého robota nás zaujímajú krútiace momenty motorov  $\tau$ , za podmienky, že poznáme ich natočenie, rýchlosť, polohu a orientáciu tela a model robota.

Vo všeobecnosti vychádza výpočet inverznej dynamiky pre kinematický strom z pohybovej rovnice [11]:

$$\tau = H(q)\ddot{q} + c(q, \dot{q}, f_{ext}) \quad (2.3)$$

Kde  $H$  je matica zotrvačnosti,  $C$  je vektor sôl odpovedajúci všetkým silám pôsobiačim v systéme (napr. gravitačná sila, odstredivé sily,...) okrem tých, ktoré sú spôsobené  $\tau$ .

Na výpočet  $\tau$  sa využíva Newton-Eulerov algoritmus.

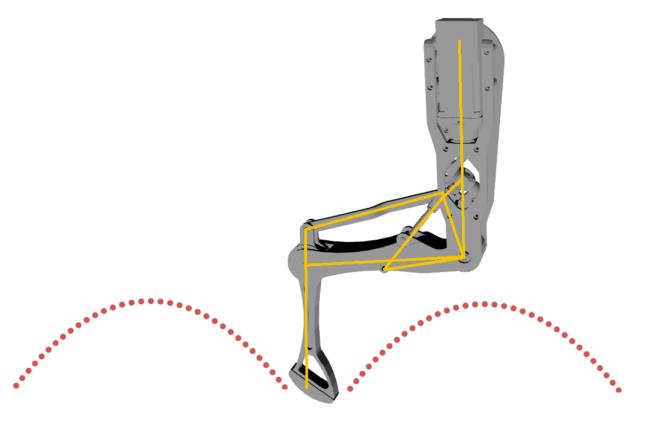
# Kapitola 3

## Výsledky

Testovali sme chôdzu robota s viacerými spôsobmi riadenia. Konkrétnie priamym pozičným ovládaním. Pomocou knižnice Quad SDK bez iverznej dynamiky a s pomocou Quad SDK a s čiastočnou inveznou dynamikou robota a Quad SDK.

### 3.1 Priame pozičné ovládanie

Motory robota vieme ovládať priamo, kedy im nastavíme pozíciu natočenia. Máme funkciu, ktorá nám generuje súradnice bodu pre každú nohu v časovom bode, do ktorého sa má noha dostať. Tieto body sa nachádzajú na krivke po ktorej sa má noha hýbať pri chôdzi, vid. obr. 3.1. Pomocou inverznej kinematiky vypočítame uhly natočenia motorov, ktoré potom pošleme ovládaču motora. Na takéto ovládanie potrebujeme poznať inverznú kinematiku, tú však už pre Artabana poznáme. Túto metódu sme otestovali v simulácii a aj na hardvéri.



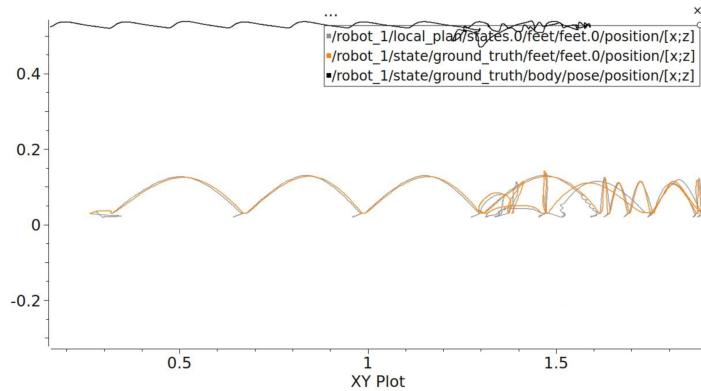
Obr. 3.1: Zobrazanie zadnej nohy a jej trajektórie.

Na hardvéri táto metóda nefungovala ideálne, keďže jej chýba akákoľvek spätná väzba a keď sa robot z nejakého dôvodu naklonil alebo inak vychýlil z plánu, čo sa vždy stalo, tak spadol. Taktiež náš model robota nie je dokonalý a aj to malo vplyv

na jeho chôdzu. Keď bol robot zavesený na lanách tak sme týmto spôsobom validovali aspoň kinematiku, rozsah nôh a funkčnosť a schopnosť hardvéru sa hýbať po želaných trajektóriách. Taktiež sme videli potrebu riadenia robota so spätnou väzbou o stave robota a následnou úpravou plánu pohybu robota na základe meraní zo senzorov o stave robota.

## 3.2 Quad SDK bez inverznej dynamiky

Toto riešenie sme validovali aj v simulácii aj na hardvéri. V simulácii sme najprv vyladili chôdzu a potom sme ju skúšali na hardvéri. Môžeme vidieť výrazné rozdiely. Tie sú spôsobené rozdielmi v hmotnosti jednolivých častí robota medzi modelom a realitou. Ďalej rozdielnymi rozmermi jednolivých častí robota, ktoré vznikli pri výrobe alebo opotrebovaním. Najväčším faktorom sú však pravdepodobne trenia vznikajúce v nohách robota, s ktorými v simulácii nepočítala.

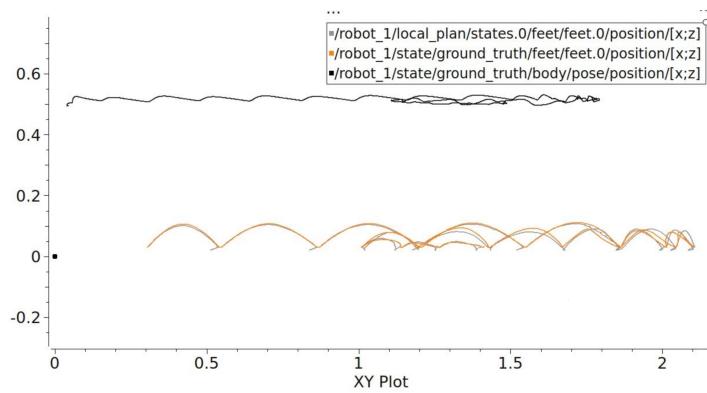


Obr. 3.2: Porovnanie plánovanej trajektórie nohy so skutočnou. Oranžovou a sivou vidíme skutočnú a plánovanú trajektóriu ľavej prednej nohy v simulácii. Čierrou je znázornená trajektória tela.

Na grafe 3.2 vidíme porovnanie plánovanej a skutočnej trajektórie ľavej prednej nohy. Najprv sme šli robotom iba krokom dopredu a vtedy robot šiel veľmi dobre, avšak keď sme ním začali otáčať už začal mať problémy a nevedel udržať v požadovanej výške telo ani sledovať naplánovanú trajektóriu nohy.

## 3.3 Quad SDK s inverznou dynamikou predných nôh

Tento spôsob riadenia sme validovali v simulácii aj na hardvéri. Zadné nohy sme riadili ako doteraz, iba pri predných sme využili aj inverznú dynamiku. Znova sme zaznamenali odchýlky medzi simuláciou a hardvérom. Na grafe 3.3 vidíme, že nohy sledujú svoju plánovanú trajektóriu veľmi dobre. Taktiež telo sa drží v požadovanej výške oveľa stabilnejšie aj pri otáčaní.



Obr. 3.3: Porovnanie plánovanej trajektórie nohy so skutočnou. Oranžovou a sivou vidíme skutočnú a plánovanú trajektóriu ľavej prednej nohy v simulácii. Čierne je znázornená trajektória tela.

# Literatúra

- [1] Gazebo. [Citované 2023-11-23] Dostupné z <https://gazebosim.org/doc>.
- [2] MPC scheme basic. [Citované 2023-11-20] Dostupné z [https://en.wikipedia.org/wiki/Model\\_predictive\\_control#/media/File:MPC\\_scheme\\_basic.svg](https://en.wikipedia.org/wiki/Model_predictive_control#/media/File:MPC_scheme_basic.svg).
- [3] ROS - ROS Wiki. [Citované 2023-11-22] Dostupné z <https://wiki.ros.org/ROS>.
- [4] What is Model Predictive Control? - MATLAB Simulink. [Citované 2023-11-20] Dostupné z <https://www.mathworks.com/help/mpc/gs/what-is-mpc.html>.
- [5] Mini cheetah is the first four-legged robot to do a backflip , 2019. [Citované 2023-11-22] Dostupné z <https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304>.
- [6] Spot, 2023. [Citované 2023-11-20] Dostupné z <https://bostondynamics.com/products/spot/>.
- [7] Jared Carlo, Patrick Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. pages 1–9, 10 2018.
- [8] Thomas Corbères, Carlos Mastalli, Wolfgang Merkt, Ioannis Havoutis, Maurice Fallon, Nicolas Mansard, Thomas Flayols, Sethu Vijayakumar, and Steve Tonneau. Perceptive Locomotion through Whole-Body MPC and Optimal Region Selection. *arXiv (Cornell University)*, 5 2023.
- [9] Jared Di Carlo. Software and control design for the mit cheetah quadruped robots, 2020.
- [10] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8484–8490, 2019.
- [11] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2007.

- [12] Martin L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, pages 1–17, 2016.
- [13] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model predictive control. 08 2022.
- [14] Ruben Grandia, Andrew J. Taylor, Aaron D. Ames, and Marco Hutter. Multi-layered safety for legged robots via control barrier functions and model predictive control. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [15] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, Jonathan Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An Open Torque-Controlled Modular robot architecture for legged locomotion research. *IEEE robotics and automation letters*, 5(2):3650–3657, 4 2020.
- [16] Lentin Joseph. *Mastering ROS for robotics programming : design, build, and simulate complex robots using robot operating system and master its out-of-the-box functionalities*. 1 2015.
- [17] D. Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via Whole-Body impulse control and model predictive control. *arXiv (Cornell University)*, 2019.
- [18] Anis Koubâa. *Robot Operating System (ROS)*. 2016.
- [19] Jongwoo Lee. Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the mit cheetah robot, 2013.
- [20] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [21] Carlos Mastalli, Saroj Chhatoi, Thomas Corberes, Steve Tonneau, and Sethu Vijayakumar. Inverse-dynamics mpc via nullspace resolution. 2022.
- [22] Zuzana Mačicová. Implementácia aproximácie priamej a inverznej kinematiky paralelného mechanizmu nôh robota artaban. Master’s thesis, Comenius University, FMFI, 2022.

- [23] Joseph Norby, Yanhao Yang, Ardalan Tajbakhsh, Jiming Ren, Justin K. Yim, Alexandra Stutt, Qishun Yu, Nikolai Flowers, and Aaron M. Johnson. Quad-SDK: Full stack software framework for agile quadrupedal locomotion. In *ICRA Workshop on Legged Robots*, May 2022.
- [24] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. 11 2015.
- [25] Fanny Risbourg, Thomas Corbères, Pierre-Alexandre Léziart, Thomas Flayols, Nicolas Mansard, and Steve Tonneau. Real-time footstep planning and control of the solo quadruped robot in 3D environments. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.