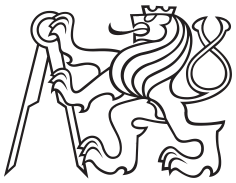
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Vývoj API pro robota Spot v simulátoru

Tomáš Oborný

Školitel: Mgr. Martin Pecka, Ph.D.
Obor: Kybernetika a robotika
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Oborný** Jméno: **Tomáš** Osobní číslo: **507384**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vývoj API pro robota Spot v simulátoru

Název bakalářské práce anglicky:

Design of an API for Simulated Spot Robot

Pokyny pro vypracování:

Cílem práce je navrhnout a implementovat API pro ovládání robota Spot v simulátoru Gazebo, včetně podpory pro ovládání integrovaného manipulátoru. Cílem je poskytnout v simulátoru velkou část API dostupného pro reálného robota Spot ([1], [3]). Důraz bude kladen zejména na věrohodnost simulace, funkce zpracování senzorických vstupů, jejich analýzu, a na ovládání manipulátoru.
Student nejdříve provede analýzu existujících API pro ovládání robota (Python API od výrobce [3], ROS API [1]). Z této analýzy sestaví seznam API užitečných v simulaci. Tato API student následně implementuje tak, aby simulované chování robota co nejvíce odpovídalo chování reálného robota. Pro ovládání manipulátoru student využije standardní rozhraní pro ovládání manipulátorů Moveit [2,4]. Následně student provede experimenty ověřující, že simulovaný i reálný robot se pro podobné vstupy chovají podobně.
Cílem práce není vyvíjet samotné funkce pro nízkoúrovňové ovládání nohou a těla robota - to je tématem jiné práce [5]. Nicméně v rámci tohoto projektu je úkolem identifikovat a zpřesnit parametry tohoto nízkoúrovňového modelu tak, aby co nejvíce odpovídal chování reálného robota Spot. Bakalářská práce zároveň bude vycházet z již hotových částí dokončených v rámci semestrálního projektu – identifikace inerciálních parametrů robota, základní naladění nízkoúrovňového ovládání, simulace API pro pohyb těla robota a vytváření výškové mapy z hloubkových kamer.

Seznam doporučené literatury:

[1] https://github.com/h euristicus/spot_ros
[2] https://github.com/estherRay/Spot-Arm
[3] https://dev.bostondynamics.com/
[4] https://ros-planning.github.io/moveit_tutorials/
[5] Jakub Jon, Analysis of State-of-the-Art Quadruped Locomotion Controllers for Use in Simulation. Bakalářská práce, FEL ČVUT, 2024.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. Martin Pecka, Ph.D. vidění pro roboty a autonomní systémy FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Mgr. Martin Pecka, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Poděkování

Rád bych poděkoval svému vedoucímu práce Mgr. Martinu Peckovi, Ph.D. za výbornou a včasnou komunikaci, diskutování problémů a celkovou pomoc s prací. Také bych rád poděkoval své rodině a přátelům za veškerou podporu.

Prohlášení

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

V Praze,
24. May 2024

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci API pro ovládání robota Spot v simulátoru Gazebo, včetně podpory pro ovládání integrovaného manipulátoru. V současné době již existuje API pro reálného robota Spot, API umožňuje ovládat robota pomocí joysticku, tabletu a API rozhraní propojené s ROSem *spot_ros*, avšak pro simulovaného robota není implementováno. Cílem této práce je vytvořit API pro simulovaného robota Spot tak, aby se tyto programátorská rozhraní co nejméně lišila.

V práci se postupně zaměříme na stručný teoretický úvod využitých nástrojů s kterými při tvorbě API a ovládání manipulátorů pracujeme, poté zmíníme využité zdroje, na které práce navazuje a vychází z nich. Dále si ukážeme, jak jsme propojili manipulátor s MoveItem, následně jsou popsány všechny možnosti ovládání jak manipulátoru, tak robota Spot. Srovnáme API reálného a simulovaného robota a nakonec porovnáme i simiuvaného a reálného Spota.

Klíčová slova: Spot, ROS, MoveIt, API

Školitel: Mgr. Martin Pecka, Ph.D.

Abstract

This bachelor thesis focuses on the design and implementation Application Programming Interface (API) for controlling the robot Spot in Gazebo simulator, including support for controlling the integrated manipulator. The API for real robot Spot, developed by Boston Dynamics, allows controlling Spot with a joystick, tablet or via an API connected with ROS from *spot_ros*. However, an API for the simulated Spot is not yet implemented. The main goal of this bachelor thesis is to create an API for the simulated robot that is as similar as possible to the API for the real robot.

This thesis begins with a brief theoretical introduction to the tools and resources used for creating API and controlling Spot's manipulator. It will then discuss existing resources that served as a foundation for this work. Following this, the thesis describes how the manipulator is integrated with MoveIt and details the methods for controlling manipulator and Spot. Next, it compares the APIs of the simulated and real robots. Finally, the thesis concludes with a comparison between the real and the simulated Spot.

Keywords: Spot, ROS, MoveIt, API

Title translation: Design of an API for Simulated Spot Robot

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Cílem této bakalářské práce je navrhnout a implementovat aplikační programové rozhraní (API) pro ovládání robota Spot v simulaci Gazebo, včetně podpory pro ovládání manipulátoru. API reálného Spota [1] již existuje a uživatel má možnost ovládat Spota pomocí tabletu, joysticku a rozhraním ROS API [6]. Cílem je vytvořit a přizpůsobit API simulovaného robota tak, aby rozdíly mezi API skuteřného a simulovaného Spota byly minimální. Práce navazuje na projekt [3], který se zabývá nízkoúrovňovým řízením robota Spot v simulaci Gazebo a na projekt *Spot-Arm* [2], který propojuje manipulátor Spota s MoveItem.

V práci se postupně zaměříme na stručný teoretický úvod využitých nástrojů, s kterými při tvorbě API a ovládání manipulátoru pracujeme, rozebereme všechny zdroje, na které jsme navazovali, nebo z nich čerpali. Ukážeme si, jak jsme postupovali při propojování modelu Spota s MoveItem. Dále popíšeme veškeré možnosti ovládání a jeho vytváření, které má možnost uživatel využít v simulaci. Rozebereme, jak Spot zvedá objekty v simulaci, srovnáme API reálného a simulovaného Spota a nakonec porovnáme jejich vlastnosti.

Kapitola 2
Teoretický úvod k použitým nástrojům

V této kapitole si představíme hlavní nástroje, které jsme při implementaci používali. Těmito nástroji jsou robotický operační systém a MoveIt, který poskytuje řízení manipulátoru.

2.1 ROS

Robotický operační systém (ROS) je volně dostupná sada softwarových knihoven a nástrojů, které usnadňují vytváření robotických aplikací. Díky této sadě nástrojů pro správu procesů, komunikace mezi různými částmi systému a integrovanou podporu pro senzory, včetně aktuátorů a dalšího hardwaru, poskytuje ROS flexibilní a distribuovanou výpočetní architekturu.

ROS můžeme rozdělit do několika hlavních komponent, kterými jsou uzly, topicy, služby a akce. Uzel je samostatný proces, který vykonává specifické úkoly. Komunikaci mezi uzly zajišťují topicy, což jsou pojmenované kanály, ze kterých uzly přijímají nebo na ně publikují zprávy. Touto architekturou ROS podporuje asynchronní komunikaci mezi různými částmi systému. Pro synchronní komunikaci ROS využívá služby, které umožňují uzlům publikovat zprávy a čekat na jejich odpověď, kdy se vyžaduje okamžitá zpětná vazba. Akce se využívají při dlouhotrvajících úkolech, umožňují sledování, řízení a případné předčasné ukončení.

ROS tímto umožňuje rozdělit jinak komplexní systémy do menších částí, které mohou být vyvíjeny, testovány a laděny separátně.

2.2 MoveIt

MoveIt je na ROSu založený nástroj pro robotické pohybové plánování a řízení. Poskytuje nástroje pro manipulaci s robotickými rameny, kinematiku, dynamiku, kolizní detekci, vizualizaci a integraci se senzory. Usnadňuje vývoj a nasazení robotických aplikací, které vyžadují precizní a efektivní řízení pohybu.

MoveIt zajišťuje plánování pohybu, pomocí prohledávájících algoritmů z knihovny OMPL (The Open Motion Planning Library) zajišťuje efektivní

plánování trajektorií v prostředí s překážkami. Podporuje dopřednou i inverzní kinematiku a integrací dynamických modelů robotů umožňuje simulaci realistických pohybů a kontrolu dynamických efektů, jakými jsou například setrvačnost a tření. MoveIt můžeme propojit s RViz, který umožňuje interaktivní vizualizaci robotických pohybů, prostředí a kolizních modelů. Uživatel má možnost interaktivně nastavovat cíle a sledovat plánované trajektorie za pomocí interaktivního kurzoru.

2.2.1 MoveIt Setup Assistant

MoveIt Setup Assistant je grafické uživatelské rozhraní, které po nahrání URDF robota umožňuje intuitivně a efektivně konfigurovat kinematický model robota, kolizní objekty, plánovací skupiny a jejich předem nadefinované pozice. MoveIt Setup Assistant dále umožňuje konfiguraci koncového efektoru, pasivních kloubů a vytvoření prvotních kontrolérů pro plánovací skupiny. Nakonec vygeneruje všechny potřebné soubory pro propojení robota s MoveItem.

2.2.2 MoveIt Servo

MoveIt Servo (dále jen Servo) umožňuje přesné řízení manipulátoru v reálném čase. Servo je navrženo pro aplikace, kde je vyžadována vysoká rychlost a nízká latence při řízení manipulátoiru, takovými situacemi jsou například ruční ovládání robota nebo sledování objektů. Servo umožňuje ovládání kloubů pozičně, rychlostně i pomocí zrychlení.

Mezi klíčové vlastnosti Serva patří řízení v reálném čase, které dosahuje frekvence odezvy až 1 kHz. Umožňuje přesné sledování trajektorií na základě vstupních příkazů. Přijímá různé typy vstupů, včetně pozice a orientace koncového efektoru a kloubových příkazů. Detekuje a předchází kolizím během pohybu a bezpečnostní funkce zahrnují limity rychlosti a zrychlení jednotlivých kloubů.

Kapitola 3

Využité zdroje na které práce navazuje

Vývoj API pro robota Spot v simulátoru navazuje na několik již existujících prací. Pro přehlednost a odkazy v textu bychom rádi všechny zmínili a udělali k nim rychlý úvod.

3.1 Nízkoúrovňové řízení Spota v simulaci

Projekt [3] poskytuje nízkoúrovňové řízení pro Spota v simulaci. Pro řízení celého těla (bez manipulátoru) se používá tzv. Model predictive control (MPC). Změnili jsme názvy kloubů a ramen robota tak, aby odpovídaly názvům skutečného Spota a upravili hmotnosti jednotlivých ramen a jejich momenty setrvačnosti. Dále jsme model doplnili o níže zmíněný manipulátor a propojili ho s MPC tak, že MPC předpokládá, že manipulátor je v neměnící se poloze, konkrétně ve *stow* pozici. Toto umožňuje jednoduché provázání těla s manipulátorem. Nevýhodou však je, že při pohybu Spota musí být manipulátor ve *stow* pozici, jinak je pohyb ovlivněn.

3.2 Model manipulátoru a jeho propojení s MoveItem

Model manipulátoru byl již vytvořen projektem estherRay [2] ze kterého jsme čerpali především model z URDF a kontrolu správného propojení manipulátoru s MoveItem. Manipulátor není propojený s tělem Spota a bylo zapotřebí provést několik úprav, jako je například propojení manipulátoru s MoveItem včetně těla Spota nebo změna parametrů kontrolérů.

3.3 Propojení MoveIt Serva a robota UR5

Při propojování manipulátoru Spota k Servu jsme postupovali podle propojení Serva [4] s robotem UR5. Abychom zajistili správnou funkčnost převzali jsme několik souborů, které jsme upravili pro naše potřeby. Z UR5 se jednalo o použití balíku *controller_manager* a správný typ kontroléru a z MoveIt Serva především o správnou konfiguraci pro Servo.

3.4 API pro reálného Spota

Využijeme oficiální API od Boston Dynamics pro reálného Spota [1] a API pro komunikaci se Spotem přes ROS [6] Vycházeli jsme především z API *spot_ros* [6]. Abychom zachovali všechny funkcionality, použili jsme celý repozitář, který jsme následně upravili pro naše potřeby. Tato úprava se týkala hlavně RViz nastavení a pro propojení simulovaného Spota bylo zapotřebí odebrat odkazy na reálného Spota. Zachovali jsme názvy komunikačních kanálů a služeb, dále jsme upravili funkce tak, aby komunikovaly se simulovaným Spotem.

Kapitola 4

Postup propojení manipulátoru s MoveItem

4.1 URDF

Jak již bylo řečeno, model manipulátoru již existuje, avšak bylo třeba ho upravit tak, aby co nejvíce odpovídal skutečnému manipulátoru. Boston Dynamics uvadí, že manipulátor má celkovou hmotnost 8 kg, která se zásadně liší od použitých 54,6 kg [1]. Podařilo se nám spojit s Boston Dynamics, kteří uvedli, že každý Spot může mít jiné parametry, protože součástky mění. Sdíleli s námi alespoň hmotnosti jednotlivých rozebraných částí Spota. V URDF jsme tedy upravili jednotlivé hmotnosti tak, aby se zásadně nelišily od skutečných hodnot a přepočítali jsme momenty setrvačnosti. Také jsme přejmenovali jednotlivé klouby a ramena u manipulátoru tak, jak je tomu u reálného Spota. U těla Spota jsme byli schopni přejmenovat pouze ramena, protože při změně názvů jsme ovlivnili nízkoúrovňové řízení [3] a nebyli jsme schopni dohledat, kde řízení předpokládá původní názvy kloubů.

4.2 Moveit Setup

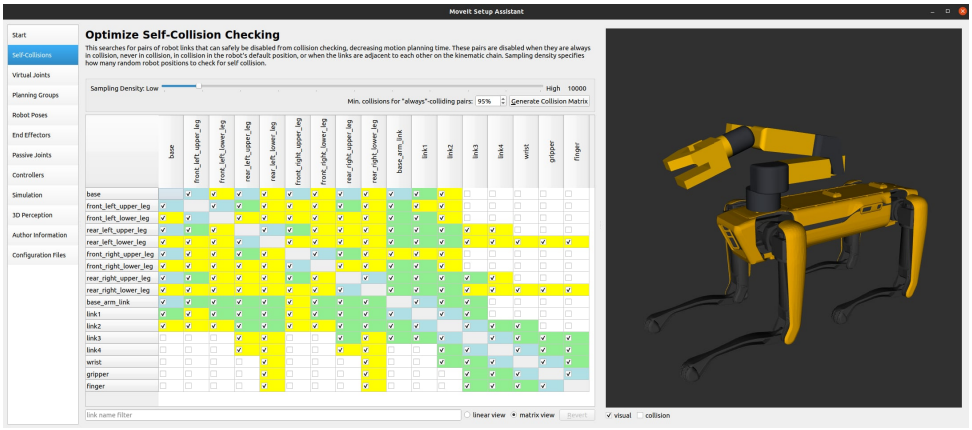
Pro prvotní propojení Spota s MoveItem jsme využili MoveIt Setup Assistant, který vygeneroval potřebné soubory. Konfigurace kterou jsme použili vypadá následovně.

4.2.1 Konfigurace

Po vložení URDF do MoveIt Setup Assistantu jsme jako první zadali, pro které části Spota MoveIt nemusí řešit kolize. Řešení kolizí je výpočetně náročné, a proto bylo důležité, abychom co nejpodrobněji vypnuli řešení kolizí mezi částmi, které do kontaktu nemohou přijít nebo naopak jsou v kolizi pořád, protože na sebe navazují. O kolize mezi tělem a nohami se MoveIt starat vůbec nemusí, také některé části manipulátoru nemají možnost zasáhnout tělo nebo nohy Spota.

Dále jsme pomocí virtuálního kloubu *arm_base_virtual_joint* připojili manipulátor k tělu, vytvořili plánovací skupiny pro manipulátor. V našem

[1] Tato hodnota je naprosto šílená a nemáme tušení, kde na ní přišli.



Obrázek 4.1: Vyobrazení zrušení kolizí mezi jednotlivými rameny.

případě jsme jej rozdělili na kinematický řetězec *arm* od kořene až po chapadlo manipulátoru a kloub pro svírání chapadla *gripper*. Důvodem, proč jsme pro chapadlo nezahrnuli i kloub, který umožňuje natočení chapadla, byl správce kontroléru, který nám poté neumožnil použít kontrolér pro ovládání manipulátoru v reálném čase. Také jsme zde definovali pózy manipulátoru pomocí hodnot úhlů kloubů, na které se při použití v simulaci můžeme odkázat. Vytvořili jsme koncový efektor, který slouží jako referenční rámec pro inverzní kinematiku. Zadefinovali jsme zde pasivní klouby, o které se MoveIt nestará a tím byly klouby na nohou Spota. A nakonec jsme vygenerovali Assistantem *SRDF*, kde jsou detaily konfigurace a prvotní kontroléry, které MoveIt používá pro ovládání fyzického hardware robota, včetně propojení robota s MoveiTem.

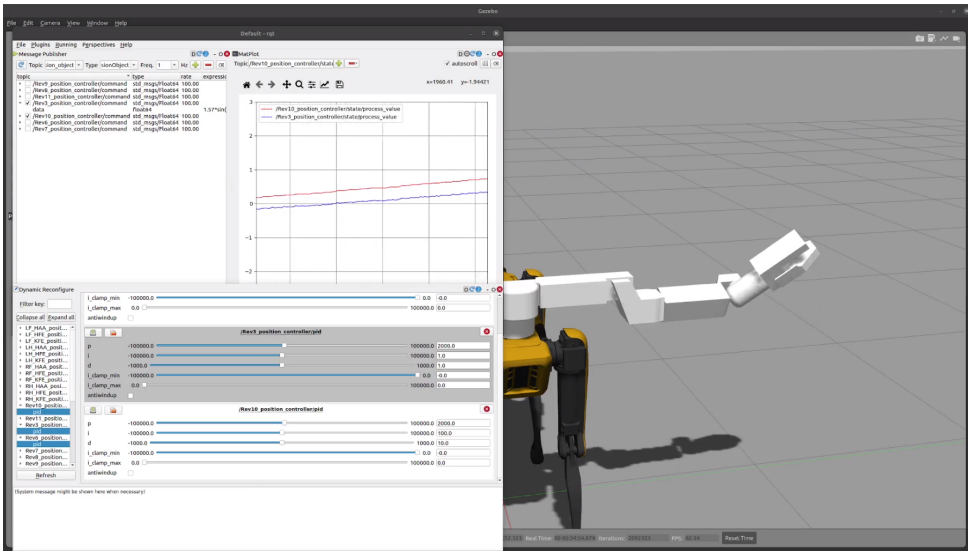
4.3 Kontroléry

Jak už bylo řečeno, prvotní kontroléry nám vygeneroval MoveIt Setup Assistent. Tyto kontroléry však byly nedostatečné a bylo třeba je upravit. Hodnoty PID regulátorů kloubů jsme upravovali pomocí *rqt_gui*, kde jsme sledovali hodnoty úhlů natočení kloubů oproti referenční hodnotě.

4.4 MoveIt Servo

Servo jsme použili pro ovládání manipulátoru v reálném čase pomocí joysticku. Bylo třeba nakonfigurovat potřebné propojení se Servem. Pro správné fungování jsme využili již vytvořené propojení robota UR5 se Servem. Využili jsme struktury, propojení a nahradili jsme poziční kontroléry za silové. Pro využití Serva bylo potřeba přidat skupinový kontrolér *joint_group_position_controller* typu *effort_controllers/JointTrajectoryController*.

Abychom umožnili přepínání mezi jednotlivými kontroléry, využili jsme balíček *controller_manager*, který však způsoboval problémy v již zmíněném chapadle, proto jsme pro chapadlo ponechali pouze kloub, který chapadlo otvírá



Obrázek 4.2: Postupné lazení hodnot PID regulátorů pro každý kloub

a zavírá. Nakonec jsme v konfigurační složce Serva *ur_simulated_config.yaml* změnili názvy ramen.

Kapitola 5

Ovládání Spota a jeho manipulátoru

V této kapitole si ukážeme všechny možnosti, kterými uživatel může ovládat Spota a jeho manipulátor v simulaci.

5.1 Ovládání Spota

Ovládání Spota v simulaci bylo tématem bakalářského projektu, aby však ovládání bylo kompletní, rozhodli jsme se v této práci uvést a vysvětlit i tuto část. U skutečného Spota jsou tyto dvě části přeci jen také provázané.

5.1.1 Komunikace mezi Spotem a ovládacími programy

Komunikace mezi všemi následujícími aplikacemi pro ovládání a ovládací jednotkou Spota probíhá pomocí zprávy *Twist*.*Twist* zpráva dodává ovládací jednotce informaci o požadované rychlosti Spota, tím je rychlost v osách x,y a z v souřadnicovém systému Spota a jeho úhlové rychlosti podle těchto os. Tato zpráva je publikována na topic */twist_cmd*. Používá se však pouze rychlost v osách x a y a úhlové rychlosti okolo osy z.

5.1.2 Ovládání Spota pomocí klávesnice

Ovládání Spota pomocí klávesnice nebylo primárním cílem, avšak uživatel může tuto funkci využít v případě, že nemá k dispozici joystick. Pomocí knihovny v pythonu *keyboard* se detekují klíčové klávesy pro pohyb. Zvolili jsme ovládání pomocí šipek (nahoru ↑ , dolů ↓, doprava →, doleva ←), které určují směr pohybu Spota v jeho souřadnicovém systému. Spot se tak pohybuje rovně dopředu nebo dozadu a do stran doleva nebo doprava. Dále jsou detekovány klávesy *a* a *d*, které slouží pro otáčení Spota proti směru *a* a ve směru *d* hodinových ručiček.

5.1.3 Ovládání pomocí ukazatele

Prostředí RViz má v horním panelu tlačítko *Publish Point*, pomocí kterého má uživatel možnost zadat cílovou polohu do mapy. Zpráva se souřadnicemi

zvoleného místa se publikuje na topic */clicked_point*. Poloha tohoto bodu je v souřadnicovém systému simulovaného prostředí.

5.1.4 Ovládání pomocí vektorového ukazatele

RViz umožňuje uživateli zadat také cílovou polohu s požadovaným natočením pomocí volby *2D Nav Goal*. Po zvolení *2D Nav Goal* uživatel zadá požadovanou polohu pomocí stisknutí levého tlačítka myši a následným potažením kurzorem do daného směru zadá požadovanou orientaci. Zpráva se souřadnicemi a orientací v quaternionech se publikuje na topic */move_base_simple/goal*.

5.1.5 Ovládání Spota joystickem

Uživatel má možnost ovládat Spota i joystickem, což bylo již vytvořeno projektem na nízkoúrovňové řízení [3].

5.2 Ovládání manipulátoru

Jak jsme již zmínili, pro ovládání manipulátoru jsme využili nástroje MoveIt, který nám umožnil kompletní ovládání manipulátoru. Simulaci spouštíme pomocí *spot_with_arm.launch*.

5.2.1 Ovládání manipulátoru v RVizu

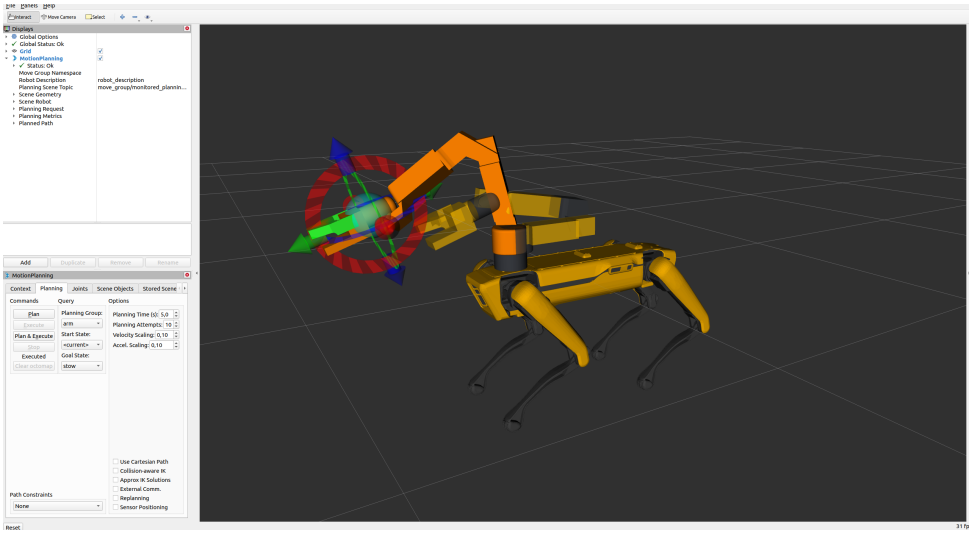
MoveIt plugin je zabudovaný do RViz a umožňuje ovládábá robotických manipulátorů. V našem případě se po spuštění otevřou dvě okna RVizu, jedno slouží pouze pro manipulaci s manipulátorem a v druhém najdeme kompletního Spota včetně manipulátoru, zde však manipulátor neovládáme.

Ovládací okno nabízí uživateli manipulovat s manipulátorem několika možnými způsoby. Uživatel pomocí interaktivního ukazatele může libovolně manipulovat plánovací scénou manipulátoru, což v reálném čase pohybuje s vizualizací chapadla. V okně v levém spodním rohu nalezneme *MotionPlanning* panel, pomocí kterého naplánujeme a vykonáme cestu do nastavené cílové polohy koncového efektoru pomocí tlačítka *Plan & Execute*.

Dále zde můžeme volit, zda chceme manipulovat s manipulátorem, nebo jeho chapadlem a v sekci *Goal State* můžeme volit předem definované pózy, které jsme vytvořili v MoveIt Setup Assistantu a náhodné pózy. *MotionPlanning* panel umožňuje manuálně nastavovat hodnoty úhlů jednotlivých kloubů plánovací scény.

5.2.2 Ovládání manipulátoru joystickem

Pro ovládání manipulátoru joystickem v reálném čase jsme využili Servo. Servo zpracovává data ze dvou topiců. Prvním topicem je */servo_server/delta_twist_cmds*, který přijímá zprávy typu *TwistStamped*, což jsou požadované rychlostní data koncového efektoru v kartézských souřadnicích. Druhým topicem je



Obrázek 5.1: Ukázka interaktivního ukazatele a *MotionPlanning* panelu.

/servo_server/delta_joint_cmds, který přijímá zprávu typu *JointJog*. Pro naše potřeby jsme zvolili první topic */servo_server/delta_twist_cmds*.

Převod vstupů z joysticku na výše zmíněné požadované rychlostní data, neboli převod zprávy typu *Joy* na zprávu typu */servo_server/delta_twist_cmds* se provádí ve funkci *joy_to_twist.py*. Provázanost jednotlivých tlačítek s rychlostními příkazy jsme určili manuálně pro námi používaný joystick.

Další nutné kroky pro ovládání manipulátoru již dělá Servo.

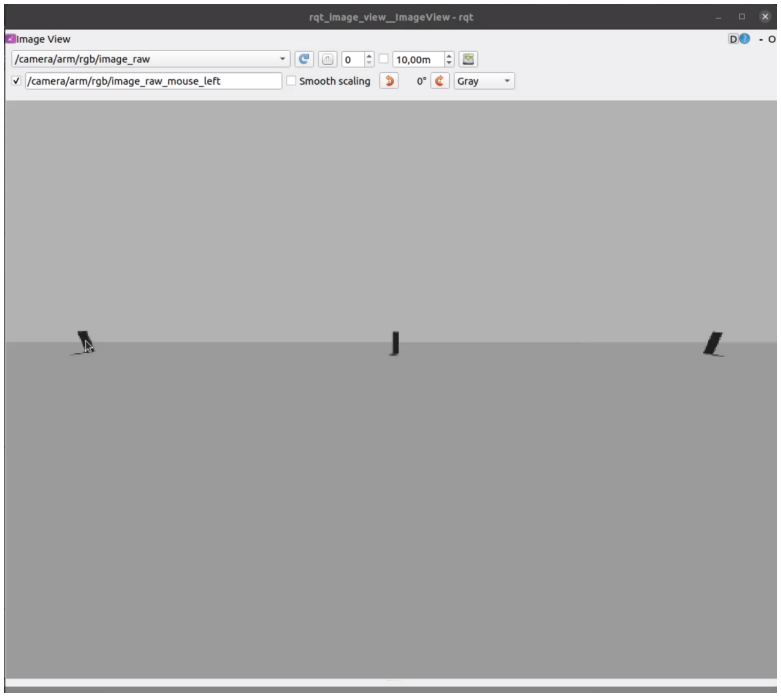
5.3 Zvedání předmětu

Úkol je rozdělen do tří hlavních částí. První část zpracovává data z kamer a dodává informaci o poloze předmětu, funkce této části jsou implementovány v souboru *process_object_position.py*. Úkolem druhé části je dojít k předmětu, implementace potřebných funkcí je v *move_to _object.py* a třetí část pohybuje s manipulátorem a zvedá předmět, funkce najdeme v *pickup.py*. Celkový proces můžeme spustit pomocí *pickup.launch*.

5.3.1 Detekce předmětu

Detekce předmětu využívá balíku *rqt_image _view*, který zobrazuje výstup kamery, která je umístěna v chapadle manipulátoru. Toto okno detekuje kliknutí levého tlačítka myši a publikuje souřadnice kliknutého pixelu na topic */detected_object_position*.

V programu sledujeme tento topic spolu s pointcloudem hloubkové kamery. Když uživatel klikne na předmět, získáme data daného pixelu z pointcloudu. Abychom dostali souřadnice předmětu v souřadnicovém systému simulace, je třeba tento bod transformovat. K tomu využíváme knihovnu *tf*, která pomocí funkce *transformPoint* tento bod transformuje. Tyto souřadnice předáváme



Obrázek 5.2: *rqt_image_view* okno, které detekuje kliknutí myši.

druhé časti.

5.3.2 Dojít k předmětu

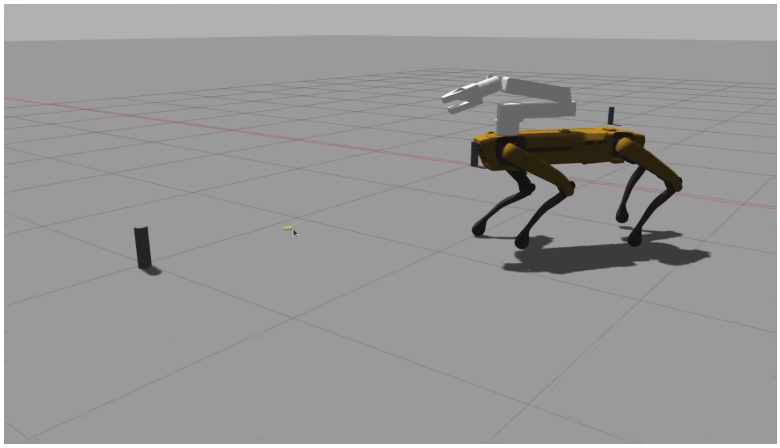
Po obdržení polohy předmětu spustíme WBC a Spota ovládáme pomocí rychlostních příkazů, které publikujeme na topic */twist_cmd*. Nejdříve se Spot natočí směrem k předmětu, a poté se vydá rovně, dokud nedojde do dostatečné vzdálenosti od předmětu, přičemž si pořád kontroluje úhel natočení. V okamžiku jak je dostatečně blízko, WBC vypneme a předáme třetí části informaci, že je Spot připraven pro úchop.

5.3.3 Zvednutí předmětu

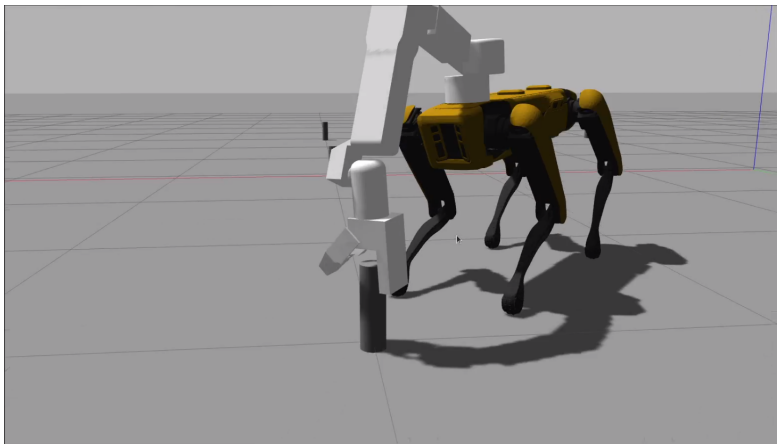
Manipulátor ovládáme pomocí balíku *moveit_commander*, který se stará o propojení s MoveItem. Manipulátor pomocí funkce *set_named_target* dostaneme do polohy, ve které vidí na předmět a pošleme žádost první části o novou, přesnější detekci. Uživatel znova klikne levým tlačítkem myši do obrázku na předmět, který chce zvednout a pošle třetí části přesné umístění předmětu.

Danou polohu převedeme do souřadnicového systému manipulátoru a nastavíme mírný odstup od předmětu. Funkci *set_pose_target* předáme požadovanou pozici a orientaci koncového efektoru. Pomocí funkce *go* s maniupulátorem dojedeme na požadovanou pozici, pokud nalezne řešení a nakonec posuneme požadovanou polohu směrem k předmětu a dojedeme k předmětu.

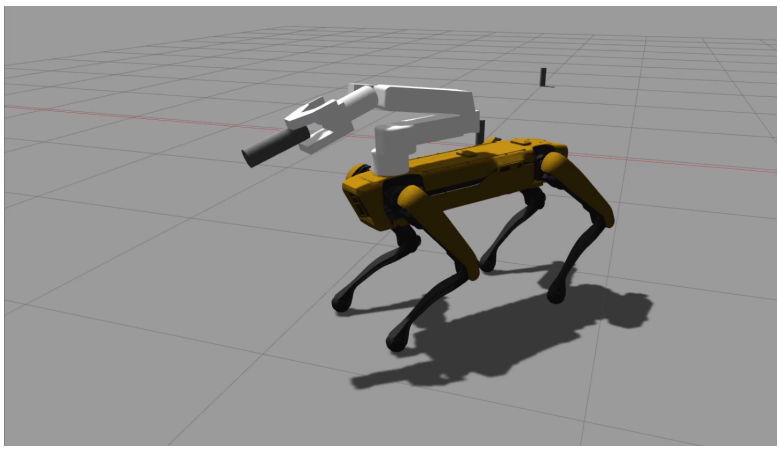
Manipulátor předmět stiskne a přesune manipulátor polohy *stow*.



Obrázek 5.3: Spot se otočil směrem k předmětu a pohybuje se do dostatečné blízkosti.



Obrázek 5.4: Spot uchopuje předmět



Obrázek 5.5: Spot uchopil předmět a zvedl ho to polohy *stow*

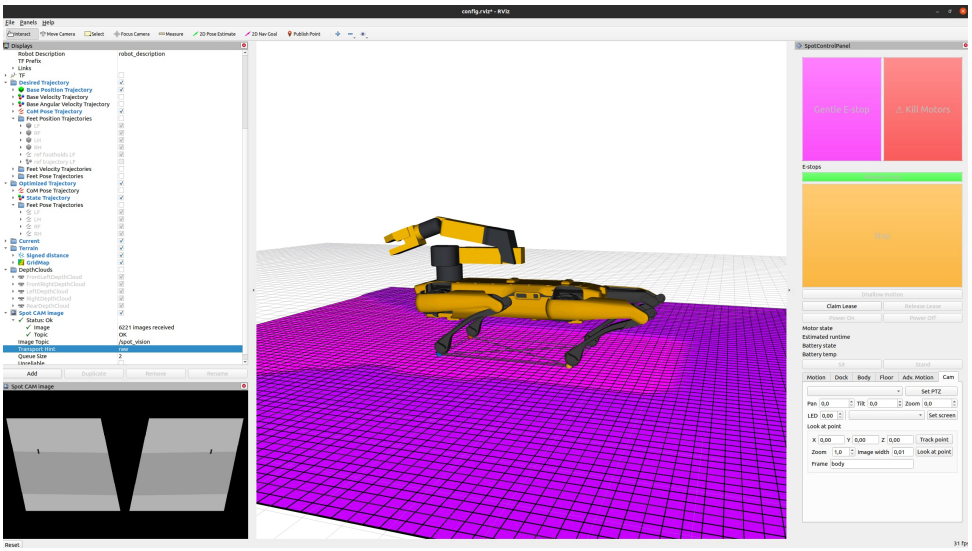
5.4 Další možné ovládání přes C++ a Python

MoveIt dále nabízí uživatelské rozhraní pro pohyb robota pomocí *Move Group* jak pro Python, tak pro C++. Toto rozhraní poskytuje ovládání pro většinu operací, které běžný uživatel potřebuje. Umožňuje nastavení pozice chapadla nebo natočení jednotlivých kloubů, vytváří pohybový plán, ovládá robota, přidává objekty do prostředí simulace a připevňuje/odpojuje tyto objekty k robotovi. *Move Group* rozhraní komunikuje přes komunikační kanály, servisy i akce v Rosu. Uživatel má tedy možnost ovládat manipulátor Spota i *Move Group* rozhraní.

Kapitola 6

Srovnání API skutečného a simulovaného Spota

Uživatelské prostředí pro ovládání skutečného Spota mimo ovladač a tablet umožňuje *spot_ros* [6], který umožňuje ovládat Spota přes počítač pomocí ROSu a RVizu. Abychom vytvořili co nejpřesnější API i pro Spota v simulaci, vycházeli jsme z tohoto repozitáře. Srovnali jsme RViz rozhraní a doplnili jej o chybějící části, což byl ovládací panel Spota, obraz ze předních kamer, vyobrazené okolí bodů z hloubkových kamer a transformace jednotlivých ramen a kamer. Pro přehlednost jsme zobrazení transformací vypli, nicméně uživatel si je může zobrazit zvolením *TF* v levém panelu RVizu. Finální konfiguraci RVizu můžeme vidět na následujícím obrázku.



Obrázek 6.1: Konfigurace RVizu doplněna o konfiguraci u reálného Spota

6.1 Seznam funkcí API v *spot_ros*

Vytvořené topicy v simulaci jsme doplnili o seznam komunikačních kanálů, které najdeme u skutečného spota. Některé komunikační kanály jsou již

vytvořené pod jiným názvem. V takovém případě data z jednoho kanálu na druhý přepublikujeme. Další kanály pro naše účely nepotřebujeme, v tomto případě jsme je nechali prázdné nebo s konstantními daty pro možné další rozšíření. Kanály, které potřebují ovlivňují chování Spota jsme propojili.

Kapitola 7

Výsledky a srovnání simulovaného a reálného Spota

7.1 Srovnání pohybu těla Spota

Srovnání pohybu těla bylo tématem bakalářského projektu, pro kompletní srovnání simulovaného a reálného Spota to však zmíníme. Srovnávání pohybu probíhalo ve třech různých situacích. Chůze po rovině, chůze přes paletu a chůze do schodů a dolů. V simulaci jsme vytvořili mapy podle parametrů palety a schodů. Pomocí nahraných rychlostních příkázů jsme ovládali simulovaného Spota a sledovali jsme jeho trajektorii v prostoru a natočení jednotlivých kloubů v daném čase.

7.1.1 Srovnávní pohybu Spota při chůzi do schodů

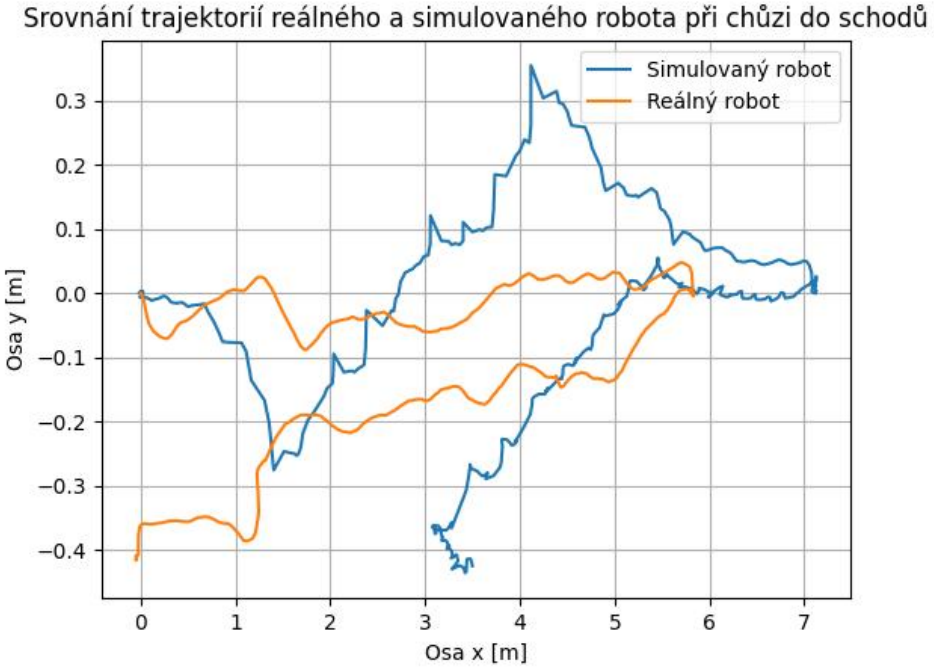
Pro srovnání pohybu v této práci jsme použili data pouze z chůze po schodech.

7.1.2 Výsledek

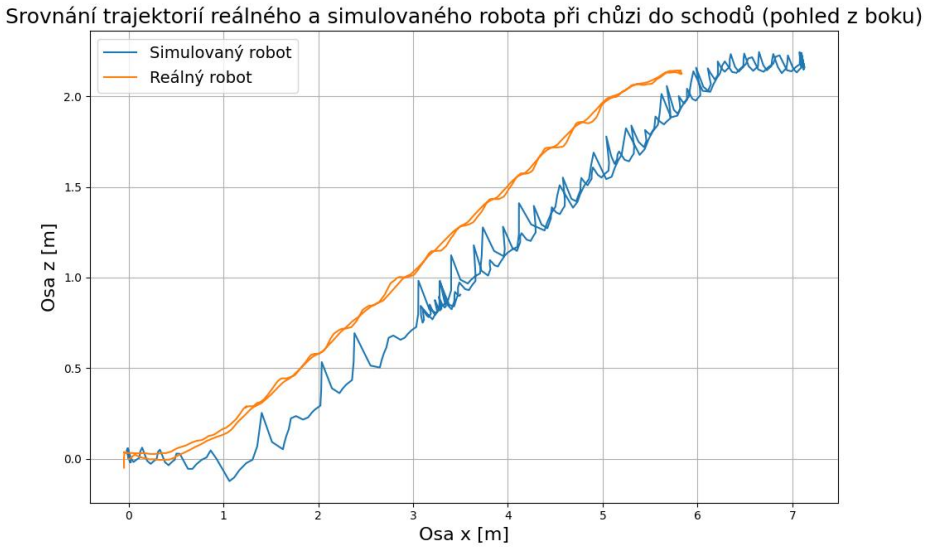
Na grafech můžeme vidět, že simulovaný robot od reálného se pro stejný vstup mírně liší. Možný důvod vzniku odchylky může být např. jiný styl chození do schodů, kde reálný robot chodí vždy po jednom schodu, simulovaný robot občas vyhodnotí, že je lepší jeden překročit. Z pohledu seshora můžeme vidět, že se simulovaný robot v jeden moment natočil a do schodů poté nešel kolmo. Z druhého grafu vidíme, že reálný Spot si mnohem lépe udržuje výšku těla a na posledním grafu můžeme vidět, že pozice natočených kloubů se také o něco liší.

7.2 MoveIt Servo

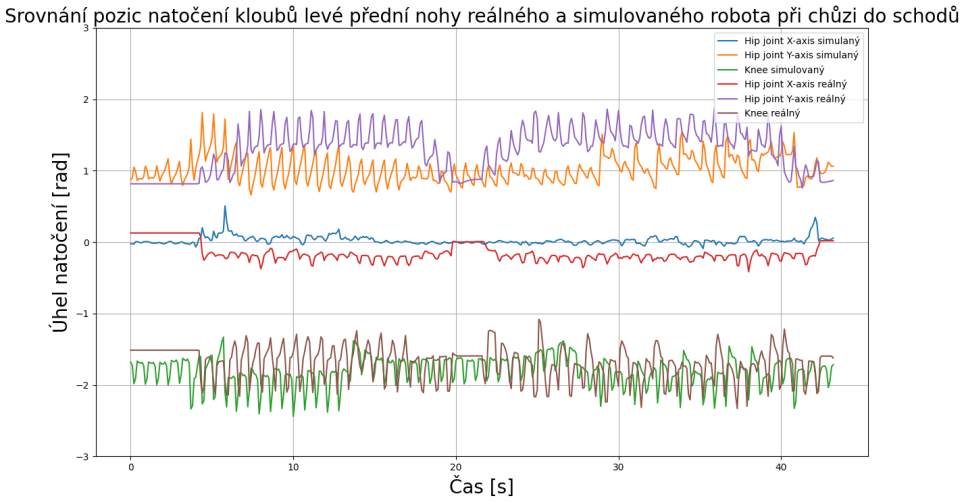
Jak jsme již zmínili, ze začátku bylo potřeba v *joint_group_effort_controller* kontroléru získat správné hodnoty PID regulátorů pro jednotlivé klouby. Důvod byl ten, že při přepnutí kontroléru z *arm_controller*, který se používá pro klasické úlohy MoveItu, na *joint_group_effort_controller* kontrolér pro ovládání pomocí Serva v reálném čase, manipulátor mírně poklesl, manipulátor



Obrázek 7.1: Pohled na trajektorii ze shora v osách x a y



Obrázek 7.2: Pohled na trajektorii z boku v osách x a z



Obrázek 7.3: Srovnání úhlů natočení jednotlivých kloubů reálného a simulovaného robota pro případ chůze do schodů.

se roztřepal a při jakémkoli dalším požadavku o pohyb klesal dolů, dokud nedopadl na zem úplně. Hodnoty PID regulátorů jsme také ladili přes *rqt_gui* a dosáhli jsme dobrých výsledků. Manipulátor se nyní pohybuje volně všemi směry a otáčí se podle všech os, ovšem někdy se dostane do bodu, ze kterého začne opět padat k zemi každým požadavkem o pohyb.

7.3 MoveIt plánování a pohyb

Pohyb pomocí MoveItu je velice spolehlivý. MoveIt poskytuje zpětnou vazbu, zda plánování a vykonání trajektorie proběhlo úspěšně. Pokud manipulátor nenajde trajektorii do požadovaného cíle, mohlo nastat několik situací. První situace nastává když požadovaná cílová pozice je taková, že některá ramena manipulátoru, popřípadě rameno s tělem jsou v kolizi. Druhá situace nastane, když požadovaná cílová pozice koncového efektoru je mimo dosah manipulátoru a nemůže se do cíle dostat za žádných okolností. Třetí situace nastává, když se vypočítaná trajektorie vykoná, ale koncový efektor má od požadované pozice větší odchylku, než je v konfiguraci povolena.

K otestování, že hodnoty PID regulátoru pro jednotlivé klouby v kontroléru *arm_controller* a pro kloub pro otvírání chapadla v kontroléru *gripper_controller* jsou dostačující a manipulátor se dostane do požadovaných pozic, jsme použili *MotionPlanning* panel. Vyzkoušeli jsme několik náhodných cílových pozic pro oba kontroléry a sledovali, zda plánování nebo vykonání pohybu neselže.

Kapitola 8

Závěr

Cílem této bakalářkské práce bylo navrhnout a implementovat aplikační programové rozhraní (API) pro ovládání robota Spot v simulátoru Gazebo, včetně podpory pro ovládání integrovaného manipulátoru. Při implementaci byl kladen důraz na to, aby byla zachována co možná největší shoda s již existuicími API pro ovládání reálného Spota.

V rámci bakalářské práce jsme propojili model těla Spota a model manipulátoru, také jsme je upravili tak, aby odpovídaly skutečnému modelu. Vytvořili jsme propojení manipulátoru s MoveItem a tím umožnili ovládat manipulátor pomocí rozhraní MoveItu v RViz a pomocí balíku Move Group uživatel má nyní možnost si definovat vlastní funkce pro manipulaci s manipulátorem a to jak v C++ tak v Pythonu. Manipulátor byl propojen s MoveIt Servem a tím jsme umožnili ovládání manipulátoru v reálném čase, také jsme umožnili jej ovládat pomocí joysticku. Implementovali jsme zvedání předmětu, jakožto jednu z funkcí, které Spot nabízí.

Nicméně při implementaci jsme narazili na několik překážek. Nepodařilo se nám přejmenovat klouby těla Spota. Pro nízkoúrovňové řízení se využívá MPC, který závisí na pojmenování jednotlivých kloubů. Dále při pohybu manipulátoru pomocí MoveIt Serva se stává, že v určitých polohách se nedaří koncovým efektorem hýbat všemi směry a manipulátor pozvolna klesá k zemi. Při zvedání objektu se nevytváříme kolizní objekt pro plánovač a občas se stane, že Spot manipulátorem do předmětu narazí a nezvedne ho. Nakonec při snaze o propojení ROS API [6] jsme narazili na problém, že většina funkcí je úzce provázaná s funkcemi a klienty skutečného Spota.

Další práce by mohla zahrnovat implementaci nových funkcí API, které by zahrnovaly komplexnější úkoly pro kontrolu těla i manipulátoru, jako může být sběr a třízení předmětů. Velkou výhodou simulace je, že nestojí to co reálný Spot a uživatel má možnost vyzkoušet vše, co ho napadne.

Na závěr bychom rádi podotkli, že celý kód je open-source [5].

Příloha A

Literatura

[1] Dynamics, Boston: *Oficiální API pro robota Spot.* https://dev.bostondynamics.com/. Version: 2024

[2] esterRay: *Model manipulátoru a jeho propojení s MoveItem.* https://github.com/estherRay/Spot-Arm. Version: 2024

[3] Jon, Jakub: *Nízkoúrovňové řízení robota Spot.* https://github.com/projectatctu/ocs2_fun. Version: 2024

[4] MoveIt: *MoveIt Servo.* https://github.com/moveit/moveit/tree/master/moveit_ros/moveit_servo. Version: 2024

[5] Oborný, Tomáš: *Implementace manipulátoru.* https://github.com/projectatctu/ocs2_fun/tree/high_five. Version: 2024

[6] Staniaszek, Michal: *ROS API pro reálného Spota.* https://github.com/heuristicus/spot_ros. Version: 2024