Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Teleoperácia humanoidného robota NICO vo virtuálnom prostredí

Bakalárska práca

2024
Adam Húserka

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Teleoperácia humanoidného robota NICO vo virtuálnom prostredí

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.
Konzultant:	Mgr. Michal Vavrečka, PhD.

Bratislava, 2024
Adam Húserka

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Adam Húserka
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Teleoperácia humanoidného robota NICO vo virtuálnom prostredí
Teleoperation of humanoid robot NICO in virtual environment

Anotácia: Jedným zo spôsobov využitia robotov je teleoperácia, teda priame ovládanie človekom vykonávajúcim motorické úkony. Vo fyzickej realite si to vyžaduje sledovanie ľudského pohybu. Teleoperáciu je možné využiť aj vo virtuálnom prostredí, ktoré umožňuje sledovanie ľudských rúk pomocou ovládačov, ktoré poskytujú informáciu o polohe, orientácii a iné, simulovanému robotovi.

Cieľ:
1. Oboznámte sa s robotickým simulačným prostredím iGibson a importujte doň model humanoidného robota NICO.
2. Pripravte robota NICO na teleoperáciu vo VR a otestujte jej funkčnosť.
3. Zosumarizujte vlastnosti (výhody a nevýhody) nového simulátora iGibson.

Literatúra:
Li C. et al. (2021) iGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks. 5th Annual Conference on Robot Learning. https://openreview.net/forum?id=2uGN5jNJROR
Choi H. et al. (2021) On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. Proceedings of the National Academy of Sciences. https://www.pnas.org/doi/epdf/10.1073/pnas.1907856118
Lei Y., Su Z., Cheng C. (2023) Virtual reality in human-robot interaction: Challenges and benefits. Electronic Research Archive. https://doi.org/10.3934/era.2023121

Vedúci: prof. Ing. Igor Farkaš, Dr.
Konzultant: Mgr. Michal Vavrečka, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 14.10.2023

Dátum schválenia: 01.11.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne vyhlasujem, že celú bakalársku prácu na tému „Teleoperácia humanoidného robota NICO vo virtuálnom prostredí“, vrátane všetkých jej príloh a obrázkov, som vypracoval/vypracovala samostatne, a to s použitím literatúry uvedenej v priloženom zozname a nástrojov umelej inteligencie. Vyhlasujem, že nástroje umelej inteligencie som použil v súlade s príslušnými právnymi predpismi, akademickými právami a slobodami, etickými a morálnymi zásadami za súčasného dodržania akademickej integrity a že ich použitie je v práci vhodným spôsobom označené.

Poďakovanie: Chcel by som sa poďakovať môjmu školiteľovi prof. Ing. Igorovi Farkašovi, Dr., môjmu konzultantovi Mgr. Michalovi Vavrečkovi, PhD., mojej rodine a všetkým mojim blízkym priateľom za ich rady, trpezlivosť a podporu. Ďalej osobitne ďakujem Mgr. Michalovi Vavrečkovi, PhD. za poskytnutie modelov robota NICO.

Abstrakt

Táto bakalárska práca sa zameriava na teleoperáciu humanoidného robota NICO v realistickom virtuálnom prostredí. Teleoperácia je kľúčovým konceptom umožňujúcim priame ovládanie strojov na diaľku človekom a má široké uplatnenie v rôznych oblastiach, od riadenia hračkárskych autíčok po ovládanie sofistikovaných robotov na misiách vo vesmíre. Využitie kombinácie technológií virtuálnej reality a simulačných prostredí ponúka cenovo dostupnú a bezpečnú možnosť vývoja teleoperačných rozhraní. Cieľom je oboznámiť sa so simulačným prostredím iGibson, integrovať do neho model robota NICO, vyvinúť a otestovať rozhranie, ktoré umožňuje ovládanie tohto robota na diaľku prostredníctvom virtuálnej reality, a zhodnotiť prácu v simulátore s dôrazom na vyvinuté rozhranie.

Kľúčové slová: teleoperácia, humanoidný robot, virtuálna realita, robot NICO, iGibson

Abstract

This bachelor thesis focuses on the teleoperation of the humanoid robot NICO in a realistic virtual environment. Teleoperation is a key concept that enables the direct remote control of machines by humans and has wide applications in various fields, from controlling toy cars to operating sophisticated robots on space missions. The use of a combination of virtual reality and simulation technologies offers a cost-effective and safe way to develop teleoperation interfaces. The objective is to become familiar with the iGibson simulation environment, integrate the NICO robot model into it, develop and test an interface that allows remote control of this robot through virtual reality, and evaluate the work in the simulator with an emphasis on the developed interface.

Keywords: teleoperation, humanoid robot, virtual reality, robot NICO, iGibson

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Technológie ľudstvu odpradávna pomáhajú v najrôznejších oblastiach našich životov. Za vznik týchto technológií vďačíme generáciám priekopníkov, ktorí ich posúvali od ľudskej myšlienky až po ich fyzickú realizáciu. Medzi takéto technológie patria, a v budúcnosti stále budú patriť, teleoperácia, virtuálna realita a robotické simulácie. Teleoperácia je fascinujúca technológia, ktorá nám umožňuje rozširovať naše schopnosti ďaleko za hranice našich fyzických schránok. Vďaka nej môžeme vykonávať úlohy na miestach, kde je ľudské telo príliš krehké, alebo kam sa nevieme efektívne dostať. Svet, v ktorom môžeme preskúmavať hlbiny mora či vzdialeného vesmíru, alebo operovať pri nebezpečných reaktoroch pomocou strojov, je dnes už realitou.Tieto úlohy museli byť doposiaľ vykonávané iba v reálnom svete, avšak v posledných desaťročiach boli ľudstvu sprístupnené nové technológie, ktoré umožňujú ich virtuálne vykonávanie vo svete simulovanom. Virtuálna realita (VR) poskytuje prostredie, v ktorom môžeme interagovať s digitálnymi objektmi a simuláciami takmer rovnako, ako by sme interagovali s reálnym svetom. To nám umožňuje vykonávať komplexné úlohy a testovať rôzne scenáre bez rizika fyzického poškodenia alebo nákladov spojených s reálnymi experimentmi. Robotické simulácie umožňujú presné modelovanie správania a pohybu robotov v rôznych prostrediach. Tieto simulácie môžu byť použité na testovanie a optimalizáciu algoritmov riadenia, plánovania trás a interakcie s objektmi. Simulované prostredia môžu byť prispôsobené na rôzne scenáre, od priemyselnej automatizácie po prieskum vesmíru. Kombinácia VR a robotických simulácií teda umožňuje vývoj a testovanie robotických systémov v bezpečných a kontrolovaných podmienkach. Samozrejme, tieto technológie majú aj svoje obmedzenia, ktoré doteraz neboli vyriešené. Technológia virtuálnej reality je limitovaná hardvérom, pričom potreba použitia komponentov, ako sú náhlavný displej alebo ovládače, obmedzuje mieru, akou nám táto realita pripadá reálna, a zároveň nás obmedzuje vo vykonávaní akcií, ktoré by sme chceli vykonať slobodne, bez limitácií našich tiel. Technológia robotickej simulácie čelí výzvam súvisiacim s výpočtovým výkonom a komplexnosťou, čo vedie k zjednodušeným reprezentáciám simulovaných svetov. Modelovanie presného správania a interakcií robotov v simulovanom prostredí môže byť zložité, čo vedie k nedokonalostiam a zjednodušeniam, ktoré môžu ovplyvniť výsledky experimentov. Napriek týmto nedostatkom sú tieto nové technológie stále úžasným výdobytkom dnešnej doby, ktoré budú časom stále viac zdokonaľované. Na

tomto všetkom staviame našu prácu. Cieľom tejto práce je preskúmanie nového simulačného prostredia iGibson, implementácia teleoperačného rozhrania pre robota NICO využívajúceho virtuálnu realitu, jeho otestovanie a zhodnotenie práce v tomto novom prostredí. Našou ambíciou je vytvorenie kompaktnej analýzy tohto nového prostredia a položenie základov v oblasti teleoperácie robota NICO.

Kapitola 1

Úvod do problematiky

1.1 Teleoperácia

Zavedieme pár pojmov, ktorých definíciu je pre ďalšie čítanie práce potrebné chápať. **Operátor** je človek, ktorý monitoruje ovládaný stroj a vykonáva potrebné riadiace úkony. **Teleoperácia** znamená ovládanie stroja alebo systému na diaľku. **Robot** je akýkoľvek automaticky ovládaný stroj, ktorý nahrádza ľudskú prácu. Svojím vzhľadom alebo funkciou sa nemusí podobať človeku. **Teleoperátor** je stroj alebo robot, ktorý je ovládaný na diaľku. **Agent** je entita alebo systém, ktorý je schopný autonómneho správania a rozhodovania na základe vnímania svojho okolia a vnútorných stavov.

Ďalej hovoríme o histórií teleoperácie.

1.1.1 História

Ľudia vykonávali teleoperačné úlohy už od dávneho praveku. Jednou z najstarších takýchto úloh bola manipulácia s ohňom. Na usporiadanie dreva alebo iného paliva na správne miesto sa zvyčajne používala ľudská ruka, ktorá bola vtedy najlepším nástrojom na presnú manipuláciu. Po zapálení prostredie ďalej neumožňuje používanie ľudských rúk kvôli vysokým teplotám, ktoré by mohli poškodiť tkanivo ruky a ľudia boli nútení používať vhodnejšie nástroje na ďalšiu manipuláciu ako napríklad drevené palice. Toto používanie nástrojov na diaľku sa dá kategorizovať ako jedna z najstarších manipulačných úloh na diaľku, alebo inak povedané jedna z najstarších teleoperačných úloh.

Časová os vývoja teleoperácie začína v polovici 40-tych rokov 20. storočia, keď Goertz vytvoril prvý mechanicky ovládaný teleoperátor master-slave (Sheridan, 1989). Goertz and Thompson (1954) vylepšili tento dizajn, keď v roku 1954 použili elektrický polohový servomechanizmus na dosiahnutie mechanického oddelenia master a slave systému. V 60-tych rokoch sa zvýšil záujem o túto oblasť, čo viedlo k niekoľkým experimentom skúmajúcim vplyv oneskorení v teleoperácii (Sheridan and Ferrell, 1963). Na

riešenie problému oneskorení bol vyvinutý dozorný režim (Ferrell and Sheridan, 1967), ktorý inšpiroval dlhú radu výskumov zameraných na vytváranie nových softvérových jazykov pre teleoperáciu a vizuálne vylepšenia pomocou prediktívnych displejov. Od polovice 80-tych rokov sa začali objavovať pokročilejšie teoretické metódy riadenia, ako analýza založená na Lyapunovej teórii (Miyazaki et al., 1986) a interné virtuálne modely (Furuta et al., 1987). Na prelome 80-tych a 90-tych rokov sa uplatnila teória sietí prostredníctvom reprezentácie impedancie, hybridnej reprezentácie, rozptylovej teórie a riadenia založeného na pasivite. Tento prístup umožnil stabilnú teleoperáciu s časovým oneskorením. Výskumy zamerané na transparentnosť ukázali potrebu obojsmerného prenosu sily a rýchlosti (Lawrence, 1992; Yokokohji and Yoshikawa, 1994). V polovici 90-tych rokov sa objavili viaceré výsledky využívajúce H_∞ teóriu, v čase keď sa začal používať internet na komunikáciu.

Potreba efektívnej manipulácie objektov vzdialených od človeka je trvalá a pretrváva aj v súčasnosti. Od manipulácie s ohňom sa teleoperácia výrazne rozvinula a je využívaná vo viacerých oblastiach života ako napríklad v priemysle, medicíne a vesmírnom výskume. Tak ako sa technológia rozvíja, aj možnosti teleoperácie sa neustále rozširujú, čím umožňujú ľuďom manipulovať s objektmi a vykonávať úlohy z diaľky s väčšou presnosťou a efektivitou ako kedykoľvek predtým.

1.1.2 Aplikácie teleoperácie

Vesmírna explorácia

Vesmír je veľmi vhodné prostredie pre teleoperačné aplikácie. Ak chceme robiť vesmírnu exploráciu na iných vesmírnych telesách než na našej Zemi, na ktorej oproti vesmíru panujú ešte pohostinné podmienky, tak nie je vhodné požadovať fyzickú prítomnosť ľudí na operovanie exploračných nástrojov a vozidiel. Ľudia sú krehké bytosti a na ich prežitie vo vesmíre je im potrebné dodávať dostatočné množstvo esenciálnych produktov a surovín, čo je v tejto dobe ešte stále náročné a drahé. Je preto oveľa efektívnejšie a bezpečnejšie vysielať za účelom explorácie robotov, satelity a sondy, ktoré sú operované na diaľku alebo len pasívne zbierajú informácie o prostredí.

Podľa Lichiardopol (2007) sa vesmírne aplikácie delia na tri skupiny:

1. **Roboty skúmajúce vesmír** - Rozdeľujú sa na kategórie pristávacích robotov, prieskumných sond a prieskumníkov hlbokého vesmíru.
2. **Satelity** - Satelitná komunikácia, GPS alebo predpovede počasia sú len pár z mnohých uplatnení, na ktoré ľudia satelity používajú.
3. **Vesmírne robotické ramená** - Pre medzinárodnú vesmírnu stanicu bolo vyrobených niekoľko robotických ramien. Použitím ramien je znížená potreba práce

ľudí vo vesmírnom priestore okolo stanice a teda sa znižuje aj celková miera nebezpečenstva pre posádku.

Vojenské aplikácie

V moderných vojenských operáciách je zber informácií pomocou teleoperovaných vozidiel bežnou praxou. Namiesto nasadenia vojakov do rizikových situácií na frontových líniách sa využívajú teleoperované prieskumné vozidlá, ktoré sú ovládané na diaľku. Týmto spôsobom sa znižuje riziko strát na životoch (Lichiardopol, 2007).

Podľa Lichiardopol (2007) existuje viacero typov teleoperovaných vozidiel, medzi ktoré patria:

- **Bezpilotné vzdušné vozidlá (UAV)** - Moderné UAV sú riadené diaľkovo pomocou rádiových alebo satelitných spojení.
- **Bezpilotné pozemné vozidlá (UGV)** - Tieto vozidlá sú široko využívané vo vojenských operáciách na prieskum, čistenie ciest a detekciu mín. Vojenské UGV sú často vybavené najmodernejším teleoperačným vybavením, ktoré poskytuje optimálnu spätnú väzbu počas rýchlych a nebezpečných operácií.

Bezpečnostné aplikácie

Podobne ako v armáde, aj v oblasti bezpečnosti sa využívajú teleoperované systémy. S rastúcim počtom teroristických útokov väčšina policajných oddelení vytvorila pyrotechnické jednotky na deaktiváciu bômb. V tomto kontexte sú veľmi užitočné teleoperované vozidlá vybavené robotickými ramenami a nástrojmi na deaktiváciu bômb (Lichiardopol, 2007).

Podvodné aplikácie

Podvodné operácie patrili medzi prvé mobilné aplikácie, kde sa začali uplatňovať teleoperačné technológie. V súčasnosti predstavujú na diaľku ovládané vozidlá (ROV) pravdepodobne najväčší komerčný trh pre mobilné teleoperačné zariadenia. ROV sa používajú na prieskum, inšpekcie, oceánografiu a rôzne jednoduché manipulačné úlohy, ktoré kedysi vykonávali potápači. ROV sú zvyčajne pripojené k povrchovej lodi a ovládané pomocou videomonitorov a joystickov. Najnovšie systémy dokážu vykonávať aj autonómne úlohy, ako je udržiavanie polohy alebo sledovanie trasy (Lichiardopol, 2007).

Lesnícke a banícke aplikácie

Lesníctvo a baníctvo patria medzi najnáročnejšie a najrizikovejšie povolania. Mnohé nebezpečenstvá, ako padajúce stromy, ťažký terén a závaly v banských chodbách, sú pre tieto oblasti typické. Využitie teleoperovaných zariadení však prináša revolúciu

do týchto odvetví tým, že výrazne znižuje alebo dokonca úplne eliminuje tieto riziká (Lichiardopol, 2007).

Podľa Lichiardopol (2007) sa teleoperácia v baníctve využíva nasledujúcimi spôsobmi:

- **Banské stroje** - Tunelovacie a vrtné stroje je možné ovládať na diaľku, čo eliminuje potrebu prítomnosti ľudí v nebezpečných oblastiach. Kompletná automatizácia bane si vyžaduje systém dopravných pásov a železničných vozňov, ktoré môžu byť riadené z centrálneho veliteľstva na povrchu.
- **Prieskumné roboty** - Pri hľadaní žíl vzácnych rúd sa používajú roboty, ktoré môžu preskúmavať a odoberať vzorky hornín z najneprístupnejších častí bane.
- **Záchranné roboty** - V prípade banských nešťastí, ako sú závaly, sú nevyhnutné rýchle záchranné akcie. Záchranné tímy využívajú roboty na lokalizáciu a hodnotenie stavu uväznených pracovníkov. Tieto roboty môžu byť taktiež použité na záchranu osôb v budovách poškodených zemetrasením.

1.1.3 Master-Slave teleoperátor

Teleoperátor je komplexný mechatronický systém, ktorého hlavné elementy sú master a slave. Tieto elementy sú prepojené komunikačným kanálom. Systém je na jednej strane prepojený s ľudským operátorom a na druhej strane s prostredím. Komponenty systému môžu dosahovať rôznu úroveň hardvérovej a softvérovej komplexity (Melchiorri, 2013).

Podľa Melchiorri (2013) kľúčové charakteristiky týchto systémov zahŕňajú:

1. Ľudský operátor je súčasťou riadiaceho cyklu pri vykonávaní úloh.
2. Poskytnutie relevantných údajov operátorovi, pokiaľ možno v reálnom čase, čo vyžaduje vhodné používateľské rozhranie a výber správnych signálov. Tieto signály môžu zahŕňať informácie o polohách slave zariadenia, grafické a video dáta, ako aj hmatové alebo akustické informácie, ktoré výrazne ovplyvňujú riadiace a výkonnostné schopnosti celého systému.
3. Komunikačný kanál medzi master zariadením a slave zariadením, ktorý môže predstavovať zdroj problémov, ak sú prítomné časové oneskorenia. Tieto oneskorenia v spätnoväzbovej slučke môžu spôsobiť nestabilitu systému.

Master

Podľa Melchiorri (2013) master systém, tiež nazývaný lokálny systém, je rozhranie, cez ktoré operátor zadáva príkazy pre celý systém. Typické vlastnosti master systému sú:

- Schopnosť prideľovať úlohy slave systému a poskytovať operátorovi informácie o vývoji úlohy. Významnou vlastnosťou master systému je schopnosť poskytovať operátorovi teleprítomnosť, t.j. pocit, že sa priamo podieľa na vykonávaní úlohy. V tejto súvislosti sa používajú rôzne riešenia, od joystickov a konzol cez VR rozhrania až po exoskelety. Tieto zariadenia môžu operátorovi poskytovať rôzne typy signálov, od jednoduchých grafických údajov až po kompletné kinetostatické informácie.
- Schopnosť získavať a spracovávať údaje od operátora aj od slave systému.

Slave

Podľa Melchiorri (2013) slave systém, tiež nazývaný vzdialený systém, je tá časť teleoperátora, ktorá priamo komunikuje s prostredím pri vykonávaní úloh. Podobne ako master, aj slave má špecifické požiadavky, ktoré zahŕňajú:

- Robotický systém na interakciu s prostredím a vykonávanie úloh naplánovaných operátorom. Tento systém musí byť prispôsobený na prácu v konkrétnych prostrediach. Kinematika a dynamika vzdialeného manipulátora sa môže líšiť od lokálneho manipulátora, čo môže spôsobiť problémy pri zabezpečení teleprítomnosti.
- Získavanie a spracovanie signálov. Senzorické schopnosti sú kľúčovou požiadavkou pre slave systém, ktorý je často vybavený videokamerami, senzormi sily a dotyku, snímačmi priblíženia atď.

1.2 Virtuálna realita

Virtuálna realita je technológia, ktorá využíva interaktívne počítačové simulácie na sledovanie pohybov a činností používateľa a prispôsobuje alebo rozširuje senzorickú spätnú väzbu pre jeden alebo viaceré zmysly. Tento proces vytvára pocit, že používateľ je mentálne ponorený alebo prítomný vo vytvorenom virtuálnom prostredí (Sherman and Craig, 2003).

1.2.1 História

Virtuálna realita (VR) má za sebou mnohé roky teoretického výskumu. V posledných rokoch kontinuálne napredovanie technológií spôsobilo to, že zariadenia, ktoré realizovali tento koncept mohli byť sprístupnené aj širokej verejnosti. Ďalej uvádzame pár významných momentov, ktoré znamenali posun dopredu v tejto oblasti.

- **1968** - Ivan Sutherland vytvoril v roku 1968 prvý headset pre virtuálnu realitu, známy ako "Sword of Damocles". Toto zariadenie integrovalo počítačovú technológiu na zobrazenie virtuálnych prostredí a je považované za prvý náhlavný displej (HMD) pre VR (Boas, 2013).

- **1984** - Jaron Lanier založil spoločnosť Visual Programming Languages, ktorá bola jednou z priekopníckych firiem v oblasti VR. Spoločnosť vyvinula a predávala VR produkty ako DataGlove, EyePhone a AudioSphere, ktoré prispeli k napredovaniu VR technológie (Basu, 2019).

- **1987** - NASA vyvinula pokročilé VR zariadenia na výskumné a inštruktážne účely. Tieto zariadenia položili základy pre súčasné VR systémy zlepšením schopností a aplikácií tejto technológie (Basu, 2019).

- **2012** - Palmer Luckey vytvoril Oculus Rift, ktorý zrevolucionalizoval moderné VR zážitky. Kombináciou realistických obrazov s technológiou pohybu rúk, Oculus Rift nastavil štandard pre následné VR headsety a získal významné investície, vrátane Facebooku (Avila and Bailey, 2014).

- **2015** - Microsoft predstavil HoloLens v roku 2015, AR zariadenie, ktoré využívalo priestorové a objektové mapovanie na projekciu vysokokvalitných obrazov a videí do reálneho sveta. Toto zariadenie znamenalo významný pokrok v AR technológii (Noor, 2016).

- **2016** - Valve a HTC vydali HTC Vive v roku 2016, VR headset známy pre svoje pokročilé sledovanie polohy a dvojicu ovládačov, ktoré umožňovali užívateľom pohybovať sa a interagovať vo virtuálnych prostrediach plynulo (Navarro and Sundstedt, 2019).

- **2022** Integrácia s rozšírenou realitou (AR), umožňujúca prechod medzi virtuálnym a reálnym svetom (Pointecker et al., 2022).

- **2024** - Pokroky v umelej inteligencii, umožňujúce realistickejšie interakcie s virtuálnymi postavami a prostrediami (Oumaima et al., 2024).

Technológie VR prešli dlhú cestu od prvých pokusov o vytvorenie alternatívnych realít. Každý krok na tejto ceste prispel k nášmu súčasnému chápaniu a schopnosti vytvárať pohlcujúce virtuálne zážitky. Cesta k ultimátnemu displeju, ktorý by priniesol dokonale simulovanú virtuálnu realitu, je stále otvorená a technológie neustále posúvajú hranice toho, čo je možné.

1.2.2 VR hardvér

Hardvér pre virtuálnu realitu zahŕňa rôzne zariadenia a vybavenie, ktoré sa používajú na vytváranie a prežívanie VR prostredí. Najdôležitejším prvkom je HMD, ktorý sa nosí na hlave a zobrazuje VR prostredie na svojej obrazovke. Na interakciu s virtuálnym prostredím sa používajú ručné ovládače, ako napríklad Oculus Touch alebo Vive wands. Ovládače sú väčšinou schopné haptickej odozvy, ktorá prináša väčšiu mieru realizmu. Niektoré systémy používajú externé kamery na sledovanie polohy HMD, zatiaľ čo iné využívajú *inside-out tracking*, kde kamery sú zabudované priamo v HMD (Lei et al., 2023).

1.3 Použitie simulácie v robotike

Fyzikálne simulátory sú základným nástrojom pre väčšinu výskumu v oblasti robotiky. Bežnou praxou je testovanie a overovanie teoretických metód najprv alebo výlučne v simulátore, pretože roboty samotné sú často drahé, krehké a málo dostupné. Fyzikálne simulátory tieto výzvy prekonávajú tým, že poskytujú lacné prostredie a umožňujú užívateľom prístup k rôznym typom robotov bez rizika ich opotrebenia alebo poškodenia. Použitie simulácií má niekoľko výhod: simulácie môžu prebiehať rýchlejšie ako v realite, čo je obzvlášť dôležité pre prístupy založené na strojovom učení, umožňujú paralelné spustenie a nevyžadujú fyzickú prítomnosť pre resetovanie prostredia (Collins et al., 2021).

Podľa Choi et al. (2021) existujú nasledovné hlavné výhody a nevýhody pri používaní simulácií v robotike:

1.3.1 Výhody použitia simulácie v robotike

Generovanie veľkého množstva tréningových dát

Simulácia ponúka možnosť rýchlo a ekonomicky generovať rozsiahle tréningové dáta pre systémy strojového učenia. Overené simulačné platformy slúžia ako ideálne testovacie prostredia pre vývoj a verifikáciu systémov, ktoré sa učia z chýb a optimalizujú nové správanie. Simulácie môžu tiež podporiť vykonávanie úloh, ktoré boli predtým nerealizovateľné.

Zrýchlenie inžinierskeho návrhového cyklu

Použitie simulácií môže podstatne urýchliť návrhový cyklus a znížiť jeho náklady. Mechanický dizajn a návrh riadenia sú dve z časovo najnáročnejších fáz pri vývoji nových robotov. Simulácie umožňujú rýchle a lacné testovanie rôznych dizajnových alternatív, čím zefektívňujú celý proces vývoja.

Virtuálne testovanie a verifikácia

Simulácie poskytujú bezpečné a plne kontrolovateľné virtuálne prostredie na testovanie a verifikáciu robotických systémov. Týmto spôsobom je možné minimalizovať riziko zlyhaní a zvýšiť bezpečnosť pred nasadením robotov do reálneho sveta.

Vývoj inteligentnejších robotov

Simulácie podporujú vývoj pokročilejších robotov umožnením experimentovania a optimalizácie riadiacich algoritmov a správania. Týmto spôsobom je možné vyvinúť sofistikovanejšie roboty schopné operovať v komplexných a dynamických prostrediach.

Porozumenie interakciám medzi človekom a robotom

Simulácie interakcií medzi človekom a robotom umožňujú testovanie rôznych scenárov a môžu prispieť k zvýšeniu bezpečnosti a efektívnosti spolupráce medzi ľuďmi a robotmi.

1.3.2 Nevýhody použitia simulácie v robotike

Nedostatok riešení

Vytvorenie funkčnej simulačnej platformy pre robotiku vyžaduje rozsiahlu multidisciplinárnu expertízu a neustály záväzok k softvérovému vývoju. Na rozdiel od tradičného počítačom podporovaného inžinierstva je simulácia v robotike menej štruktúrovaná a omnoho viac multidisciplinárna.

Nedostatok zrelých modelovacích jazykov

Súčasné modelovacie jazyky sú stále nedostatočne rozvinuté a ontológia simulácie v robotike sa len pomaly formuje. Existuje potreba zlepšiť kompozíciu modelov a zvýšiť rýchlosť simulácií.

Komplexnosť interakcií

Simulácie musia zahrnúť zložité interakcie medzi robotmi a medzi človekom a robotom. To zahŕňa simuláciu vnímania, reprezentáciu neštruktúrovaného prostredia a modelovanie komunikácie medzi agentmi.

Simulácia má potenciál významne prispieť k rozvoju robotiky, avšak je potrebné prekonať viaceré prekážky. Pokroky v tejto oblasti môžu urýchliť vývoj a nasadenie inteligentných robotov, čím sa zlepšia ich schopnosti a bezpečnosť v rôznych aplikáciách.

1.3.3 Manipulácia v simulácií

Existuje viacero požiadaviek, ktoré musia výskumné simulátory spĺňať, aby boli vhodné pre simuláciu manipulačných úloh. Simulátory musia obsahovať modely ovládačov pre riadenie polohy, rýchlosti a krútiaceho momentu, pretože tieto spôsoby ovládania sú najčastejšie používané pri fyzických ramenách. Taktiež musia podporovať snímače krútiaceho momentu a vizuálne snímače. Dôležité sú aj vstavané funkcie špecifické pre manipulátory, ako sú riešiče inverznej a doprednej kinematiky a plánovače ciest. Modelovanie deformovateľných objektov sa stáva čoraz dôležitejším, pretože predpoklad, že svet robotiky je úplne rigidný, už neplatí. Ako sa výpočtový výkon stáva dostupnejším, je možné efektívne modelovať deformovateľné objekty, čo pridáva na realistickosti simulácií (Collins et al., 2021).

Súčasný stav v manipulačnej robotike zahŕňa simuláciu viacúlohových alebo viackrokových scenárov, ktoré si vyžadujú presné modelovanie pohybov a kontaktov s tuhými telesami. Potreba stabilnej fyziky, ktorá spoľahlivo zvláda kontakty, zužuje výber vhodných simulátorov (Collins et al., 2021).

1.4 Simulátor iGibson

V ostatných rokoch sa simulačné prostredia rýchlo rozšírili ako účinný a bezpečný spôsob tréningu robotov a interaktívnych agentov. Tieto prostredia umožňujú agentom učiť sa fyzické interakcie, navigáciu pomocou senzorov a plánovacie úlohy. Simulácie poskytujú možnosť meniť senzorové vstupy agentov a stav okolného prostredia, čo je kľúčové pre efektívne učenie agentov (Shen et al., 2021).

Dostupné simulátory často limitujú rozsah úloh a obsahujú len malé, prázdne scény. Niektoré simulátory obsahujú väčšie scény, ako napríklad domy alebo kancelárie, avšak často neumožňujú plnú interakciu so scénou alebo využívajú zjednodušené formy interakcie. Tieto obmedzenia sťažujú prenos naučených stratégií na reálne roboty (Shen et al., 2021).

Simulačné prostredie iGibson 1.0 je navrhnuté pre vykonávanie interaktívnych úloh vo veľkých realistických scénach. Interaktivitu so scénami dosahuje využitím fyzikálneho enginu, ktorý spracováva všetky prvky scény, umožňujúc manipuláciu pevných a artikulovaných objektov, ako aj mobilitu agenta. iGibson 1.0 integruje viacero aspektov robotickej simulácie, ako je fyzikálna simulácia pre interakciu s objektmi a riadenie robotov, kvalitne simulované senzory, integrácia so systémami strojového učenia a realistické vnútorné scény, ktoré odrážajú distribúciu objektov v skutočných domoch (Shen et al., 2021).

Podľa Shen et al. (2021) simulátor prináša tieto novinky:

- Pätnásť plne interaktívnych scén so 108 miestnosťami modelovanými podľa sku-

točných domov, s možnosťou importovať ďalšie rozloženia scén z CubiCasa5K a 3D-Front, čím poskytuje prístup k viac ako 12 000 ďalším plne interaktívnym scénam.

- Realistické virtuálne senzorové signály vrátane fyzikálneho renderera pre RGB obrazy, vykresľovanie normál, hĺbky, bodových mrakov, virtuálnych LiDAR signálov a optického/scene flow.
- Nástroje pre vývoj robotických riešení v simulácii, ako je užívateľské rozhranie pre zber ľudských demonštrácií a plánovače pohybu pre navigáciu a manipuláciu.

Vďaka realistickým virtuálnym senzorovým signálom a mechanizmom náhodnej zmeny domén je možné trénovať robustnejšie a všeobecnejšie senzorimotorické politiky, zbierať ľudské demonštrácie a trénovať politiky imitácie učenia pre manipulačné úlohy. Agenti sa dokážu taktiež učiť vizuálne reprezentácie spojené s interaktivitou scény, čo urýchľuje tréning manipulačných úloh (Shen et al., 2021).

Krátko po vydaní simulátora iGibson verzie 1.0 bola predstavená nová verzia iGibson 2.0 (Li et al., 2021), ktorá priniesla zásadnú novinku v podobe VR rozhrania. iGibson 2.0 obsahuje nové VR rozhranie kompatibilné s dvoma hlavnými komerčnými VR systémami - Meta Quest a HTC VIVE Pro Eye. Všetky stavy sú zaznamenávané počas vykonávania a môžu byť deterministicky znovu prehraté v simulátore, čo umožňuje dodatočné generovanie virtuálnych senzorových signálov alebo vizualizácií interakcií a vývoj riešení na učenie z imitácie. Toto VR rozhranie umožňuje používateľom poskytovať demonštrácie pre nové úlohy a trénovať politiky učenia pre bimanuálne operácie (Shen et al., 2021).

V ďalších častiach si predstavíme všetky časti simulátora a ich interakcie medzi nimi.

1.4.1 Dátové balíčky a assety

Assety obsahujú potrebné súbory na vytvorenie simulačných scén a iných nástrojov, ako napríklad: URDF modely robotov, modely niektorých interaktívnych a artikulovaných objektov na testovanie a kópie dátových balíčkov RBO a YCB pre vkladanie do scén. Kvôli ich veľkej veľkosti nie sú tieto súbory distribuované ako súčasť inštalačných balíčkov, ale ako samostatné balíčky.

Autori simulátora poskytujú na stiahnutie viacero rozsiahlych dátových balíčkov, ktoré okrem iného obsahujú:

- Pätnásť 3D rekonštrukcií skutočných prostredí, ktoré sú plne interaktívne.
- Viac ako 500 statických rekonštrukcií domov a kancelárií.
- Modely objektov s fyzikálnymi a sémantickými anotáciami.

Bez týchto balíčkov by nebolo možné simulátor v očakávanej miere používať bez toho, aby sme si všetky potrebné scény a objekty importovali sami. Dokonca aj už v priložených skriptoch, ktoré testujú funkčnosť simulátora po jeho inštalácií je prítomná kontrola prítomnosti základného dátového balíčku, bez ktorého by táto ukážka funkčnosti nebola možná.

Tvorcovia simulátora poskytujú výborný rozsah simulačných prvkov, ktoré výskumníci používajú na učenie agentov vykonávať najrôznejšie úlohy.

1.4.2 Rozšírené a logické stavy

Objekty v simulátore iGibson majú okrem základných fyzikálnych vlastností typických pre fyzikálne simulácie, ako napríklad poloha, rýchlosť a zrýchlenie, aj špeciálne vlastnosti. Tieto vlastnosti, nazývané rozšírené stavy, sú uvedené v tabuľke 1.1.

Rozšírený stav	**Popis**
Teplota	Reálna hodnota, ktorá sa mení na základe blízkosti k zdroju alebo absorbéru tepla.
Úroveň vlhkosti	Celé číslo, ktoré sa mení pri kontakte s kvapkami vody.
Čistota	Objekty môžu mať na povrchu prach alebo škvrny. Úroveň sa znižuje, keď sa tieto častice odstránia.
Stav prepnutia	Binárny stav, indikujúci, či je objekt zapnutý alebo vypnutý.
Stav rozrezania	Indikuje, či bol objekt, ako napríklad ovocie, rozrezaný na dve polovice.

Tabuľka 1.1: Tabuľka rozšírených stavov, ktoré môžu artikulované objekty nadobudnúť

Vďaka týmto rozšíreným stavom sa zvyšuje interaktivita a realizmus prostredia. Simulátor dokáže poskytovať kontinuálnu realistickú simuláciu prostredia s realistickými interakciami, čo umožňuje ľahšie prenášanie naučených pravidiel na robotov v reálnom svete.

Na základe kinematických a rozšírených stavov iGibson implementuje logické predikáty, ktoré určujú, či sú určité podmienky pravdivé alebo nepravdivé. Pár príkladov logických predikátov je uvedených v tabuľke 1.2.

Nie všetky objekty majú všetky rozšírené stavy. Objekty sú anotované len tými stavmi, ktoré umožňujú zmysluplné a užitočné logické stavy pre daný objekt.

Charakteristiky, ktoré môžu byť objektom anotované, sú uvedené v tabuľke 1.3.

Existujú aj ďalšie charakteristiky objektov, ktoré ovplyvňujú logické stavy iných objektov. Tie sú:

- Zdroj/odtok tepla (ToggledState),

Logický predikát	**Popis**
InsideOf(o1,o2)	Overuje, či je objekt o1 vo vnútri objektu o2.
InContactWith(o1,o2)	Overuje, či sú objekty o1 a o2 v kontakte.
Under(o1,o2)	Overuje, či je objekt o1 pod objektom o2.
OnFloor(o1,o2)	Overuje, či je objekt o1 na podlahe miestnosti o2.
Cooked(o)	Overuje, či teplota objektu „o" prekročila hranicu pre uvarenie, ale nie pre spálenie.
InHandOfAgent(o)	Overuje, či je objekt „o" uchopený agentom.
InReachOfAgent(o)	Overuje, či je objekt „o" do vzdialenosti 2 metrov od agenta.
InSameRoomAsAgent(o)	Overuje, či sa objekt „o" nachádza v rovnakej miestnosti ako agent.

Tabuľka 1.2: Tabuľka logických stavov, ktoré simulátor používa na testovanie stavu prostredia.

Charakteristika	**Popis**
Variteľný	Môže byť uvarený (MaxTemperature, Temperature).
Horľavý	Môže byť spálený (MaxTemperature, Temperature).
Zamŕzajúci	Môže byť zmrazený (Temperature).
Nasiakavý	Môže byť nasiaknutý (WetnessLevel).
Prašný	Môže byť zaprášený (DustinessLevel).
Zašpinený	Môže byť zašpinený (StainLevel).
Prepínateľný	Môže byť zapnutý/vypnutý (ToggledState).
Rezateľný	Môže byť rozrezaný (SlicedState).

Tabuľka 1.3: Tabuľka charakteristík objektov. V zátvorkách sú uvedené jednotlivé s nimi korešpondujúce rozšírené stavy.

- Zdroj kvapiek (ToggledState),
- Čistiaci nástroj,
- Kvapalina,
- Rezací nástroj.

1.4.3 Renderer

Autori simulátora vyvinuli vlastný efektívny a rýchly **MeshRenderer**, ktorý podporuje prispôsobiteľné konfigurácie kamery ako napríklad šírka, výška obrazu a nastavenie veľkosti vertikálneho zorného pola, a rôzne obrazové modality. Podporovaných je šesť obrazových modalít: RGB, povrchové normály, segmentácia, 3D bodové oblaky, optický a scene flow. Tiež sú podporované dva typy LiDAR senzorov: 1-lúčový a 16-lúčový.

1.4.4 Viewer

Autori simulátora vyvinuli ľahko použiteľné rozhranie pre používateľov s názvom **Viewer** na prehliadanie a interakciu so scénami a objektmi. Viewer je virtuálna kamera, ktorá dokáže renderovať a ukazovať používateľovi jej pohľad na scénu. Viewer sa automaticky zobrazí, ak je použitý režim *gui_non_interactive* alebo *gui_interactive* pri inicializácií triedy **Simulator**.

Viewer má tri režimy:

Navigačný režim - Umožňuje užívateľom voľne sa pohybovať po scéne a meniť pohľad kamery. Tento režim je určený na prehliadanie a skúmanie prostredia.

Manipulačný režim - Umožňuje užívateľom interagovať s objektmi v scéne. Môžu objekty uchopiť, presúvať a manipulovať s nimi pomocou vytvárania spojovacích kĺbov, čo je užitočné pre testovanie a simuláciu fyzických interakcií.

Plánovací režim - Umožňuje užívateľom plánovať a vykonávať pohyby robota. Môžu nastaviť ciele pre pohyb základne robota alebo koncových efektorov na dosiahnutie konkrétnych bodov v scéne, čo je užitočné pre simuláciu úloh a navigačných scenárov.

Ovládanie je implementované pomocou klávesnice.

1.4.5 Prostredia

Simulátor iGibson poskytuje niekoľko **prostredí**, ktoré nasledujú rozhranie OpenAI gym, čím uľahčujú trénovanie agentov strojovým učením pre navigačné úlohy a úlohy mobilnej manipulácie. Prostredie inštancuje **scénu**, **objekty** a **roboty** a importuje ich

do obsiahnutého **Simulátora**. Prostredie taktiež inštancuje zoznam **senzorov** (zvyčajne ako súčasť pozorovacieho priestoru) a **úlohu**, ktorá ďalej zahŕňa zoznam **funkcií** odmeny a **podmienok** ukončenia. Najviac všeobecné prostredie je iGibsonEnv.

Senzory

Poskytované sú rôzne typy senzorov ako wrappery okolo renderera. Okrem základných šiestich obrazových modalít, ktoré ponúka renderer, sú poskytované aj modely šumu senzora s náhodným výpadkom na simuláciu reálneho šumu senzorov a uľahčenie prechodu zo simulácie do reálneho sveta. Množstvo šumu je možné kontrolovať.

Úlohy

Úloha je abstrakcia na definovanie toho, čo má agent dosiahnuť v scéne. Rovnaké scény môžu obsahovať viacero úloh. Každá úloha by mala implementovať funkcie na resetovanie scény a agenta, vykonanie kroku úlohy a vrátenie nesenzorických pozorovaní, ako napríklad propriocepcia agenta. Každá úloha by mala tiež obsahovať funkcie odmeny a podmienky ukončenia.

1.4.6 Scény

Štyri typy scén ponúkané simulátorom iGibson sú uvedené v tabuľke 1.4

Typ scény	Popis
EmptyScene	Jednoduchá scéna s rovnými povrchmi a bez prekážok, vhodná na ladenie.
StadiumScene	
StaticIndoorScene	Statické 3D scény.
InteractiveIndoorScene	Plne interaktívne 3D scény.

Tabuľka 1.4: Tabuľka rôznych typov scén v simulátore iGibson.

Scény typov *StaticIndoorScene* a *InteractiveIndoorScene* sú schopné textúrovej a objektovej randomizácie.

Textúrová randomizácia ako názov napovedá náhodne mení textúry a materiály všetkých objektov importovaných v scéne.

Objektová randomizácia zamieňa pôvodné objekty za objekty tej istej kategórie, pričom si nový objekt snaží zachovať polohu a orientáciu pôvodného objektu. Simulátor má implementovanú funkciu na kontrolu korektnosti scény pre prípady, keď nové objekty sú v nechcených kolíziách alebo ak je bránené artikulovaným objektom v žiadanom pohybe.

1.4.7 Objekty

Simulátor iGibson poskytuje širokú škálu objektov, ktoré môžu byť importované do simulátora. Niektoré druhy objektov sú: *YCBObject*, *RBOObject*, *ArticulatedObject*, *URDFObject*, *Cube*, *VisualMarker*, *ShapeNetObject*, *PedestrianSoftObject*.

Každý z týchto objektov má definovanú svoju vlastnú funkciu *load*, ktorá objekt importuje do PyBullet.

1.4.8 Roboty

Simulátor iGibson poskytuje viacero predpripravených robotov na používanie. Roboty využívajú súbory URDF alebo MuJoCo XML na import do PyBullet pomocou funkcie *load*.

Každý robot má definovanú vlastnú triedu. Táto trieda dedí podľa morfológie robota z abstraktných tried definovaných simulátorom. Všetky abstraktné triedy robotov dedia od triedy **BaseRobot**.

Zdedené triedy ponúkané simulátorom sú:

- LocomotionRobot,
- TwoWheeledRobot,
- ManipulationRobot,
- ActiveCameraRobot.

Tieto abstraktné triedy popisujú rôzne aspekty morfológie robota, takže výsledná trieda robota môže dediť z viacerých abstraktných tried. Napríklad robot NICO dedí z tried ManipulationRobot a ActiveCameraRobot.

Každá abstraktná trieda robota implementuje užitočné funkcie na ovládanie a prístup k vlastnostiam robota. Napríklad ManipulationRobot obsahuje funkcie na získanie stavu ramien a implementuje viacero režimov uchopenia, vrátane niektorých zjednodušených režimov uchopenia ako *assisted grasping* alebo *sticky mitten*. Pri vytváraní nových tried robotov na import vlastných robotov sa odporúča dôrazne dodržiavať hierarchiu robotov definovanú autormi.

1.4.9 Simulator

Trieda **Simulator** udržiava inštancie tried **Renderer** a **PhysicsEngine** a poskytuje metódy na importovanie inštancií tried **Scene**, **Object** a **Robot** do oboch a udržiava ich synchronizáciu po celý čas. Ak Simulátor používa režim **gui**, tak štandardne udržiava tiež **Viewer**. Simulátor teda združuje všetky vyššie spomínané časti simulátora iGibson. Jeho úlohou je vykonávať kroky simulácie a synchronizovať renderer a fyziku poskytovanú knižnicou pyBullet.

Kapitola 2

Ciele a metodika práce

Ciele, ktoré chceme v tejto práci splniť, sú:

1. Oboznámenie sa s robotickým simulačným prostredím iGibson, čo zahŕňa pochopenie jeho základných funkcií, možností a obmedzení, a následné úspešné importovanie modelu humanoidného robota NICO do tohto prostredia.
2. Príprava robota NICO na teleoperáciu vo VR, vrátane implementácie jednoduchého teleoperačného rozhrania a následného testovania funkčnosti tohto rozhrania v rôznych situáciách.
3. Sumarizácia vlastností (výhod a nevýhod) nového simulátora iGibson.

   V práci sa sústreďujeme na časti simulátora, ktoré sa týkali importovania a teleoperácie robota NICO.

V ďalších sekciách vysvetľujeme koncepty, s ktorými je potrebné byť oboznámení pre lepšie pochopenie ďalších častí tejto práce, najmä jej výsledkov.

2.1 Import robota do simulátora iGibson

Na demonštráciu postupu všeobecného importu robotov do simulátora iGibson použijeme príklady súborov `robot.py` a `robot.yaml`.

Na to, aby sme akéhokoľvek robota importovali, musíme mať k dispozícii tieto súbory:

- Súbor vo formáte URDF, ktorý definuje kostru celého robota a jej vlastnosti.
- Súbor `robot.py`, v ktorom definujeme nevyhnutné prepojenia medzi simulátorovým systémom a URDF súborom robota.
- Súbor `robot.yaml`, ktorý obsahuje základnú konfiguráciu, ktorá sa používa pri vkladaní modelu robota do scén simulátora.

2.1.1 Súbor URDF

Súbor vo formáte URDF definuje presnú kostru robota. Kostra sa skladá z kĺbov a článkov medzi kĺbmi robota. Je hierarchicky definovaná, takže každý kĺb spájajúci dva články má definované, ktorý článok je jeho rodič a ktorý je jeho potomok.

Podľa druhu kĺbu sa ďalej definujú informácie ako os otáčania a uhlové limity. Existuje viacero druhov kĺbov, ako napríklad: revolute, continuous, prismatic, fixed (ROS.org, 2024). Napríklad model robota NICO používa iba kĺby typov revolute a fixed. Revolute kĺb sa otáča iba okolo jednej osi a má obmedzený rozsah určený hornými a dolnými limitmi. Fixed kĺb ani kĺbom vlastne nie je, keďže všetky stupne voľnosti sú mu odobraté.

Článkom sa definujú ich vizuálne, zotrvačné a kolízne vlastnosti. Meshe častí robota definujú geometriu vizuálom a kolíziám a pri definícií zotrvačnosti definujeme hmotnosť článkov.

Keď simulátorový systém súbor URDF spracuje, tak trieda, ktorá si z tohto spracovania pamätá esenciálne informácie na zobrazovanie a následné náležitosti, týkajúce sa pohybu jednotlivých častí robota, si v simulátore ukladá kĺbové informácie iba o kĺboch, ktoré nie sú typu fixed a/alebo nemajú názov ignore alebo jointfix.

2.1.2 Opis súboru robot.py

V tomto súbore sa definuje trieda pre importovaného robota, ktorá má typicky rovnaký názov ako samotný robot. Táto trieda sleduje definovanú hierarchiu tried robotov v simulátore a dedí od nich podľa špecifikácie importovaného robota. Napríklad trieda `Nico` dedí od tried `ManipulationRobot` a `ActiveCameraRobot`.

V tejto triede sa ďalej definuje:

- Mapovanie častí robota na indexy v riadiacom vektore.
- Poradie kontrolérov.
- Základné polohy kĺbov.
- Nastavenie bodov pre asistovaný úchop.
- Páry článkov, ktoré majú medzi sebou vypnuté kolízie.

Rodičovské triedy

Trieda, ktorú je potrebné vytvoriť, by mala ideálne dediť vlastnosti a metódy od už preddefinovaných robotických tried v simulátore. V simulátore existuje viacero druhov robotických tried, no nás primárne zaujíma trieda `ManipulationRobot`.

Trieda `ManipulationRobot` poskytuje všeobecné rozhranie pre roboty s manipulačnými schopnosťami. Aby bola trieda použiteľná, musí mať robot definované mapovanie

pre jeho končatiny, ktoré chceme použiť na manipuláciu. To znamená, že musíme definovať mapovanie pre každé rameno a efektor robota. Viac o mapovaní sa dozviete v časti 2.1.2. Ďalej je potrebné definovať názvy jednotlivých rúk, ak má robot viac ako jednu ruku, názvy efektorov a nakoniec názvy článkov a kĺbov prstov robota. Ak je toto pre robota zadefinované, zdedené metódy a vlastnosti budú použiteľné na ovládanie novo importovaného robota.

Režimy úchopu

Trieda `ManipulationRobot` umožňuje robotom nadobudnúť tri režimy úchopu: *fyzický*, *asistovaný* a *sticky*.

Ak je režim úchopu nastavený na *fyzický*, neaplikuje sa žiadna asistovaná pomoc pri úchope a spolieha sa na kontaktné trenie a silu prstov.

Pre asistovaný úchop sa definujú začiatočné a koncové body, medzi ktorými sa kontroluje prítomnosť objektu. Ak objekt pretína priamku, ktorá je vytvorená medzi akýmkoľvek začiatočným a koncovým bodom, tak asistovaný úchop bude po zadaní príkazu na uchopenie aktivovaný.

V režime *sticky* sa každý objekt, ktorý sa dotkne prstov a je zadaný príkaz na uchopenie, automaticky prichytí.

Mapovanie

Riadiaci vektor je pole, kde jeho indexy zodpovedajú jednotlivým kĺbom robota. Ako sme už spomenuli, simulátor pri importovaní robota spracuje informácie o kĺboch tak, že kĺby typu fixed sú ignorované. Ostatné kĺby sú podľa poradia, v ktorom boli prečítané z URDF súboru, postupne ukladané do zoznamu vytvoreného simulátorom. Napríklad, ak je prvým revolute kĺbom v URDF súbore *head_y_rjoint* a tento kĺb je aj prvý spracovaný, potom index 0 v riadiacom vektore bude zodpovedať práve tomuto kĺbu. Keď poznáme štruktúru kostry robota 2.1.1, tak vieme definovať mapovanie názvov častí robota (kamera, rameno, efektor) na indexy v riadiacom vektore. Toto mapovanie má potom široké využitie pri riadení pohybu robota.

Kontroléry

Ďalej v súbore priraďujeme každej pohyblivej časti robota kontrolér a ich poradie. Definíciou poradia kontrolérov definujeme kinematickú reťaz, ktorá je potrebná pri simulátorom vykonávaných výpočtoch.

Kontrolér v iGibsone má na starosti vykonávanie pohybov časti robota, pre ktorú je definovaný. Kontroléry majú všeobecnú štruktúru stavanú na vstupoch a výstupoch. Podľa nastavenia kontroléra sa môžu posielať príkazy, ktoré sa budú interpretovať v kontexte pozície, rýchlosti alebo krútiaceho momentu. Napríklad pozičné príkazy môžu

byť zmena uhlu kĺbu, alebo pozícia a natočenie bodu, ktorý sa má dosiahnuť. Výstupy sú už riadiace signály, ktoré simulátor spracuje a kĺby robota sa budú podľa nich hýbať.

Celý robot sa riadi pomocou jedného poľa, ktoré nazývame „akcia". Každý kontrolér má definovaný svoj rozmer vstupu. Podľa rozmerov a poradia jednotlivých kontrolérov sa pole akcie napĺňa príkazmi.

V simulátore sú rôzne preddefinované kontroléry. Tie sú:

- DifferentialDriveController,
- InverseKinematicsController,
- JointController,
- MultiFingerGripperController,
- NullGripperController.

JointController

JointController je definovaný v súbore `joint_controller.py`. Pre naše účely je tento kontrolér používaný na riadenie pozície kĺbov, kde pozícia sa v tomto ponímaní chápe ako uhol kĺbu. Je to priamočiary spôsob nastavenia pozícií kĺbov, ktoré na riadenie nevyžadujú výpočet kinematickej reťaze alebo iný netriviálny spôsob riadenia.

Vstupný príkaz má definovaný rozmer ako počet kĺbov, ktoré kontrolér ovláda. Výstupom je podľa nastavenia kontroléra buď signál na nastavenie pozície kĺbov na hodnotu vstupného príkazu, alebo v delta režime signál, ktorý pripočítal k pôvodnej hodnote kĺbu hodnotu vstupného príkazu.

InverseKinematicsController

InverseKinematicsController je definovaný v súbore `ik_controller.py`. Tento kontrolér používa výsledky z funkcie *calculateInverseKinematics* z knižnice PyBulllet na nastavenie kĺbov robota. Na to, aby sme mohli túto funkciu využiť, musíme najskôr správne zadefinovať s akými kĺbmi pracujeme, a ktorý z nich je koncový efektor.

Kontrolér môžeme nastaviť na viacero režimov. Napríklad nami vybraný režim, *pose_ delta_ ori*, je základný a vo vstupnom príkaze sa posiela delta zmena pozície a orientácie. Táto zmena je uplatnená ako translačná a rotačná transformácia pôvodnej pozície a tá sa stane cieľovou pozíciou, potom sa spustí už spomínaná metóda *calculateInverseKinematics*.

Ostatné kontroléry

Ostatné kontroléry boli pre naše účely nevyužité. Len v rýchlosti popíšeme, aké časti robotov ovládajú ostatné kontroléry a akú funkcionalitu danej časti prinášajú.

DifferentialDriveController ovláda kĺby kolies robota. Tento kontrolér je nevhodný pre naše účely, keďže náš robot je stacionárny a nemá žiadne mobilné možnosti.

MultiFingerGripperController ovláda kĺby všetkých prstov efektora buď binárnym alebo ternárnym spôsobom. V binárnom móde sa napovedane správa binárne a hodnoty príkazov sa pohybujú na intervale $[0, \infty)$. Ak je prichádzajúci príkaz väčší než 0, tak kĺby prstov nastaví na ich horný limit. Ak nie, tak sú nastavené na ich spodný limit. V ternárnom móde sa hodnoty príkazov pohybujú na intervale $[-1, 1]$. Takéto správanie nie je pre naše účely žiaduce, keďže chceme mať možnosť simulovať postupné stupňovanie úchopu.

NullGripperController je tzv. dummy kontrolér, vstupný príkaz má nulový rozmer a výstupný kontrolný signál je tiež prázdne pole, takže pri riadení robota sa v akcií nevyčleňuje pre tento kontrolér žiadna časť akcie.

Základná a reset poloha kĺbov

Každý robot musí mať zadefinovanú základnú polohu všetkých nefixovaných kĺbov. Okrem základnej polohy je možné definovať aj polohu, ktorá sa má dosiahnuť pri resete polohy robota. Ak nedefinujeme osobitnú polohu pre resetovanie, tak sa namiesto nej pri resetovaní používa základná poloha.

Vypnutie kolíznych párov

Ak je potrebné, je možné vypnúť kolízie medzi konkrétnymi článkami robota.

2.1.3 Popis súboru robot.yaml

V tomto súbore sa definuje konfigurácia pre robotov. Keď je robot importovaný do simulátora, tak je načítaný podľa tohto konfiguračného súboru. Základná časť, ktorá je v každom takomto konfiguračnom súbore je časť "robot". V tejto časti sa nachádza konfigurácia pre samotného robota. Nastavujú sa v nej veci ako názov robota, škála modelu, povolenie kolízií robota so samým sebou a konfigurácia kontrolérov pre časti robota. Do konfiguračného súboru je možné definovať viac častí pre scénu, úlohy a odmeny pre učenie agentov, randomizácia domény atď., no pre naše účely je táto časť konfiguračného súboru postačujúca.

2.2 Súradnicová sústava a nastavenia simulátora

Simulátor iGibson používa pravotočivý karteziánsky súradnicový systém. Pravotočivý systém sa riadi pravidlom pravej ruky. Vysvetlíme, ako sú jednotlivé osi orientované. Ak ukazovák ukazuje v smere osi X a prostredník v smere osi Y, tak palec, ktorý je

kolmo na obidva prsty, bude ukazovať v smere osi Z. Hodnoty osi X sa zvyšujú od tela dopredu, osi Y doľava a osi Z hore.

2.2.1 Pozícia a orientácia

Opíšeme, čím sa myslí pozícia a orientácia v simulátore iGibson.

Pozícia: Pozícia objektu je definovaná ako trojrozmerný vektor (x, y, z), ktorý určuje jeho polohu v priestore.

Orientácia: Orientácia objektu je definovaná ako štvorrozmerný vektor (x, y, z, w), ktorý sa nazýva kvaternión a určuje jeho rotáciu v priestore.

Kvaternióny

Kvaternióny sú štvorrozmerné vektory (x, y, z, w), ktoré predstavujú rotáciu v 3D priestore:

- x, y, z sú zložky vektora osi rotácie.
- w je skalárna zložka reprezentujúca mieru rotácie okolo tejto osi.

Matematicky, kvaternión (x, y, z, w) reprezentuje rotáciu okolo vektora $\mathbf{v} = (x, y, z)$ s uhlom θ, kde:

$$w = \cos\left(\frac{\theta}{2}\right)$$

$$(x, y, z) = \mathbf{v}\sin\left(\frac{\theta}{2}\right)$$

Kvaternióny majú niekoľko výhod:

- **Vyhnutie sa gimbal locku**: Kvaternióny sa vyhýbajú gimbal locku, ktorý spôsobuje stratu jedného stupňa voľnosti pri určitých rotáciách.
- **Kompaktnosť a efektívnosť**: Kvaternióny sú výpočtovo efektívnejšie a zaberajú menej miesta ako matice rotácie.

Eulerove uhly

Eulerove uhly definujú rotáciu okolo troch ortogonálnych osí:

- **Roll** (ϕ): Rotácia okolo osi X.
- **Pitch** (θ): Rotácia okolo osi Y.
- **Yaw** (ψ): Rotácia okolo osi Z.

V simulátore iGibson sa Eulerove uhly používajú v poradí $(\text{roll}, \text{pitch}, \text{yaw})$. Aj keď sú intuitívnejšie, môžu trpieť gimbal lockom.

2.2.2 Jednotky

Simulátor iGibson používa SI jednotky, čo znamená, že vzdialenosti sú v metroch, sily v Newtonoch a rotácie v radiánoch.

2.2.3 Gravitácia

Gravitácia je zvyčajne nastavená tak, aby pôsobila proti smeru osi Z, teda smerom nadol.

2.3 VR hardvér

Simulátor iGibson zabezpečuje kompatibilitu s viacerými typmi VR systémov, ako napríklad Oculus alebo HTC Vive. Na to, aby sa dosiahla táto kompatibilita, je v simulátore umožnené nastavenie potrebnej konfigurácie pre akékoľvek VR zariadenie kompatibilné so systémom SteamVR (Valve Corporation, 2024). My sme pre vývoj rozhrania používali systém Meta Quest 2. Pre teleoperáciu sú používané dva základné hardvérové komponenty: HMD a VR kontroléry schopné haptickej odozvy.

Kapitola 3

Výsledky práce

3.1 Import robota NICO do simulátora iGibson

Ako sme už spomínali v sekcií 2.1 k importovaniu robota NICO boli potrebné tri súbory, a to: Súbor vo formáte URDF, `nico.py` a `nico.yaml`. Popíšeme, čo sa v každom súbore muselo upraviť alebo definovať.

3.1.1 Súbor URDF

Boli nám dodané tri URDF súbory:

- `nico_upper_rh6d.urdf`,
- `nico_upper_rh6d_dual.urdf`,
- `nico_upper_head_rh6d_dual.urdf`.

Každý z nich sa líšil množstvom pohyblivých častí modelu.

Model `nico_upper_rh6d.urdf` mal kĺby ľavého ramena, ľavého efektora, hlavy a krku zafixované.

Model `nico_upper_rh6d_dual.urdf` mal zafixované iba kĺby hlavy a krku.

Model `nico_upper_head_rh6d_dual.urdf` nemal už ani jednu zo spomínaných častí zafixovanú. Ďalej budeme hovoriť už iba o modeli `nico_upper_head_rh6d_dual-.urdf`. Model mal v kóde zakomentované časti, ktoré sa týkali prstov robota. Boli to konkrétne články a kĺby prostredníkov a malíčkov obidvoch rúk. Po odkomentovaní a doladení boli prsty funkčné. Bohužiaľ model nemal definovaný palec ani na jednej ruke, takže vykonanie úspešného a použiteľného či už realistického, alebo asistovaného úchopu by bolo pre neho takmer nemožné. Ďalšou komplikáciou boli meshe, ktoré boli používané. Simulátor nedokázal správne spracovať a načítať meshe, ktoré boli typu `.stl` a museli sme im zmeniť formát na `.obj`. Po tejto zmene sa nám podarilo načítať robota NICO v prostredí iGibson, no prechod na nový formát spôsobil problémy s materiálmi meshov, v dôsledku čoho robot ostal čierny.

Pre ďalšiu prácu sme museli vedieť, aké kĺby má robot definované, a v akom sú poradí. Použitý finálny model robota NICO mal celkovo definovaných 24 kĺbov, z toho je 20 typu revolute a 4 typu fixed. Kĺby v URDF súbore, ktoré sme si rozdelili do skupín hlava a krk, pravé rameno, pravý efektor, ľavé rameno, ľavý efektor sú uvedené v rovnakom poradí v tabuľkách 3.2, 3.3, 3.4, 3.5 a 3.6.

3.1.2 Súbor nico.py

V súbore `nico.py` sme museli vytvoriť triedu `Nico` pre robota NICO. Táto trieda dedí metódy a vlastnosti z tried `ManipulationRobot` a `ActiveCameraRobot`. Dedenie z triedy `ActiveCameraRobot` je nevyhnutné pre ovládanie hlavy a krku robota, a zároveň poskytuje základ pre budúce rozšírenia tejto práce.

V tejto triede museli byť upravené alebo definované nasledovné časti:

- Mapovanie častí robota na indexy v riadiacom vektore.
- Kontroléry a ich poradie.
- Základné polohy kĺbov.
- Nastavenie bodov pre asistovaný úchop.
- Ďalšie potrebné vlastnosti pre funkčnosť robota.

Mapovanie

Pre jednotlivé časti robota NICO bolo podľa jeho poradia kĺbov v URDF súbore definované nasledovné mapovanie: Pre kameru sme definovali indexy $[0, 1]$, ďalej pre pravé rameno $[2, 3, 4, 5, 6, 7]$, pre pravý efektor $[8, 9, 10]$, pre ľavé rameno $[11, 12, 13, 14, 15, 16]$ a pre ľavý efektor $[17, 18, 19]$.

Kontroléry a ich poradie

Pre jednotlivé častí robota boli použité nasledovné kontroléry: **JointController** pre ovládanie kĺbov hlavy a krku, **InverseKinematicsController** pre ovládanie kĺbov ramien a znova **JointController** pre ovládanie prstov efektorov.

Každý kontrolér má pre organizáciu a správne fungovanie priradený názov časti, ktorý ovláda. Priradenia sú nasledovné:

- $camera \rightarrow JointController$,
- $arm\_right \rightarrow InverseKinematicsController$,
- $arm\_left \rightarrow InverseKinematicsController$,
- $gripper\_right \rightarrow JointController$,

- $gripper\_left \rightarrow JointController$.

Poradie kontrolérov bolo definované nasledovne: *camera, arm_right, gripper_right, arm_left, gripper_left*. Každý z kontrolérov je špecificky definovaný a tieto definície teraz opíšeme.

Kontrolér časti *camera* používaný pre ovládanie kĺbov hlavy a krku nepotreboval žiadne dodatočné nastavenia. Kontrolér nie je nastavený na prijímanie delta príkazov a teda akýkoľvek príkaz jemu podaný je interpretovaný ako absolútne nastavenie uhlov kĺbov krku a hlavy robota.

Kontrolér častí *arm_right* a *arm_left* používaný pre ovládanie kĺbov ramien taktiež nepotreboval žiadne dodatočné nastavenia. Jeho základné nastavenie používa režim *pose_delta_ori*, ktorého fungovanie sme už vysvetlili v sekcií 2.1.2.

Kontrolér častí *gripper_right* a *gripper_left* používaný pre ovládanie kĺbov prstov koncových efektorov boli definované nasledovne:

- Režim ovládania motorov je poziční - chceme nastavovať pozíciu, resp. uhol kĺbov prstov.
- Je používaný paralelný režim - tento režim určuje spôsob prijímania vstupov. Kontrolér prijíma jeden vstup a primerane ho škáluje pre každý kĺb.
- Je používaný invertovaný režim pre pravý efektor - tento režim určuje spôsob mapovania vstupných príkazov na výstupné.
- Je používaný delta režim - na ovládanie kĺbov sa používajú delta zmeny pozície.
- Rozsah vstupných príkazov je $[0, 1]$ - VR kontroléry majú výstupy tiež v rozmedzí $[0, 1]$, a teda aj kontroléry efektorov sme chceli ovládať v rovnakom rozmedzí.
- Rozsah výstupných príkazov je limitovaný na limity kĺbov prstov.

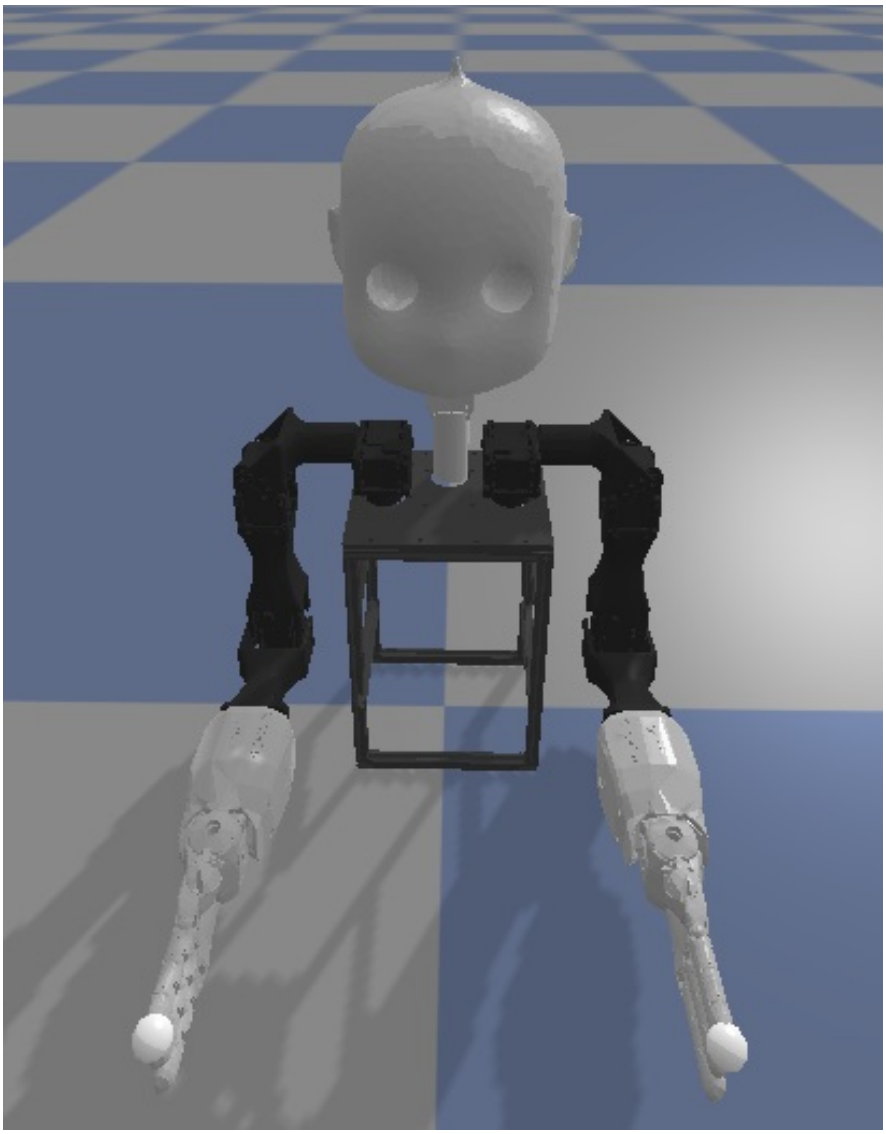
Základná poloha kĺbov

Základné polohy kĺbov sme nastavili tak, aby robot bol v neutrálnej polohe, ktorú je vidieť na obrázku 3.1 a 3.2. Do tejto polohy je robot nastavený aj pri jeho resetovaní.

Nastavenie bodov pre asistovaný úchop

Kvôli anatomickým obmedzeniam modelu robota NICO nie je v tomto momente možné efektívne používať iný režim uchopovania ako režim *sticky*. Napriek tomu sme sa rozhodli implementovať definíciu bodov pre asistovaný úchop s ohľadom na budúce rozšírenia našej práce.

Pre každý efektor boli definované dva začiatočné body. Ich pozície sú relatívne voči pozíciám článkov *right_palm* a *left_palm*. Prvý začiatočný bod je definovaný na



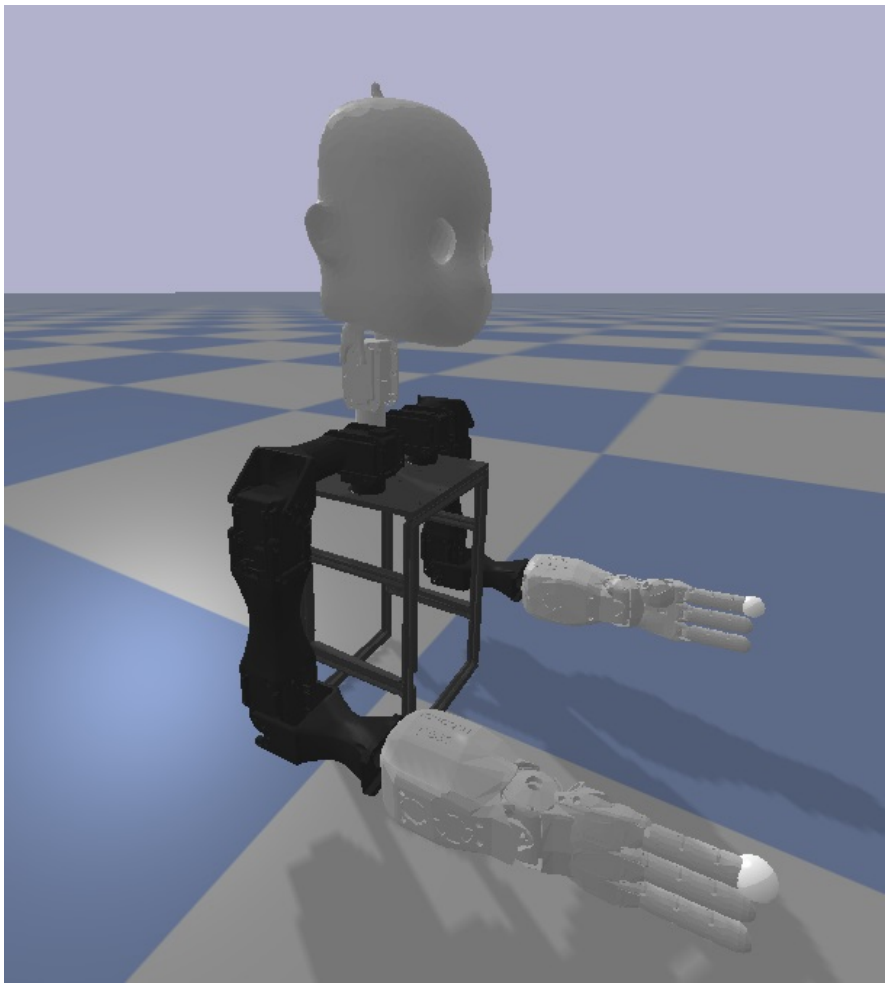
Obr. 3.1: Robot NICO v základnej polohe spredu.

pozícií **PALM_BASE_POS**, teda $(0.025, 0, 0)$. Druhý bod je definovaný na pozícií **PALM_CENTER_POS**, teda $(0.035, 0, 0)$.

Ďalej pre každý efektor boli definované koncové body na koncoch prstov. Ich pozície sú relatívne voči pozíciám článkov prstov. Každý z bodov je definovaný na pozícií **FINGER_TIP_POS**, teda $(0.05, 0, 0)$.

Vypnutie kolíznych párov

Bolo potrebné vypnúť kolízie medzi dvojicami článkov (*neck*, *right_shoulder*) a (*neck*, *left_shoulder*). Ak kolízie medzi nimi neboli vypnuté, dochádzalo k situáciám, kedy bol detekovaný kontakt medzi týmito článkami, ktorý systém následne prezentoval ako haptickú odozvu pomocou VR kontrolérov.



Obr. 3.2: Robot NICO v základnej polohe zboku.

Ďalšie časti

Ďalej sa definovali vlastnosti ako názvy článkov efektorov, názvy kĺbov a článkov prstov efektorov a adresa k URDF súboru, ktorý má byť pri importovaní robota načítaný a použitý.

3.1.3 Súbor nico.yaml

V súbore sa definovala základná konfigurácia pre robota, ktorá je používaná na nastavenie robota pri jeho importovaní do scény.

3.2 Implementácia teleoperácie

Po importovaní a následnom definovaní všetkých častí potrebných pre správne fungovanie ovládania robota v simulátore, sme sa pustili do návrhu a implementácie teleoperačného algoritmu a vytvorenia teleoperačného rozhrania, ktoré sú súčasťou cieľov

našej práce.

Teleoperačný algoritmus pozostáva z týchto častí:

- Zber dát z HMD.
- Vypočítanie pozícií kĺbov hlavy.
- Zber dát z kontrolérov o stlačení reset tlačidla.
- Vykonanie reset-u pri stlačení reset tlačidla.
- Zber dát z kontrolérov o ich pozícií a orientácií.
- Unifikácia súradnicovej sústavy, ak je potrebná.
- Vypočítanie delty translácie a orientácie.
- Zber dát z kontrolérov o stave stlačenia úchopu.
- Výpočet delty úchopu.

Priblížime si bližšie algoritmus, ktorý je popísaný v pseudokóde 1. Algoritmus sa spúšťa pri výpočte každého nového kroku simulátora, aby vypočítal aktuálnu akciu pre robota. Na začiatku sa zozbierajú dáta z VR hardvéru. Pre každý individuálny kus hardvéru sa vždy vracia aj informácia o validite jeho dát, t.j. či daný hardvér bol v okamihu zberu dát aktívny a vracal aktuálne dáta.

Najskôr zbierame dáta z HMD. Pre nášho robota sme mohli tieto dáta spracovať priamo, keďže potrebujeme vedieť iba informáciu o orientácii HMD, konkrétne o pozíciách osí Y a Z, pretože náš robot má iba také osi kĺbov krku a hlavy. Žiadny netriviálny výpočet nebol potrebný a informácia o pozícii sa dá priamo posunúť kontroléru kĺbov hlavy a krku.

Potom kontrolujeme, či nebol prijatý signál o resete robota. Ak áno, nastavíme robota do jeho základnej polohy.

Ďalej nasleduje cyklus, ktorý zabezpečuje výpočet translácie a rotácie pre každú ruku robota. Na začiatku sa rovnako ako pre HMD zbierajú dáta z príslušných VR kontrolérov. Ak nie sú validné, pokračuje sa ďalej k ďalšej ruke robota. Ak áno, pokračuje sa k testu, ktorý kontroluje, či sme kontrolér už stlačili predtým a teda či je potrebné resetovať pôvodné začiatočné pozície.

Reset je potrebné vykonávať, aby sme zaviedli unifikovanú súradnicovú sústavu, pomocou ktorej budeme počítať výslednú transláciu a rotáciu. V pseudokóde 2 je lepšie vidieť, ako tieto počiatočné pozície figurujú pri počítaní pozičnej akcie. Transláciu počítame ako jednoduché odčítanie vektorov. Keď pôvodné začiatočné pozície resetujeme, hodnoty `robotArmOriginPos` a `controllerOriginPos` sú rovné hodnotám `robotArmActualPos` a `controllerActualPos` v aktuálnom snímku, resp. v aktuálnom výpočtovom cykle simulátora. Ak je tlačidlo podržané, sú používané naposledy zresetované počiatočné pozície.

Algorithm 1 Generovanie akcie robota

```
action ← zerosArray(robotActionSpace)
v ← generatedVrData()
valid, hmdPos, hmdOrn ← v.get(hmd)
if valid then
    y ← getY(hmdOrn)
    z ← getZ(hmdOrn)
    action[robotCameraJointIndexes] ← [z, y]
end if
if v.get(reset) then
    resetRobot()
end if
for arm ← robotArms do
    valid, vrControllerPos, vrControllerOrn ← v.get(armController)
    if !valid then
        continue
    end if
    if teleoperationControlToggled then
        if resetOrigin then
            resetRobotAndVrControllerOrigin()
        end if
        posAction ← calculatePosAction()
        ornAction ← calculateOrnAction()
        action[robotArmControlIndexes] ← [posAction, ornAction]
    else
        resetOrigin ← True
    end if
    vrControllerTrigger ← v.get(armTrigger)
    graspDelta ← calculateGraspAction()
    action[robotGripControlIndexes] ← [graspDelta]
end for
    return action
```

Algorithm 2 Kalkulácia pozičnej akcie

```
robotPosOffset ← robotArmActualPos − robotArmOriginPos
controllerPosOffset ← controllerActualPos − controllerOriginPos
posAction ← controllerPosOffset − robotPosOffset
    return posAction
```

Ďalej počítame rotačnú, resp. Eulerovu akciu. Výpočet samotný robíme pomocou kvaterniónov, no finálnu rotačnú akciu musíme podať v podobe Eulerových uhlov, ktoré sú vyjadrené v radiánoch. V pseudokóde 3 vidíme, že používame funkciu `quatDiff`, ktorá vypočíta a vracia kvaterniónový rozdiel orientácií medzi počiatočnou pozíciou a aktuálnou pozíciou ruky/kontroléra.

Algorithm 3 Kalkulácia rotačnej akcie

$robotQuatOffset \leftarrow quatDiff(robotArmActualOrn, robotArmOriginOrn)$
$controllerQuatOffset \leftarrow quatDiff(controllerActualOrn, controllerOriginOrn)$
$quatAction \leftarrow controllerQuatOffset - robotQuatOffset$
$eulerAction \leftarrow quatToEuler(quatAction)$
return $eulerAction$

Poslednou časťou algoritmu 1 je kalkulácia akcie pre úchop. V pseudokóde 4 vidíme, že výpočet je rozdielny pre ľavú a pravú ruku robota. Tento rozdiel je spôsobený rôznymi limitmi kĺbov prstov pre ľavú a pravú ruku. Simulátor normalizuje pozície kĺbov prstov ľavej ruky inak ako pravej.

Pre pravú ruku platí, že keď sú prsty v otvorenej polohe, kĺby nadobúdajú pri normalizovaných hodnotách hodnotu 1. Pre ľavú ruku platí opak, čiže kĺby nadobúdajú pri otvorenej polohe hodnotu -1. Hodnotu `currentTriggerValue`, ktorú chceme používať ako indikátor aktuálneho stavu otvorenia úchopu, chceme mať normalizovanú medzi hodnotami 0 a 1, pretože aj hodnoty zbierané z VR kontrolérov sú na intervale [0, 1].

Ďalší rozdiel je použitie funkcií **min** a **max**. Pre každú ruku chceme mieru úchopu počítať podľa najviac zavretého prsta, čo pre pravú ruku znamená zobrať minimum z hodnôt kĺbov prstov a pre ľavú ruku zobrať maximum.

Algorithm 4 Kalkulácia akcie úchopu

$vrControllerTriggerValue \leftarrow v.get(trigger)$
if $rightArm$ **then**
  $currentTriggerValue \leftarrow 1 - min(normalizedJointEEFpositions + 1)/2$
else
  $currentTriggerValue \leftarrow max(normalizedJointEEFpositions + 1)/2$
end if
$deltaTrigger \leftarrow vrControllerTriggerValue - currentTriggerValue$
return $deltaTrigger$

Týmto je zabezpečený správny pohyb prstov pri úchope a upustení úchopu, no stále tu bol problém s logikou toho, či má byť objekt rukou uchopený, alebo nie. Na to, aby sme túto logiku správne upravili museli sme preťažiť metódu *_handle_assisted_grasping*, ktorá bola za ňu zodpovedná. V našej metóde sme rozlišovali medzi pravou a ľavou

rukou. V pôvodnej metóde podmienka pre zapnutie úchopu bola definovaná nasledovne:

$$np.any(desired\_positions < activation\_thresholds) \tag{3.1}$$

Hodnota `desired_positions` je pole, v ktorom sú hodnoty požadovanej pozície, ktorú by chceli kĺby prstov ruky dosiahnuť. Hodnota `activation_thresholds` je pole, v ktorom sú určené aktivačné hranice pre každý kĺb prstov ruky. Je počítaná nasledovne:

$$\begin{gathered} activation\_thresholds = \\ (1 - ASSIST\_ACTIVATION\_THRESHOLD) * self.joint\_lower\_limits \\ + ASSIST\_ACTIVATION\_THRESHOLD * self.joint\_upper\_limits \end{gathered} \tag{3.2}$$

Parameter `ASSIST_ACTIVATION_THRESHOLD` sme nastavili na hodnotu **0.5**. Pre ľavú ruku sme museli hodnoty `desired_positions` a `activation_thresholds` obrátiť, resp. vynásobiť číslom -1. Kĺby ľavej ruky majú uhlové rozpätie v intervale od 0 až po 2.57 radiánov. Problém nastáva vtedy, keď ideme testovať podmienku 3.1. Keďže `activation_thresholds` je pre každý kĺb na hodnote $\frac{2.57}{2}$ a hodnoty v poli `desired_positions` v stave, kedy nie je prikázané uchopenie, začínajú na nule, podmienka je už v pasívnom stave vyhodnotená ako pravdivá, a to spôsobí nežiadané uchopovanie objektov.

Ďalej bolo potrebné ošetriť aj podmienky pre upustenie úchopu. Hodnoty, ktoré sú používané na kontrolu, sú:

$$\begin{aligned} thresholds &= np.maximum(clipped\_current\_positions, activation\_thresholds) \\ releasing\_grasp &= np.all(desired\_positions > thresholds) \end{aligned} \tag{3.3}$$

A podmienka samotná je:

$$\text{constraint\_violated or releasing\_grasp} \tag{3.4}$$

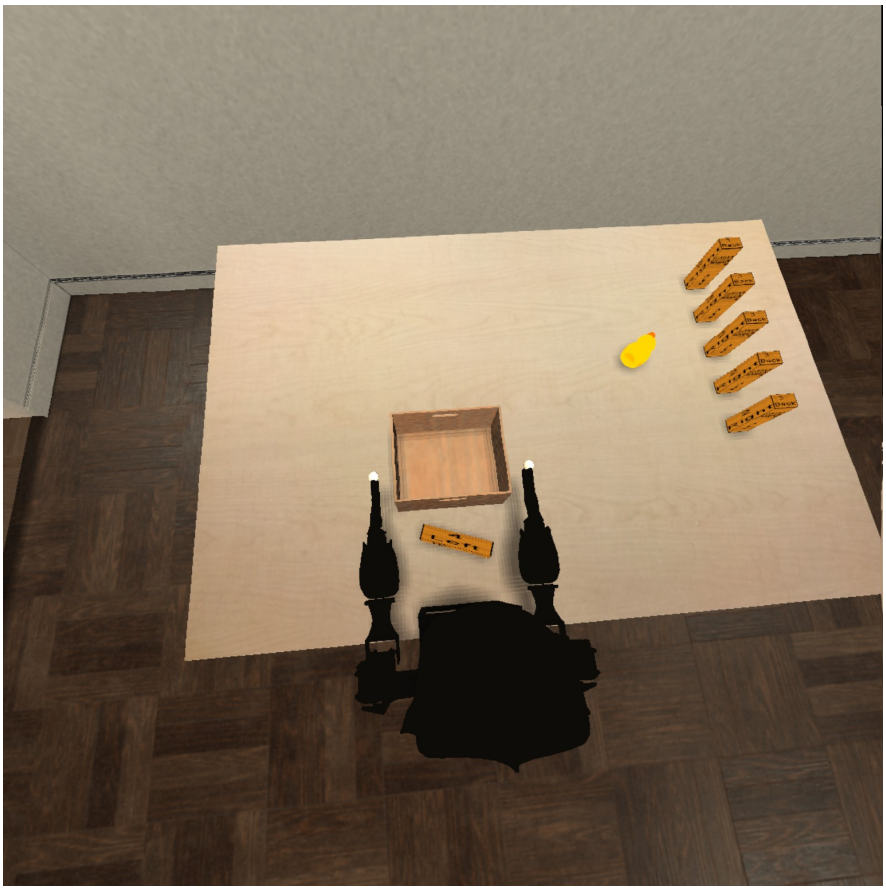
Opäť vidíme problém so znamienkom porovnania pri vyhodnocovaní hodnoty `releasing_grasp`. Na to, aby sme dosiahli žiadaného správania upustenia úchopu stačilo hodnotu `clipped_current_positions` taktiež obrátiť tak, ako sme to urobili s predošlými hodnotami. Po týchto úpravách boli úchopy obidvoch rúk v *sticky* režime už plne funkčné.

Implementácia tohto algoritmu nastala v metóde *gen_vr_robot_action* triedy **SimulatorNicoVR**, ktorá dedí vlastnosti a metódy od triedy **SimulatorVR**. Tá bola využívaná na VR simuláciu modelu BehaviourRobot a je v simulátore už celá implementovaná. Pohyb avatara reprezentujúceho operátora v scéne je taktiež implementovaný v pôvodnej triede a je riadený ľavým joystickom ľavého kontroléra v smeroch osí X a Y. Pravý joystick riadi pohyb avatara na osi Z. Ďalej bola implementovaná aj citlivosť robota na kontakt s prostredím, čo spôsobovalo haptickú odozvu vo VR kontroléroch.

3.3 Otestovanie teleoperačného systému

Ďalším krokom po naprogramovaní a vyladení teleoperačného rozhrania bolo jeho otestovanie. Testovanie prebiehalo so vzorkou piatich ľudí, každá osoba má rôzne skúsenosti s VR hardvérom, čo sa predpokladane ukázalo aj vo výsledkoch testovania.

Na testovanie sme mali predpripravené 3 testovacie scény. Všetky scény boli odlišné len v pozícií, počte a druhoch objektov, ktoré sa nachádzali na stole pred robotom. Prvá scéna slúžila na oboznámenie sa s teleoperačným rozhraním. Scénu vidíme na obrázkoch 3.3 a 3.4. Ovládanie rozhrania je operátorovi vysvetlené. Pred robotom sa nachádza jedna kocka jengy a jedna krabica. Operátor má za úlohu vložiť túto kocku pomocou teleoperovania robota do krabice. Ak sa mu toto podarí, tak započne ďalšia fáza testovania.



Obr. 3.3: Pohľad zvrchu na testovaciu scénu slúžiacu na oboznámenie operátora.

Operátor bol presunutý do scény, kde sa pred robotom nachádza viacero ľahkých objektov, ktoré má operátor rovnako ako predtým za úlohu pomocou teleoperácie robota dostať do krabice. Scénu vidíme na obrázkoch 3.5 a 3.6. Objekty sú natoľko ľahké, aby bolo pomocou teleoperovania iba jednej ruky robota možné ich vložiť do krabice. Scenár sa opakuje pre každého operátora päťkrát. Úspešnosť každého pokusu sa zaznamenáva. Na to, aby operátor splnil úlohu úspešne, tak musel splniť nasledovné

Obr. 3.4: Pohľad zboku na testovaciu scénu slúžiacu na oboznámenie operátora.

podmienky:

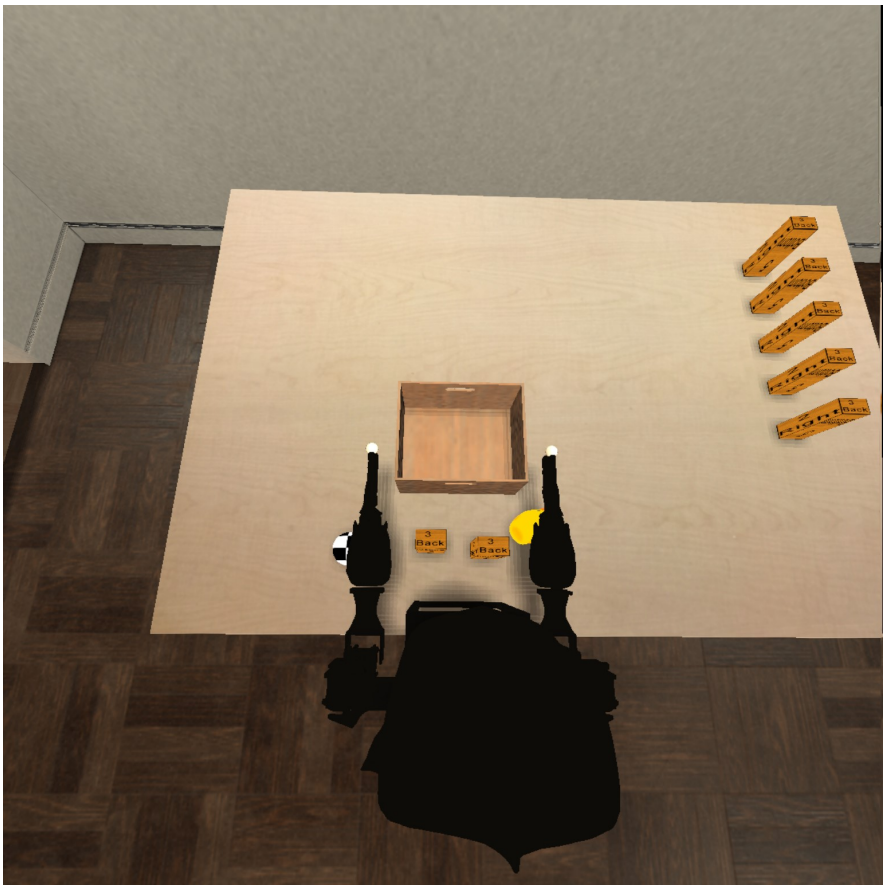
- Operátor dokončil úlohu bez toho, aby nestratil ani jeden objekt kvôli jeho chybe.
- Operátor dokončil úlohu do piatich minút.

Po dokončení testovania v scéne s ľahkými objektmi, operátor bol presunutý do scény, kde sa pred robotom nachádza jeden ťažší objekt. Scénu vidíme na obrázkoch 3.7 a 3.8. Tento objekt je natoľko ťažký, že na presun do krabice je potrebné použiť obidve ruky robota. Podmienky úspešnosti pokusu sú rovnaké ako pri scenári s ľahkými objektmi.

Nakoniec operátor ohodnotil rozhranie v kategóriach intuitívnosti a ovládateľnosti na škále od 1 do 5, kde ohodnotenie 1 bolo najhoršie a ohodnotenie 5 najlepšie. V kategórií intuitívnosť sa hodnotila subjektívna náročnosť zvykania si na neznáme používateľské rozhranie. V kategórií ovládateľnosť sa hodnotila subjektívna schopnosť operátora manipulovať s objektmi pomocou teleoperovania robota v testovacom scenári.

Ostatné možné kategórie ako napríklad pocit imerzie, alebo iné spôsoby testovania ako napríklad meranie časov trvania konkrétnych manipulačných úloh neboli pre

testovanie nášho rozhrania vhodné, keďže imerzia a ďalšie stránky nás nezaujímali a merané časy by nemohli byť s ničím relevantným porovnané.



Obr. 3.5: Pohľad zvrchu na testovaciu scénu s ľahkými objektmi.

3.3.1 Výsledky testovania

Ďalším krokom je zhodnotenie testovania a aktuálneho stavu rozhrania. V tabuľke 3.1 vidíme úspešnosti vykonávania úloh jednotlivých operátorov a ich ohodnotenia rozhrania v kategóriách intuitívnosti a ovládateľnosti. Operátori 1, 2 a 5 testovaním skúšali VR rozhranie po druhýkrát a operátori 3 a 4 mali s rozhraním viac skúseností. To sa odrazilo na výsledkoch testov. Priemerné ohodnotenie v kategórií intuitívnosti je **4.4/5** a v kategórií ovládateľnosti **3.4/5**.

3.4 Zhodnotenie teleoperačného systému

Po otestovaní sme dospeli k nasledujúcim záverom:

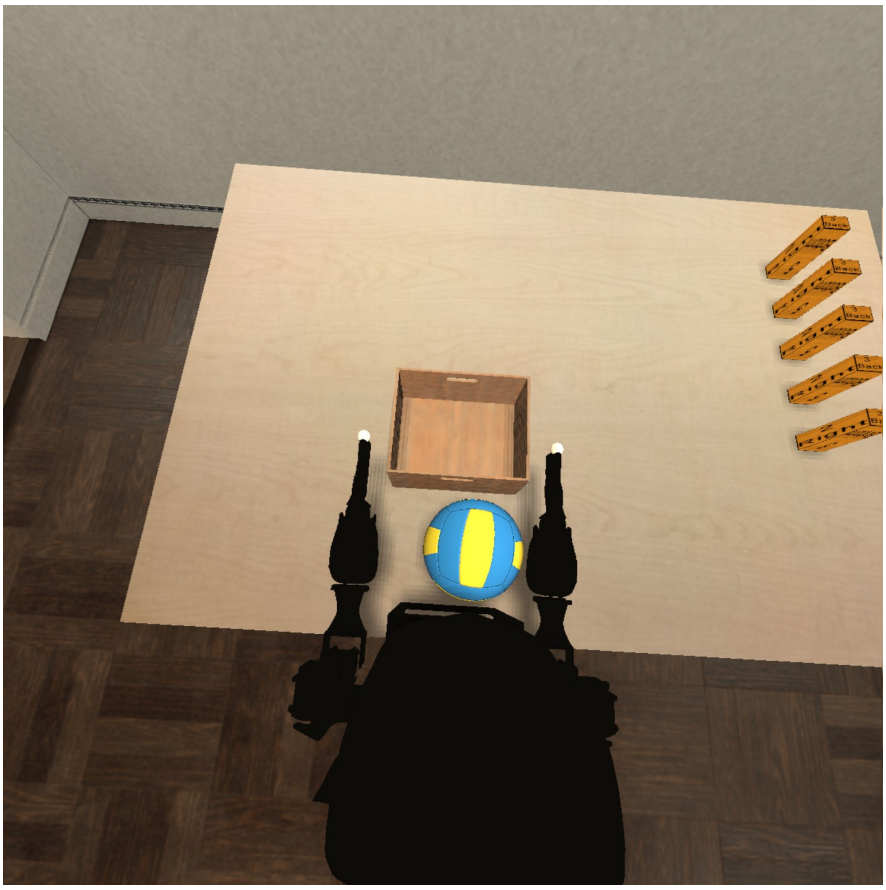
- Na základe relatívne vysokého ohodnotenia v kategórii intuitívnosti je rozhranie natoľko jednoduché, že aj menej skúsení operátori dokážu po krátkom oboznámení teleoperovať robota s dobrým pomerom úspechu.

Obr. 3.6: Pohľad zboku na testovaciu scénu s ľahkými objektmi.

- Teleoperácia ľahkých objektov nespôsobovala operátorom vážnejšie problémy. Objekty boli častejšie vkladané do krabice pravou rukou. Pri teleoperovaní sa občas vyskytol nežiadaný jav zo strany systému, keď pri uchopení objektu bola celá ruka ťahaná do výšky a dozadu. Po upustení objektu a jeho opätovnom uchopení sa tento problém už neobjavil. Toto správanie bolo nakoniec vyriešené, keď sa zistilo, že v metóde *gen_vr_data* bola zakomentovaná časť kódu, ktorá bola zodpovedná za uloženie informácií o pozícii a rotácii VR kontrolérov.

- Teleoperácia ťažšieho objektu vyžadovala použitie oboch rúk, čo bolo pre každého operátora rôzne náročné. Niektorí operátori si našli overený spôsob zdvihnutia objektu a ten použili na úspešné vykonanie všetkých úloh. Usúdili sme, že na tvare objektu, aspoň pri *sticky* režime úchopu, nezáležalo. Operátori objekt uchopovali z bokov a nie zospodu.

Príčina niektorých neúspechov bola ľudská chybovosť pri teleoperovaní robota, čo viedlo k pádom a neopatrným kolíziám s objektmi. Niektorí operátori zabudli pri vykonávaní úloh na správanie rozhrania pri teleoperácii, resp. zabudli na primárny koncept delta príkazov, ktorými je robot riadený. To spôsobovalo nežiadané správanie robota, ako napríklad príliš pomalú rýchlosť pohybu ruky robota do vybranej strany.



Obr. 3.7: Pohľad zvrchu na testovaciu scénu s ťažkým objektom.

Možnosť resetovať polohu robota bola prínosná hlavne z hľadiska skrátenia potrebného času na vykonanie úlohy.

3.4.1 Návrhy na vylepšenie

Nami vytvorené teleoperačné rozhranie je stále len v počiatku vývoja, ale je to dobrý základ, od ktorého môže nasledovať ďalší vývin v rôznych jeho smeroch. Napadá nás viacero návrhov na jeho vylepšenie. Jedným z nich je vytvorenie intuitívneho grafického používateľského rozhrania (GUI), ktoré poskytne operátorovi všetky potrebné informácie na to, aby mohol robota teleoperovať.

Ďalšími návrhmi sú možnosť vidieť scénu z pohľadu očí robota, čo by zlepšilo presnosť a efektívnosť teleoperácie, a nahrávanie a ukladanie dát z teleoperácie. Nahraté dáta môžu byť použité na učenie modelov umelej inteligencie robota NICO. Pohľad z očí robota bude prispievať hlavne k tomuto aspektu. Operátor bude vidieť to isté, čo vidí robot, takže dáta z teleoperácie budú viac realistické a vierohodné, čo bude viesť k lepšiemu učeniu modelov. Ďalším návrhom je implementácia realistického resetu polohy robota, ktorý zabezpečí, že robot sa vždy vráti do východiskovej pozície spôsobom, ktorý je prirodzený a zohľadňuje fyzické obmedzenia robota.

Obr. 3.8: Pohľad zboku na testovaciu scénu s ťažkým objektom.

Je tu aj potenciál na prenášanie akcií simulovaného robota na akcie robota v reálnom svete. Aby to bolo možné, je potrebné mať model robota NICO, ktorý je dostatočne podobný originálu, aby nevznikali nekonzistencie v polohách a pohyboch robota. Po dodaní modelu s palcom a prstami vyrobenými z viac ako jedného článku by sa mali implementovať rôzne režimy úchopu. Aktuálne využívaný *sticky* režim by mohol byť doplnený o asistovaný režim úchopu, na ktorý už máme pripravené základy implementácie. Toto rozšírenie by umožnilo robotovi realistickejšie manipulovať s predmetmi a vykonávať zložitejšie úlohy.

3.5 Zhodnotenie práce v simulátore iGibson

V tejto práci sme sa zamerali na importovanie a uvedenie do prevádzky robota NICO, ako aj na návrh, implementáciu a testovanie teleoperačného rozhrania. Na dosiahnutie týchto cieľov bolo nevyhnutné pochopiť vnútornú logiku simulátora pre riadenie robotov. Dokumentácia simulátora pokrýva všetky jeho časti v dostatočnej miere, avšak na hlbšie porozumenie vnútorných mechanizmov sme museli nahliadnuť priamo do zdrojového kódu.

Operátor	**Ľahké objekty**		**Ťažký objekt**		**Intuitívnosť**	**Ovládateľnosť**
	Úspešné	Neúspešné	Úspešné	Neúspešné		
1	5	0	2	3	5	3
2	4	1	5	0	4	4
3	5	0	5	0	5	4
4	5	0	5	0	4	3
5	5	0	5	0	4	3

Tabuľka 3.1: Úspešnosť testovania a hodnotenia VR rozhrania

Hlavnou prekážkou pri pokračovaní v práci bola neprítomnosť ucelenej dokumentácie pre niektoré kľúčové časti, ako sú ukladanie dát robota (kĺby, články), logika riadenia robota pomocou akcií a logika kontrolérov. Toto spôsobilo neucelené chápanie jednotlivých komponentov simulátora, čo následne predĺžilo čas potrebný na ladenie a debugovanie našej implementácie. Napríklad pri implementácii úchopu ľavého efektora bolo nutné preťažiť metódu triedy **ManipulationRobot**, ktorá riadila, kedy má byť objekt uchopený rukou. Kým sme zistili, že takýto zásah je potrebný, skúšali sme rôzne nastavenia kontroléra ľavého efektora, no po viacerých pokusoch sme zistili, že týmto spôsobom nedosiahneme požadované správanie. Práca v simulátore teda vyžaduje časovú investíciu na pochopenie vnútorného riadenia celého systému.

Keď však pochopíme vnútorné fungovanie simulátora, práca v ňom sa stáva pomerne jednoduchou. Početné možnosti, ktoré simulátor ponúka, sú vítanou výhodou. Má veľký potenciál pre vykonávanie realistických simulácií, ktoré môžu byť využité na rôznorodé účely, ako napríklad testovanie kinematiky robota, manipulačné úlohy, úlohy vyžadujúce mobilitu, alebo učenie inteligentných agentov rôznym pravidlám. Učenie je robustnejšie a všeobecnejšie vďaka prítomnosti randomizácie textúr.

Simulátor má svoje datasety zamerané na prostredia obývané ľuďmi a ich objekty. Ak by sme však mali k dispozícii vlastné originálne scény a objekty vo formátoch kompatibilných so simulátorom, ktoré by sa kvalitou približovali k originálnym datasetom, nebol by problém ich vložiť do simulátora. Tým by sme dokázali vykonať kvalitné simulácie aj v týchto prostrediach a s týmito objektami.

Celkovo hodnotíme toto prostredie ako kvalitné a rýchle, s veľkou rozmanitosťou možností pre vývoj robotických systémov.

Názov	Os	Rodič	Dieťa	Limity	
				Spodný	Vrchný
head_z_rjoint	Z	torso	neck	-1.5708	1.5708
head_y_rjoint	Y	neck	head	-0.8726	0.4363

Tabuľka 3.2: Kĺby hlavy a krku robota NICO. Limity sú udávané v radiánoch.

Názov	Os	Rodič	Dieťa	Limity	
				Spodný	Vrchný
r_shoulder_z_rjoint	Z	torso	right_shoulder	-0.4363	1.3963
r_shoulder_y_rjoint	Y	right_shoulder	right_collarbone	-0.5236	3.1416
r_upperarm_x_rjoint	X	right_collarbone	right_upper_arm	0	1.2217
r_elbow_y_rjoint	Y	right_upper_arm	right_lower_arm	0.8726	3.1416
r_wrist_z_rjoint	Z	right_lower_arm	right_wrist	-1.571	1.571
r_wrist_x_rjoint	X	right_wrist	right_palm	-0.78	0.78

Tabuľka 3.3: Kĺby pravého ramena robota NICO. Limity sú udávané v radiánoch.

Názov	Os	Rodič	Dieťa	Limity	
				Spodný	Vrchný
gripper_rjoint	Y	right_palm	gripper	-2.57	0
middlefinger_rjoint	Y	right_palm	middlefinger	-2.57	0
litllefinger_rjoint	Y	right_palm	littlefinger	-2.57	0

Tabuľka 3.4: Kĺby pravého efektora robota NICO. Limity sú udávané v radiánoch.

Názov	Os	Rodič	Dieťa	Limity	
				Spodný	Vrchný
l_shoulder_z_rjoint	Z	torso	left_shoulder	-1.3963	0.4363
l_shoulder_y_rjoint	Y	left_shoulder	left_collarbone	-0.5236	3.1416
l_upperarm_x_rjoint	X	left_collarbone	left_upper_arm	0	1.2217
l_elbow_y_rjoint	Y	left_upper_arm	left_lower_arm	0.8726	3.1416
l_wrist_z_rjoint	Z	left_lower_arm	left_wrist	-1.571	1.571
l_wrist_x_rjoint	X	left_wrist	left_palm	-0.78	0.78

Tabuľka 3.5: Kĺby ľavého ramena robota NICO. Limity sú udávané v radiánoch.

Názov	**Os**	**Rodič**	**Dieťa**	**Limity**	
				Spodný	**Vrchný**
grippel_rjoint	Y	left_palm	gripper	0	2.57
middlefingel_rjoint	Y	left_palm	middlefinger	0	2.57
litllefingel_rjoint	Y	left_palm	littlefinger	0	2.57

Tabuľka 3.6: KĺBy ľavého efektora robota NICO. Limity sú udávané v radiánoch.

Záver

V tejto bakalárskej práci sme sa zamerali na import robota NICO do simulátora iGibson, čo zahŕňalo úpravu a definovanie rôznych súborov a implementáciu teleoperačného systému. Úspešne sme importovali robota a vytvorili funkčný teleoperačný systém, ktorý umožňuje ovládanie robota na diaľku prostredníctvom VR rozhrania. Testovanie systému preukázalo jeho schopnosti a obmedzenia, pričom sme identifikovali niekoľko oblastí na jeho vylepšenie. Náš teleoperačný systém, hoci je ešte v počiatočnej fáze vývoja, predstavuje pevný základ pre budúce inovácie. Simulátor iGibson sa ukázal ako užitočný nástroj na testovanie a vývoj teleoperačných a iných robotických systémov. Jeho široká paleta možností a úsilie o dodržanie realizmu simulácie sú veľmi cenené. Tieto vlastnosti umožňujú bezpečné a efektívne testovanie a učenie robotov v rôznych prostrediach bez rizika poškodenia reálneho hardvéru. Na základe výsledkov našej práce máme niekoľko návrhov na jej nadstavbu:

1. Vytvorenie intuitívneho grafického používateľského rozhrania, ktoré poskytne operátorovi všetky potrebné informácie na teleoperáciu robota.

2. Implementácia možnosti vidieť scénu z pohľadu očí robota, čo by zlepšilo presnosť a efektívnosť teleoperácie, a nahrávanie a ukladanie dát z teleoperácie. Tieto dáta môžu byť použité na učenie modelov umelej inteligencie robota NICO.

3. Realistický reset polohy robota, ktorý zabezpečí, že robot sa vždy vráti do východiskovej pozície spôsobom, ktorý zohľadňuje fyzické obmedzenia robota.

4. Prenášanie akcií simulovaného robota na akcie robota v reálnom svete, čo vyžaduje model robota NICO podobný originálu, aby nevznikali nekonzistencie v polohách a pohyboch robota.

5. Po dodaní modelu s palcom a prstami vyrobenými z viac ako jedného článku, implementácia rôznych režimov úchopu. Aktuálny "sticky"režim by mohol byť doplnený o asistovaný režim úchopu, čo by umožnilo robotovi realistickejšie manipulovať s predmetmi a vykonávať zložitejšie úlohy.

Tieto návrhy predstavujú možné smery pre ďalší vývoj a vylepšenie teleoperačného systému, čím by sa zvýšila jeho funkcionalita a efektívnosť.

Literatúra

Avila, L. and Bailey, M. (2014). Virtual reality for the masses. *IEEE Computer Graphics and Applications*, 34(05):103–104.

Basu, A. (2019). A brief chronology of Virtual Reality. *arXiv e-prints*, page arXiv:1911.09605.

Boas, Y. (2013). Overview of virtual reality technologies. In *Interactive Multimedia Conference*, volume 2013, page 4.

Choi, H., Crump, C., Duriez, C., Elmquist, A., Hager, G., Han, D., Hearl, F., Hodgins, J., Jain, A., Leve, F., et al. (2021). On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1):e1907856118.

Collins, J., Chand, S., Vanderkop, A., and Howard, D. (2021). A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431.

Ferrell, W. R. and Sheridan, T. B. (1967). Supervisory control of remote manipulation. *IEEE spectrum*, 4(10):81–88.

Furuta, K., Kosuge, K., Shiote, Y., and Hatano, H. (1987). Master-slave manipulator based on virtual internal model following control concept. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 567–572.

Goertz, R. C. and Thompson, W. M. (1954). Electronically controlled manipulator. *Nucleonics (U.S.) Ceased publication*, 12(11).

Lawrence, D. (1992). Stability and transparency in bilateral teleoperation. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 2649–2655 vol.3.

Lei, Y., Su, Z., and Cheng, C. (2023). Virtual reality in human-robot interaction: Challenges and benefits. *Electronic Research Archive*, 31(5):2374–2408.

Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., Kurenkov, A., Liu, C. K., Gweon, H., Wu, J., Fei-Fei, L., and Savarese, S. (2021). iGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks.

Lichiardopol, S. (2007). *A survey on teleoperation*. DCT rapporten. Technische Universiteit Eindhoven. DCT 2007.155.

Melchiorri, C. (2013). *Robot Teleoperation*, pages 1–14. Springer London, London.

Miyazaki, F., Matsubayashi, S., Yoshimi, T., and Arimoto, S. (1986). A new control methodology toward advanced teleoperation of master-slave robot systems. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 997–1002.

Navarro, D. and Sundstedt, V. (2019). Evaluating player performance and experience in virtual reality game interactions using the htc vive controller and leap motion sensor. pages 103–110.

Noor, A. K. (2016). The hololens revolution. *Mechanical Engineering*, 138(10):30–35.

Oumaima, D., Mohamed, L., Hamid, H., and Mohamed, H. (2024). Application of artificial intelligence in virtual reality. In Lanka, S., Sarasa-Cabezuelo, A., and Tugui, A., editors, *Trends in Sustainable Computing and Machine Intelligence*, pages 67–85, Singapore. Springer Nature Singapore.

Pointecker, F., Friedl, J., Schwajda, D., Jetter, H.-C., and Anthes, C. (2022). Bridging the gap across realities: Visual transitions between virtual and augmented reality. In *2022 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 827–836. IEEE.

ROS.org (2024). Unified Robot Description Format (URDF) in ROS. http://wiki.ros.org/urdf. Accessed: 2024-06-04.

Shen, B., Xia, F., Li, C., Martín-Martín, R., Fan, L., Wang, G., Pérez-D'Arpino, C., Buch, S., Srivastava, S., Tchapmi, L. P., Tchapmi, M. E., Vainio, K., Wong, J., Fei-Fei, L., and Savarese, S. (2021). iGibson 1.0: a Simulation Environment for Interactive Tasks in Large Realistic Scenes.

Sheridan, T. B. (1989). Telerobotics. *Automatica*, 25(4):487–507.

Sheridan, T. B. and Ferrell, W. R. (1963). Remote manipulative control with transmission delay. *IEEE Transactions on Human Factors in Electronics*, (1):25–29.

Sherman, W. R. and Craig, A. B. (2003). Understanding virtual reality. *San Francisco, CA: Morgan Kauffman.*

Valve Corporation (2024). SteamVR. https://www.steamvr.com. Computer software.

Yokokohji, Y. and Yoshikawa, T. (1994). Bilateral control of master-slave manipulators for ideal kinesthetic coupling-formulation and experiment. *IEEE Transactions on Robotics and Automation*, 10(5):605–620.

Príloha A: Zdrojový kód a dáta z testovania

Všetky súbory teleoperačného systému nájdete na webovej adrese `https://github.com/AdamHuserka/bachelor-thesis/tree/main/iGibson/needed_files`. Nájdete tam súbory `nico.py`, `nico.yaml`, `simulator_nico_vr.py`, `nico_upper_head_rh6d_dual.urdf` a `vr_config.yaml` spolu s dvoma priečinkami, v ktorých sú meshe robota.

Pre spojazdnenie nášho programu vložte jednotlivé súbory a priečinky do súborového stromu simulátora nasledovne:

- Súbor `nico.py` vložte do priečinku ./igibson/robots.
- Súbor `nico.yaml` vložte do priečinku ./igibson/configs/robots.
- Súbor `simulator_nico_vr.py` vložte do priečinku ./igibson.
- Súbor `nico_upper_head_rh6d_dual.urdf` spolu s priečinkami `mesh` a `meshes` vložte do priečinku ./igibson/data/assets/models/nico.
- Súbor `vr_config.yaml` vložte do priečinku ./igibson.

Testovanie rozhrania prebiehalo pomocou scén, ktoré nájdete na webovej adrese `https://github.com/AdamHuserka/bachelor-thesis/tree/main/iGibson/testing_scripts/vr`. Nájdete tam 3 scény: `vr_teleop_demo.py`, `vr_light_objects_grasping.py` a `vr_heavy_objects_grasping.py`.

Na správne fungovanie scén je potrebné mať stiahnuté autormi poskytované datasety 'g' a 'ig'. Scéna `vr_heavy_objects_grasping.py` vyžaduje, aby v URDF súbore objektu lopty bola nastavená jeho hmotnosť aspoň na hodnotu 3 kg. URDF súbor nájdete v súborovom strome tu: ./igibson/data/ig_dataset/objects/ball/ball_000/ball_000.urdf.

Ovládanie rozhrania sa líši podľa použitého VR hardvéru. Definície mapovania jednotlivých tlačidiel k funkcionalite nájdete v súbore `vr_config.yaml`. V tomto súbore definujeme mapovanie pre aktiváciu teleoperácie a resetovanie polohy robota. V súbore `simulator_nico_vr.py` je v metóde *gen_vr_robot_action* implementované zisťovanie stavu úchopu od konkrétnych tlačidiel ovládačov, a preto mapovanie pre úchop nie je zahrnuté v súbore `vr_config.yaml`.

Dáta z testovania rozhrania sú prístupné na adrese: `https://drive.google.com/drive/folders/1wqkMHBp6a9BC3PO3sU1XBkxSYSn03sPI?usp=sharing`.