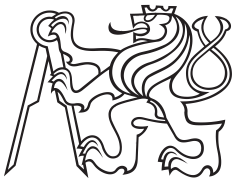
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Automatické vytváření trajektorie robota pro svařování

Kamil Horný

Vedoucí: Ing. Vladimír Smutný, Ph.D.
Obor: Kybernetika a robotika
Studijní program: Kybernetika a robotika
Květen 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Horný** Jméno: **Kamil** Osobní číslo: **457175**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Automatické vytváření trajektorie robota pro svařování

Název diplomové práce anglicky:

Automatic Trajectory Planning for Robot Welding

Pokyny pro vypracování:

1. Seznamte se s problémem svařování plastů extruderem.
2. Seznamte se se systémem Robot Operating System (ROS), knihovnou Open Motion Planning Library (OPML) a problematikou plánování trajektorie redundantních robotů.
3. Vyberte nebo navrhněte vhodnou metodu pro plánování pohybu.
4. Implementujte v ROSu vybraný systém plánování, zohledněte praktické požadavky na funkci systému.
5. Proveďte experimenty v simulátoru a výsledky zhodnoťte. Na základě výsledků simulací formulujte doporučení pro realizaci.

Seznam doporučené literatury:

[1] A.Gasparetto, V.Zanotto: A new method for smooth trajectory planning of robot manipulators, Mechanism and Machine Theory, Volume 42, Issue 4, April 2007, Pages 455-471, Elsevier
[2] ATEF A. ATA: OPTIMAL TRAJECTORY PLANNING OF MANIPULATORS: A REVIEW, Journal of Engineering Science and Technology, Vol. 2, No. 1 (2007) 32-54, School of Engineering, Taylor's University College
[3] Jinchao Guo, Xinjin Wang and Xiaowan Zheng, "Trajectory planning of redundant robot manipulators using QPSO algorithm," 2010 8th World Congress on Intelligent Control and Automation, Jinan, 2010, pp. 403-407, doi: 10.1109/WCICA.2010.5553846.
[4] Robot Operating System (ROS), https://www.ros.org/
[5] Open Motion Planning Library (OPML), https://ompl.kavrakilab.org/

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Vladimír Smutný, Ph.D., robotické vnímání CIIRC

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

Datum zadání diplomové práce: **25.01.2021** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce letního semestru 2021/2022

Ing. Vladimír Smutný, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Rád bych poděkoval své rodině a přátelům za trpělivost a podporu v průběhu celého studia. Dále děkuji svému vedoucímu Ing. Vladimíru Smutnému, Ph.D. za technickou a informativní podporu při vypracovávání diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2021

Abstrakt

Tato diplomová práce se zabývá automatickým plánováním trajektorie pro redundantního robota s 9 DoF. V první části jsou popsána současná řešení využívaná k hledání trajektorie pro redundantní roboty. Dále se práce věnuje popisu robotické svařovací buňky a použitým softwarovým nástrojům. V následující části jsou podrobně popsána kritéria, která slouží k optimálnímu plánování pohybu robota. V práci je dále zdokumentována implementace od vytvoření modelu svařovací robotické buňky až po integraci softwaru, který byl použit k vyřešení zadaného úkolu. Na závěr je plánování trajektorie otestováno a vyhodnoceno na základě experimentů.

Klíčová slova: plánování, trajektorie, optimalizace, OMPL, MoveIt

Vedoucí: Ing. Vladimír Smutný, Ph.D.
Český institut informatiky, robotiky a kybernetiky, ČVUT,
Jugoslávských partyzánů 1580/3,
160 00 Praha 6

Abstract

This master's thesis focuses on automatic planning of redundant robot with 9 DoF. The first part is dedicated to the state of the art redundant robots planning. The next part describes robotic welding cell and software used for trajectory planning. Attention is also paid to optimizing criteria used for determining optimal planning of the robot's movements. The thesis further describes the implementation process which includes creation of the workspace model and integration of used software. The final part describes experiments of trajectory planning and their evaluation.

Keywords: planning, trajectory, optimization, OMPL, MoveIt

Obsah

Obrázky

Kapitola 1

Úvod

V průběhu posledních let je v průmyslu trendem zavádění robotizace a automatizace. Většina firem se musí v rámci zachování konkurenceschopnosti se tomuto trendu podřídit a přizpůsobit. Použití robotů ve výrobě má nesporné výhody v mnoha směrech. Robotické manipulátory jsou přínosem při vykonávání fyzicky náročné práce nebo v práci ve zdraví nebezpečném prostředí. Mezi další výhody automatizace výroby patří omezení závislosti na lidských faktorech. Roboti nemohou onemocnět a nepotřebují pauzy či dovolenou a, navíc můžeme jejich použitím minimalizovat množství chyb při vykonávání repetetivních a stereotypních úkonů. Na druhou stranu je ale potřeba počítat s vyššími požadavky na dostatečnou kvalifikaci personálu a s dalšími možnými náklady na údržbu.

Česká firma STP Plast s.r.o. se zaměřuje na výrobu a montáž plastových dílů. Jejich portfolio tvoří plastové technologické celky, nádrže, jímky, potrubní rozvody a bazény. Sídlo firmy včetně výrobní haly se nachází ve Stráži pod Ralskem. Společnost působí na trhu již od roku 1997 a své produkty dodává zákazníkům nejen v Čechách ale i po celé Evropě. Vzhledem k neustále rostoucímu konkurenčnímu na mezinárodní úrovni se jednatelé firmy shodli na modernizaci výroby s použitím robotů. Dalším motivačním faktorem je problém najít dostatečně kvalifikovaný personál, kde zaučení jednoho zaměstnance trvá až několik měsíců.[1]

Jako první projekt robotizace si společnost vybrala automatické svařování plastových nádrží. Jedná se o nádrže tvaru kvádru a válce, velikost půdorysu může být maximálně 2×2 m. Jelikož společnost nemá ve směru automatizace dostatek zkušeností a kvalifikovaného personálu, dohodla se na spolupráci

s průmyslovým partnerem Triotec s.r.o., který se na tuto oblast specializuje.

Firma STP Plast s.r.o. zákazníkům nabízí osobní přístup a úprava výrobků na míru již od návrhu produktu, neexistuje tedy fixní množství variací nádrží. Vzhledem k velikosti a značné složitosti výrobků není dostačující využití pouze standardního robota s 6 stupni volnosti připevněného na statickou konstrukci, ale je potřeba řešení rozšířit o externí pohyblivé klouby a ramena. Cílem mé diplomové práce je připravit plánování trajektorie pro celou svařovací robotickou buňku, na základě dat vygenerovaných 3D CAD/CAM softwarem vzniklých při konstrukčním návrhu nádoby.

Obrázek 1.1: Ukázka plastové nádrže od firmy STP Plast s.r.o.[1]

Kapitola 2

State of the art plánovacích algoritmů pro redundantní roboty

V této práci se zaměřuji na plánování trajektorie redundantního robotu. Redundantní manipulátory se od klasických liší tím, že mají více stupňů volnosti neboli Degrees of Freedom - DoF. Počet stupňů volnosti u mechanismu znamená, jaký je počet nezávislých parametrů, které ho v prostoru jednoznačně popisují. Kinematická redundance u robotů nastává v případě, kdy má manipulátor více stupňů volnosti, než potřebuje pro vykonání žádaného pohybu. V současnosti jsou za standardní neredundantní roboty považovány takový, který mají do 6 DoF. Právě 6 stupňů volnosti je potřeba pro jednoznačný popis tělesa ve 3D prostoru. Za typického redundantního robota s 7 DoF může být považován například LBR iiwa od firmy KUKA.

Řešení přímé kinematické úlohy není u redundantních robotů výpočetně náročnější oproti standardním. Hlavní rozdíl nastává u řešení inverzní kinematické úlohy, kde mají neredundantní roboti pouze omezený počet řešení. Výjimku tvoří pouze singularity, stavy které mají nekonečně mnoho řešení a je nutné se jim vyvarovat kvůli nežádoucímu mechanickému namáhání. U manipulátorů s redundantními stupni volnosti existuje pro všechny stavy nekonečně mnoho řešení inverzní kinematické úlohy. Tuto problematiku můžeme řešit jako optimalizační úlohu, kde každý stav lze popsat pomocí charakteristické hodnoty. Tento optimalizační parametr poté popisuje vhodnost jednotlivých stavů v rámci kontextu celého pracovního prostoru robota. Na inverzní kinematickou úlohu lze následně nahlížet jako na optimalizaci tohoto parametru. Při řešení je nezbytné detekovat a eliminovat stavy, které se nacházejí v kolizi s prostředím buňky.[4]

V současné době existuje veliké množství algoritmů, které mohou být použity pro bezkolizní plánování trajektorie pro redundantní manipulátory. Tyto algoritmy se od standardních liší pouze v rozšířeném popisu prostoru o další stupně volnosti. Většina plánovacích algoritmů využívá principů a metod umělé inteligence.[5]

Při své práci jsem využil systém ROS - Robotic Operating System a jeho nadstavbovou knihovnu MoveIt!, která je připravena pro manipulační úlohy robotů. V ROSu, případně MoveItu existuje možnost využití některé z připravených knihoven. Ty mají funkční implementované komunikační rozhraní ulehčující jejich nastavení. Nejpoužívanější z těchto knihoven jsou OMPL - Open Motion Planning Library a CHOMP - Covariant Hamiltonian Optimization for Motion Planning.[6]

2.1 OMPL - Open Motion Planning Library

V knihovně OMPL jsou implementovány plánovací algoritmy pracující s diskrétním stavovým prostorem. Koncept vzorkování se využívá pro jednodušší dotazování plánovacích dotazů v prostoru s velký počtem stupňů volnosti nebo s vysokým počtem překážek. Hlavní výhodou tohoto postupu je výrazné snížení výpočetní náročnosti, například při detekci kolizí nebo při závěrečném zpracování trajektorie.

OMPL lze označit za abstraktní plánovací knihovnu, neobsahuje žádné nástroje sloužící k detekci kolizí nebo vizualizaci. Chybějící funkce lze vyřešit použitím dalších nástrojů, například pomocí ROSu a MoveItu v nichž jsou tyto funkce již integrované. Knihovna dále obsahuje funkce, umožňující optimalizaci nalezených trajektorií.[7]

Open Motion Planning Library obsahuje například implementace následujících algoritmů a jejich různé variace:

- **Rapidly-exploring Random tree - RRT**

  RRT je navržen k efektivnímu prohledávání nekonvexního vícedimenizionálního prostoru. Struktury jsou tvořeny inkrementálním prozkoumávání uzlů s cílem co nejvíce minimalizovat vzdálenost od cíle.

  Algoritmus je vhodný právě pro řešení plánovacích úloh v neholonomním prostředí. Jeho samostatné použití ale není mnohdy dostatečné, jelikož

výsledná trajektorie není dostatečně plynulá. A proto se musí vygenerovaná cesta dále optimalizovat pro reálné využití. Pro vylepšení vlastností prohledávání bylo odvozeno mnoho výpočetně efektivnějších variant: RRT Connect, RRT*, Lazy RRT, Sparse Table RRT,...[8]

- **Probabilistic roadmaps - PRM**

  Algoritmus PRM je složen ze dvou hlavních fází, konstrukční a dotazovací. V první fázi je vytvořena tzv. roadmapa, která vznikla propojením n nekolizních stavů z navzorkovaného stavového prostoru. Ve druhé dotazovací fázi se již hledá trajektorie na připravené roadmapě.

  PRM umožňuje tzv. multi-query plánování, tzn. že na již připravené roadmapě je algoritmus schopen odpovědět více dotazů na naplánování cesty. Z PRM je podobně jako z RRT odvozeno několik variant, například LazyPRM, PRM*, LazyPRM*.[8]

- **Expansive Space Trees - EST**
- **SPArse Roadmap Spanner algorithm - SPARS**
- **Kinematic Planning by Interior-Exterior Cell Exploration - KPIECE**

2.2 CHOMP - Covariant Hamiltonian optimization for motion planning

CHOMP na rozdíl od většiny ostatních plánovacích procedur nerozděluje proces hledání optimální trajektorie na dvě po sobě následující fáze, hledání cesty a její následné optimalizace. Na začátku procesu je dána kolizní trajektorie. Následně algoritmus využívá gradientního přístupu v optimalizační fázi. Při ní se CHOMP snaží po nejrychlejší dráze vyhnout možným kolizím. V rámci optimalizace jsou také brány v potaz dynamické aspekty, například omezení rychlostí a zrychlení kloubů.

CHOMP nepatří mezi kompletní plánovací metody. Plánovací algoritmus je kompletní, jestliže dokáže v konečném čase na každý plánovací dotaz nalézt řešení a nebo potvrdí, že žádné neexistuje. CHOMP tyto podmínky nesplňuje a proto může se stát, že při prohledávání prostoru skončí pouze v lokálním minimu.[9]

2.3 Ant colony algoritmus

Princip algoritmu Ant colony vychází z chování mravenců při hledání potravy, kteří na cestě zanechávají feromonovou stopu. Tato stopa slouží mravencům k nalezení nejkratší cesty, jelikož feromony postupně vyprchávají a na delší cestě tedy rychleji mizí. V oblastech robotiky a plánování jsou mravenci nahrazeni agenty, kteří spolu navzájem komunikují.

Algoritmus je používán pro generování optimálních trajektorií redundantních manipulátorů. Mezi významné aplikace patří plánování průmyslových procesů, zejména obrábění a aditivní výroba. [10]

2.4 Evoluční algoritmus využívající GPU pro plánování trajektorií redundantních robotů

Evoluční algoritmy jsou podobně jako Ant Colony inspirovány přírodou. V průběhu vývoje se mění struktura populace. Na základě Darwinovy teorie o přirozeném výběru přežijí pouze ti nejsilnější. Algoritmy se snaží využít evolučních principů tak, aby nalezly řešení náročných a rozsáhlých úloh.

Pro plánování průmyslových manipulátorů je možné využít i evoluční algoritmy. Podrobněji se o nich píše v článcích [11], [12] a [13]. U redundantních manipulátorů nastává v použití evolučních algoritmů největší problém s výpočetní náročností. V textu [14] je ukázáno, že problém s efektivitou lze překonat použitím paralelní architektury.

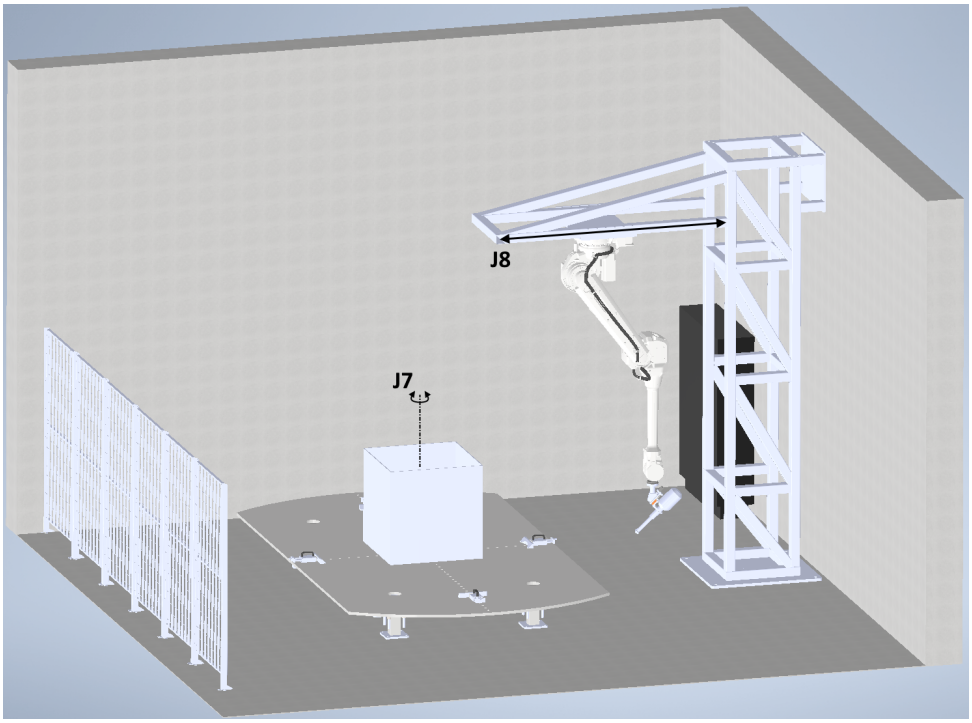
Kapitola 3

Popis problému

Návrh řešení celé buňky vytvořili konstruktéři z firmy Triotec s.r.o. Při jeho tvorbě hrály hlavní roli faktory: technická omezení, velikost zařízení a výsledná cena. Firma STP Plast disponuje pouze omezenými prostory pro výrobu, proto je kladen důraz na to, aby byla výsledná buňka z prostorového hlediska co nejúspornější. S tím souvisí i celková cena zařízení, kterou je potřeba v rámci schváleného rozpočtu nepřesáhnout. Současně ale musí výsledná konstrukce umožňovat dodržení správného pracovního postupu, vyžadovaného při svařování plastů.

3.1 Hardware

V původním řešení buňky se konstruktéři snažili eliminovat komplexnost zařízení použitím pouze dvou externích os, včetně robota se u zařízení předpokládalo využití 8 DoF. V návrhu byl použit robot IRB 4600 od firmy ABB. Řešení je zobrazeno na obrázku 3.1. Model bylo ale nutné upravit, neboť po bližším prozkoumání nebylo možné splnit požadavky kladené na výslednou funkčnost celého zařízení. V druhém návrhu se již počítá s použitím 3 externích os, celkově tedy 9 DoF. V obou případech se uvažovalo s připevněním základny robota k posuvnému kloubu umístěného ke stropu a natočeného směrem k zemi.



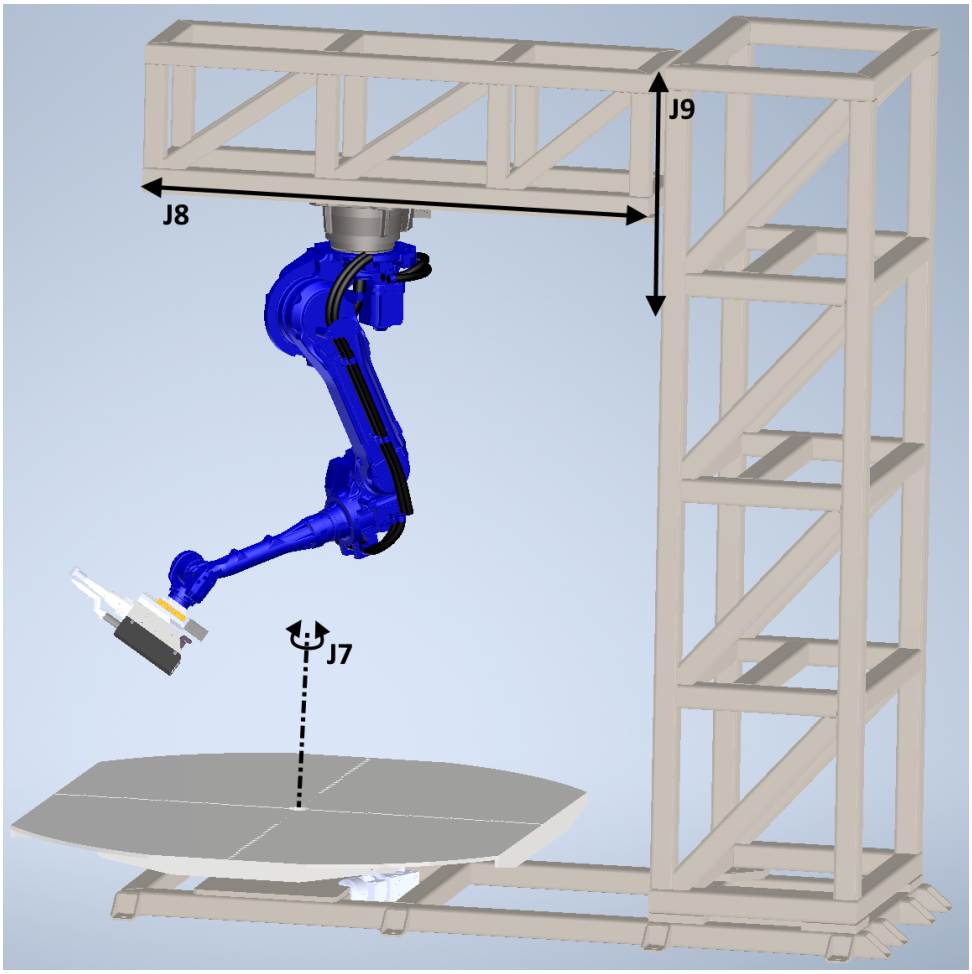
Obrázek 3.1: Původní návrh svařovacího robotického pracoviště

Výsledné předpokládané rozměry zařízení jsou 5600 × 3400 × 4300 mm. Svařovaná plastová nádoba bude připevněna na řízeném otočném stole ve výšce 43 mm. U posuvných kloubů jsem v simulaci pracoval s rozsahem 2250 mm u J8 a 750 mm u J9. Názorné umístění kloubů je zvýrazněno na obrázku 3.2.

3.1.1 Extruder

Při svařování plastových nádrží se nejdříve horkým vzduchem zahřejí oba plasty v místě dotyku. Do spoje je následně pod tlakem natlačen roztavený plast, které oba kusy propojí.[15] Pro tento projekt bylo vybráno svařovací extruder LEISTER WELDPLAST 200-i.

Použitý extruder umožňuje nahřívat spojované plasty horkým vzduchem o teplotě až 350 °C. Svařovací materiál je přiváděn do extruderu ve formě struny o průměru 3 nebo 4 mm je zahřátý na 260 °C. Hmotnost samostatného extruderu je 15 kg a jeho rozměry jsou 660 × 191 × 210 mm. Extruder je zobrazen na obrázku 3.3.[2]



Obrázek 3.2: Vylepšený návrh robotického pracoviště s 9 DoF

Obrázek 3.3: Extruder Leister Weldplast 200-i [2]

3.1.2 Robot

Původně zamýšlený robot ABB IRB 4600 nebylo možné využít, protože disponoval nosností pouze 40 kg[16]. Extruder váží sice jen 15 kg, ale je potřeba ho pomocí příruby připevnit na zápěstí robota. Samotná příruba bude rozšířena o mechanické rozvolnění, pokrývající nepřesnosti mezi modelem a reálnou buňkou. Vyšší nároky na robustnost manipulátoru navíc zvyšuje potřeba vyvynutí přítlaku na svár v průběhu svařování. Z tohoto důvodu konstruktéři vybrali robota Motoman GP 88, robotické rameno s 6 DoF od firmy Yaskawa. GP 88 má maximální dosah 2236 mm a nosnost 88 kg.[3]

Obrázek 3.4: Robot Motoman GP 88[3]

3.2 Software

Následující sekce podrobněji popisuje použitý software a rozšiřující knihovny. Všechny programy, které jsem v práci použil jsou vedeny jako open source,

jejich zdrojový kód je dostupný zdarma a může být volně modifikován a sdílen při dodržení autorské licence.[17]

3.2.1 Robot Operating System - ROS

ROS je flexibilní framework, který se používá k vývoji a testování programů pro roboty. S pomocí ROSu jsou vývojáři schopni software otestovat na robotech od různých výrobců. Díky tomu jsou inženýři schopni otestovat své řešení a částečně eliminovat chyby způsobné špatným návrhem řešení daného problému.

ROS umožňuje připojení hardwaru, předávání zpráv mezi jednotlivými procesy a sjednocuje rozhraní na správu jednotlivých nezávislých balíčků. Systém tvoří široká kolekce knihoven a nástrojů usnadňující práci s celou platformou. Hlavní programovací jazyky, podporované pro psaní zdrojového kódu, jsou C++, Python a Lisp. Další programovací jazyky jako Java, Go nebo Node.js jsou vedeny pouze jako experimentální, a při jejich použití může programátor narazit na mnoho problémů.

Výsledný spustitelný program v ROSu je tvořen uzly, které spolu komunikují. Předávání informací může probíhat různými způsoby, pomocí Topiců ve formě Publisher a Subcriber, s využitím Messages způsobem Client - Server a nebo uchováváním dat přímo na Parameter Serveru.

Roboti jsou v ROSu reprezentováni ve formátu URDF - Unified Robot Description Format, který odpovídá XML souboru. Pro zjednodušení popisu a současně i čitelnosti kódu je doporučeno použít makra, zvaná Xacra. O transformaci mezi jednotlivými souřadnicovými systémy obstarává knihovna TF. K vizualizaci modelu se používá integrovaný nástroj ROSu, RVIZ - ROS Vizalization.[18][19]

3.2.2 MoveIt

MoveIt patří mezi state of the art knihovny určené pro plánování pohybu robotů. Knihovna se specializuje na manipulaci s objekty, 3D vnímaní, řízení a navigování u robotů. MoveIt zároveň obsahuje moduly k detekci kolizí a řešení přímé a inverzní kinematické úlohy.

Přímo pro MoveIt existuje již více než 120 připravených reálných průmyslových modelů robotů, včetně jejich kinematických vlastností a kolizních modelů. Jedná se převážně o modely nejznámějších výrobců jako KUKA, ABB, Fanuc a jiných. K vizualizaci modelů využívá nástroj RVIZ, který je rozšířen o doplňující funkce.

Knihovna současně obsahuje připravený interface k dalším plánovacím knihovnám, například OMPL, CHOMP nebo Pilz Industrial Motion Planner. Jejich využití programátorům usnadňuje nastavení plánovacích algoritmů. Běžnému uživateli díky tomu stačí nastavit plánovací algoritmy pomocí několika konfiguračních souborů místo zdlouhavého a komplikovaného manuálního propojení jednotlivých komponent.[20]

3.2.3 Open Motion Planning Library - OMPL

OMPL je knihovna, která pokrývá celou sadu nejnovějších plánovacích algoritmů pracujících s navzorkovaným prostorem. Knihovnu využila například i NASA k plánování pohybu Robonauta2 na mezinárodní stanici ISS. Bližší informace o algoritmech implementovaných v OMPL jsou popsány v kapitole 2.[7]

3.2.4 Ceres-Solver

Pro řešení optimalizačních úloh jsem využil programový nástroj Ceres-Solver. Knihovna je určená pro modelování a řešení velkých optimalizačních problémů. Knihovna je psaná v jazyce C++. Od roku 2010 knihovnu ve svých programech využívá Google.[21]

Kapitola 4

Teoretická část

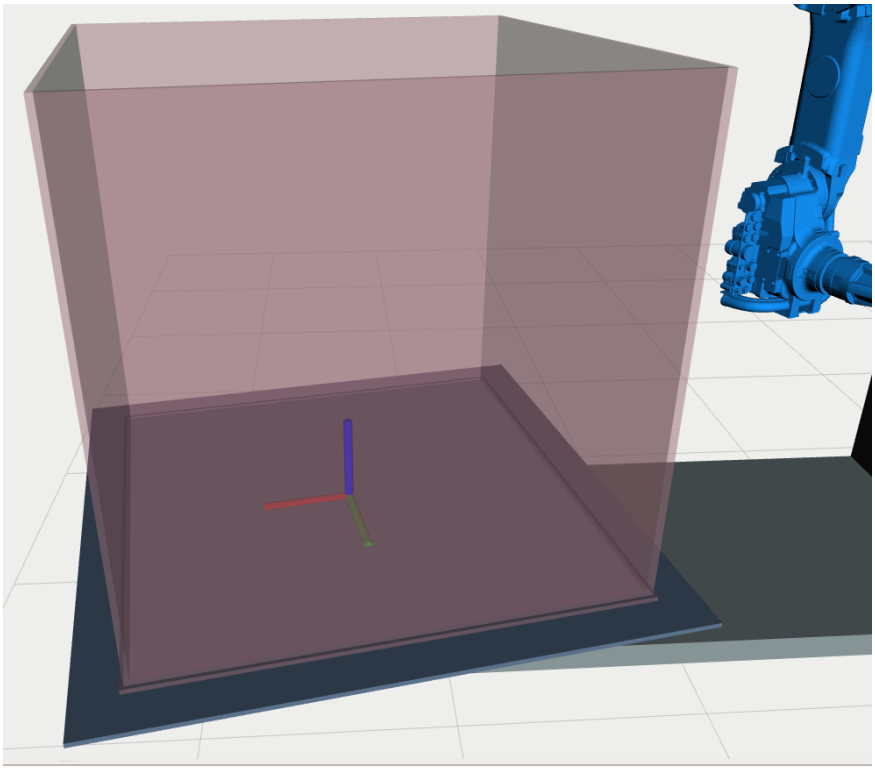
Při vypracování práce ještě nebyla přesně rozhodnuta výsledná forma vstupních dat. Jako dočasné řešení jsem zvolil zadávání svařovacích bodů pomocí vstupního souboru. Jeho formát je podobný G-codu, který bude v budoucnu generován pomocí CAD/CAM softwaru. V následující části jsou popsány jednotlivé komponenty, které jsou nutné k vygenerování trajektorie.

4.1 Plánování trajektorie

Při plánování trajektorie robota jsem pohyb rozdělil na dva typy, nájezd do počáteční svařovací polohy a na pohyb po přesně dané trajektorii. Před začátkem plánování je vždy zadána počáteční poloha manipulátoru v kloubových souřadnicích.

Pro bližší popis hledání trajektorie jsem musel definovat dva souřadnicové systémy. Prvním je souřadnicový systém nádrže. Počátek tohoto systému se nachází na průsečíku horní plochy otočného stolu a jeho osy otáčení, viz. obrázek 4.1. Poloha tohoto systému je současně shodná se souřadnicovým systémem nádrže, transformační matici mezi těmito systémy odpovídá jednotková matice. Druhý systém představuje souřadnicový systém extruderu. Počátek systému se nachází v koncovém bodu extruderu, ze kterého je vytlačován roztavený plast. Osa x je orientována po směru vytlačované hmoty. Osa y udává směr pohybu extuderu při svařování. Osa z doplňuje ostatní tak, aby dohromady tvořily pravoúhlý pravotočivý souřadnicový systém. Systém

extruderu je zobrazen na obrázku 4.2. Osa x je znázorněna červeně, osa y - modře a osa z - zeleně. Toto značení dodržuje obecnou konvenci ROSu a u všech obrázků v této práce je dodrženo.

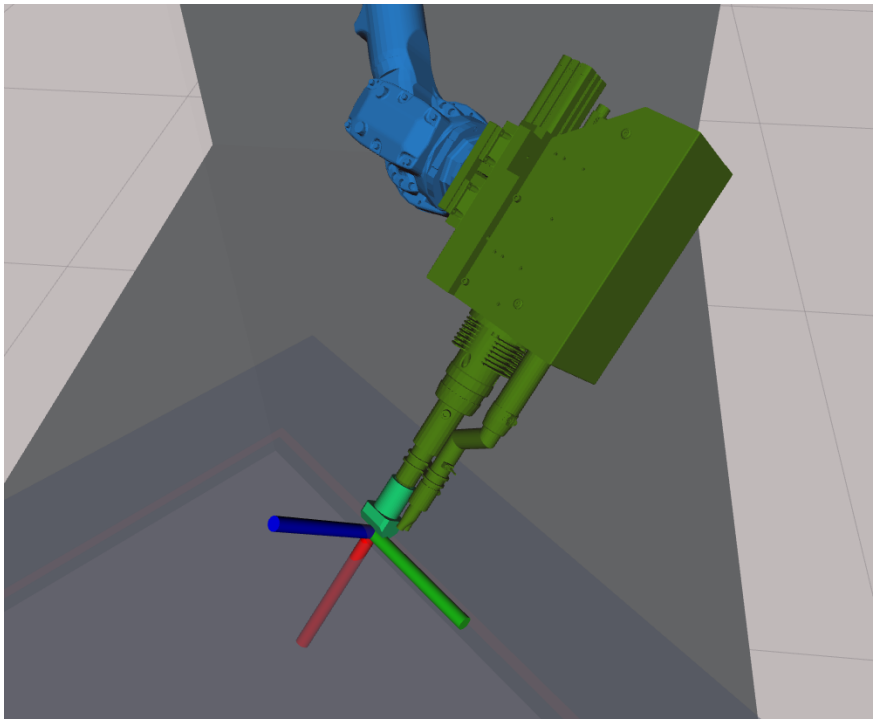


Obrázek 4.1: Souřadnicový systém nádrže

Poloha svařovacích bodů je zadána jako požadovaná poloha souřadnicového systému extruderu vyjádřená v souřadnicovém systému nádrže. Pro plánování nájezdové trajektorie známe cílovou polohu systému extruderu a počáteční pozici celého robota v kloubových souřadnicích. Při plánování svařovací trajektorie známe výchozí pozici robota, a zároveň máme určenou polohu extruderu v průběhu celého pohybu.

4.2 Optimalizace

Při hledání ideální trajektorie robota jsem musel brát v úvahu hned několik faktorů. Pro roboty je důležité se vyvarovat singulárním stavům. Singulární polohy manipulátoru jsou takové polohy koncového efektoru, které významně ovlivňují jeho kinematické vlastnosti. V blízkosti singulárních poloh je potřeba



Obrázek 4.2: Souřadnicový systém extruderu

pro malé rychlosti koncového efektoru vyvinout velkých kloubových rychlostí manipulátoru.[22] Dalším faktorem je snaha, aby byla naplánovaná trajektorie co nejkratší a pohyb robota plynulý. Současně je také potřeba dodržet správnou orientaci extruderu vůči svaru. Posledním důležitý faktor tvoří snaha vyvarovat se takovým polohám, kde se robot přibližuje nebezpečně blízko ke kolizi.

4.2.1 Vzdálenost dvou systémů

Na vzdálenost koncového efektoru od cílové polohy lze pohlížet jako na vzdálenost dvou souřadnicových systémů. V mém případě jsem samostatně specifikoval vzdálenost počátků těchto soustav a na rozdíl v jejich orientaci. V následujícím textu budu uvažovat dva souřadnicové systémy definované jejich počátečními body P_1 a P_2 a orientace vyjádřené pomocí kvatermionů $\boldsymbol{q}$ a $\boldsymbol{n}$.

$$P_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \; q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \; P_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}, \; n = \begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \end{bmatrix}. \tag{4.1}$$

4.2.2 Posunutí

První kritérium t_d představuje relativní vzdálenost bodu P_1 v počátku souřadnicové soustavy cíle a bodu P_2, který se nachází v počátku souřadnicového systému extruderu v místě, odkud je vytlačován roztavený plast. Hodnotu kritéria jsem vypočítal jako euklidovskou vzdálenost těchto bodů:

$$t_d = \|P_1 P_2\|. \tag{4.2}$$

4.2.3 Relativní orientace systémů

Další kritérium představuje relativní orientaci soustavy extruderu vůči soustavě cíle. Orientace jednotlivých soustav jsou vyjádřeny pomocí kvaternionů, z toho důvodu je nejprve potřeba specifikovat operace násobení kvaternionů a inverze kvaternionu.

Operace s kvaterniony

Kvaternion $\boldsymbol{q} = (q_0, q_1, q_2, q_3)^T$ je uspořádaná čtveřice čísel, kde q_0, q_1, q_2, q_3 jsou reálná čísla. Kvaternion $\boldsymbol{q}$ lze také zapsat ve formě zápisu:

$$q = q_0 + q_1 i + q_2 j + q_3 k, \tag{4.3}$$

kde i, j, k představují imaginární jednotky pro které platí:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{4.4}$$

Máme zadány dva kvaterniony $\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k$ a $\boldsymbol{p} = p_0 + p_1 i + p_2 j + p_3 k$. Jejich násobek $\boldsymbol{qp}$ potom bude roven:

$$\begin{aligned} qp = &(q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3) \; + \\ &(q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2)\, i \; + \\ &(q_0 p_2 - q_1 p_3 + q_2 p_0 + q_3 p_1)\, j \; + \\ &(q_0 p_3 + q_1 p_2 - q_2 p_1 + q_3 p_0)\, k. \end{aligned} \tag{4.5}$$

Konjugovaný kvaternion $\bar{\boldsymbol{q}}$ ke kvaternionu $\boldsymbol{q}$ se vypočítá pomocí vzorce:

$$\bar{q} = q_0 - q_1 i - q_2 j - q_3 k. \tag{4.6}$$

Inverzní kvaternion $\boldsymbol{q^{-1}}$ je definován pomocí vzorce:

$$q^{-1} = \frac{\bar{q}}{q\bar{q}} = \frac{\bar{q}}{N(q)}, \tag{4.7}$$

kde $N(q) = q_1^2 + q_2^2 + q_3^2 + q_4^2$ je norma kvaternionu $\boldsymbol{q}$.[23],[24]

Výpočet kritéria

Orientace obou systémů jsou vyjádřené pomocí kvaternionů. Kvaternion relativní rotace mezi soustavami $\boldsymbol{r}$ spočítám vynásobením kvaternionu popisujícího orientaci systému cíle $\boldsymbol{q_g}$ a inverzí kvaternionu vyjadřující orientaci souřadnicového systému efektoru $\boldsymbol{q_e}$:

$$r = \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = q_g \; q_e^{-1}. \tag{4.8}$$

Kvaternion $\boldsymbol{q_d}$ převedeme na vyjádření rotace pomocí osy a úhlu. Získaný úhel vyjadřuje požadovanou úhlovou vzdálenost a_d:

$$a_d = 2 \arccos(r_0). \tag{4.9}$$

4.2.4 Manipulovatelnost

Jakobián v robotice vyjadřuje vztah mezi rychlostmi kloubů robota $\dot{q}$ a rychlostí pohybu koncového efektoru $\dot{X}$. Vzájemnou závislost můžeme vyjádřit

pomocí vzorce:

$$\dot{X} = J\dot{q}. \tag{4.10}$$

V případě mé práce představuje $\dot{X}$ sloupcový vektor vyjadřující rychlost pohybu extruderu v souřadnicovém systému nádrže a $\dot{q}$ sloupcový vektor reprezentující rychlost otáčení nebo posuvu jednotlivých kloubů robota:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}, \; q = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{q_5} \\ \dot{q_6} \\ \dot{q_7} \\ \dot{q_8} \\ \dot{q_9} \end{bmatrix}, \tag{4.11}$$

kde $q_1, q_2, q_3, q_4, q_5, q_6$ odpovídají kloubovým souřadnicím robotického ramene GP 88, q_7 představuje kloubovou souřadnici otočného stolu a q_8 a q_9 odpovídají posuvným kloubům. Jednotlivé prvky jakobiánu lze vyjádřit:

$$\Big[J\Big] = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} & \frac{\partial x}{\partial q_4} & \frac{\partial x}{\partial q_5} & \frac{\partial x}{\partial q_6} & \frac{\partial x}{\partial q_7} & \frac{\partial x}{\partial q_8} & \frac{\partial x}{\partial q_9} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} & \frac{\partial y}{\partial q_4} & \frac{\partial y}{\partial q_5} & \frac{\partial y}{\partial q_6} & \frac{\partial y}{\partial q_7} & \frac{\partial y}{\partial q_8} & \frac{\partial y}{\partial q_9} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial yz}{\partial q_3} & \frac{\partial z}{\partial q_4} & \frac{\partial z}{\partial q_5} & \frac{\partial z}{\partial q_6} & \frac{\partial z}{\partial q_7} & \frac{\partial z}{\partial q_8} & \frac{\partial z}{\partial q_9} \\ \frac{\partial \alpha}{\partial q_1} & \frac{\partial \alpha}{\partial q_2} & \frac{\partial \alpha}{\partial q_3} & \frac{\partial \alpha}{\partial q_4} & \frac{\partial \alpha}{\partial q_5} & \frac{\partial \alpha}{\partial q_6} & \frac{\partial \alpha}{\partial q_7} & \frac{\partial \alpha}{\partial q_8} & \frac{\partial \alpha}{\partial q_9} \\ \frac{\partial \beta}{\partial q_1} & \frac{\partial \beta}{\partial q_2} & \frac{\partial \beta}{\partial q_3} & \frac{\partial \beta}{\partial q_4} & \frac{\partial \beta}{\partial q_5} & \frac{\partial \beta}{\partial q_6} & \frac{\partial \beta}{\partial q_7} & \frac{\partial \beta}{\partial q_8} & \frac{\partial \beta}{\partial q_9} \\ \frac{\partial \gamma}{\partial q_1} & \frac{\partial \gamma}{\partial q_2} & \frac{\partial \gamma}{\partial q_3} & \frac{\partial \gamma}{\partial q_4} & \frac{\partial \gamma}{\partial q_5} & \frac{\partial \gamma}{\partial q_6} & \frac{\partial \gamma}{\partial q_7} & \frac{\partial \gamma}{\partial q_8} & \frac{\partial \gamma}{\partial q_9} \end{bmatrix}. \tag{4.12}$$

Manipulovatelnost (Dexterity) D se vypočítá z jakobiánu pomocí vzorce:

$$D = \frac{1}{\mathrm{cond}(J)}, \tag{4.13}$$

kde J představuje jakobián a cond(J) je podmíněnost matice J.

V našem případě ale není možné počítat manipulovatelnost pro celý jakobián. U našeho robota s 9 DoF má jakobián rozměr 6 $\times$ 9. Protože robot disponuje současně rotačními i posuvnými klouby, tak sloupce 8 a 9 jakobiánu odpovídají posuvným kloubům. V jakobiánu se nachází prvky s různými jednotkami, které jsou zobrazeny v rovnici 4.14.

$$J = \begin{bmatrix} \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & 1 & 1 \\ \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & 1 & 1 \\ \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & \frac{\mathrm{m}}{\mathrm{rad}} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \frac{\mathrm{rad}}{\mathrm{m}} & \frac{\mathrm{rad}}{\mathrm{m}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \frac{\mathrm{rad}}{\mathrm{m}} & \frac{\mathrm{rad}}{\mathrm{m}} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \frac{\mathrm{rad}}{\mathrm{m}} & \frac{\mathrm{rad}}{\mathrm{m}} \end{bmatrix} \tag{4.14}$$

Výpočet manipulovatelnosti jsem rozdělil na čtyři složky podle fyzikálních jednotek:

$$J_{6\times 9} = \begin{bmatrix} J_{t\ 3\times 7} & J_{p\ 3\times 2} \\ J_{r\ 3\times 7} & J_{d\ 3\times 2} \end{bmatrix}. \tag{4.15}$$

Matice J_d vyjadřuje závislost orientace koncového efektoru na posuvných kloubech j_8 a j_9. Tyto klouby nemají na orientaci extruderu žádný vliv a tudíž budou všechny prvky této matice rovny nule. Podmíněnost této matice je vždy rovna nekonečnu:

$$\operatorname{cond}(J_d) = \operatorname{cond}\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\right) = \infty. \tag{4.16}$$

V průběhu pohybu robotické buňky dochází u manipulovatelnosti k velkým rozdílů hodnot v porovnání s ostatními kritérii. Z toho důvodu jsem jako kritérium použil logaritmus těchto manipulovatelností.

Při optimalizaci jsem chtěl ceny stavů minimalizovat. Z toho důvodu jsem výsledný logaritmus vynásobil číslem -1, aby byl stav s horší manipulovatelností penalizován vyšší cenou. Výsledné manipulovatelnosti D_t, D_r a D_p vypočítám dle vzorců:

$$D_t = -\log\left(\frac{1}{\operatorname{cond}(J_t)}\right), \tag{4.17}$$

$$D_r = -\log\left(\frac{1}{\operatorname{cond}(J_r)}\right). \tag{4.18}$$

$$D_p = -\log\left(\frac{1}{\text{cond}(J_r)}\right). \tag{4.19}$$

4.2.5 Kolizní vzdálenost

Jednou ze složek kritéria tvoří vzdálenost ke kolizi d_c. Čím blíže se robot blíží ke kolizi, tím více je penalizován. Přesná podoba výpočtu kolize je popsána následujícím vzorcem:

$$d_c = -\log(c_d), \tag{4.20}$$

kde c_d značí nejmenší reálnou vzdálenost robota od okolních objektů. Vzdálenost c_d je získána s pomocí MoveItu a jeho třídy *PlanningScene*.[25] Logaritmus jsem zde použil proto, aby bylo více penalizováno velké přiblížení se robota ke kolizi.

4.2.6 Váhy

Výsledná cena C každého stavu je tvořena součtem všech vyjmenovaných složek, které jsou navíc vynásobeny příslušnou váhou α_i:

$$C = \alpha_1\ t_d + \alpha_2\ a_d + \alpha_3\ D_t + \alpha_4\ D_r + \alpha_5\ D_p + \alpha_6\ d_c. \tag{4.21}$$

Kapitola 5

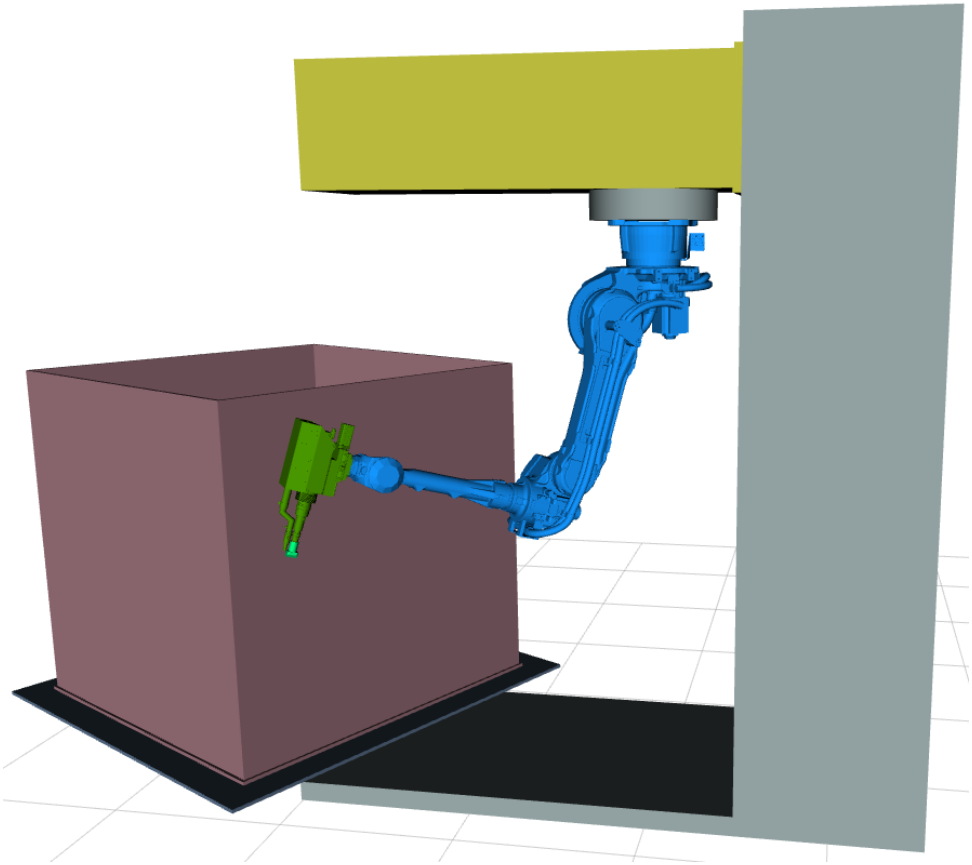
Implementační část

Pro vypracování diplomové práce jsem se rozhodl použít verzi ROSu Melodic Morenia. Tato distribuce je primárně určena pro operační systém Ubuntu 18.04. ROS disponuje i novějšími verzemi, které podporují i operační systém Windows 10 a MacOS, ale z důvodu lepší stability jsem si vybral starší a již prověřenou verzi systému.[26]

Dříve, než jsem v práci začal s programováním plánování robotické buňky, tak bylo potřeba připravit model celého zařízení. Postup implementace celé úlohy je popsán v následující části.

5.1 Vytvoření modelu

URDF popis celé buňky se skládá z řetězce ramen a kloubů. Tvary jednoduchých částí buňky popisuji pomocí připravených primitiv, kvádrů a válců. Pro účely vizualizace robota a extruderu jsem využil jejich 3D model ve formátu .stl. Jednotlivá ramena jsou pospojována pomocí tří typů kloubů: posuvných, rotačních a fixních.[18] Pro lepší přehlednost modelu jsem jednotlivé prvky zvýraznil odlišnými barvami. Pevné nepohyblivé části jsou zbarvené šedě, posuvná externí ramena žlutě, svařovaná nádoba růžově, robotické rameno GP 88 modře a extruder včetně koncové botky zeleně. Dynamické vlastnosti jednotlivých ramen jsem v modelu nespecifikoval. Vizuální model je zobrazen na obrázku 5.1.



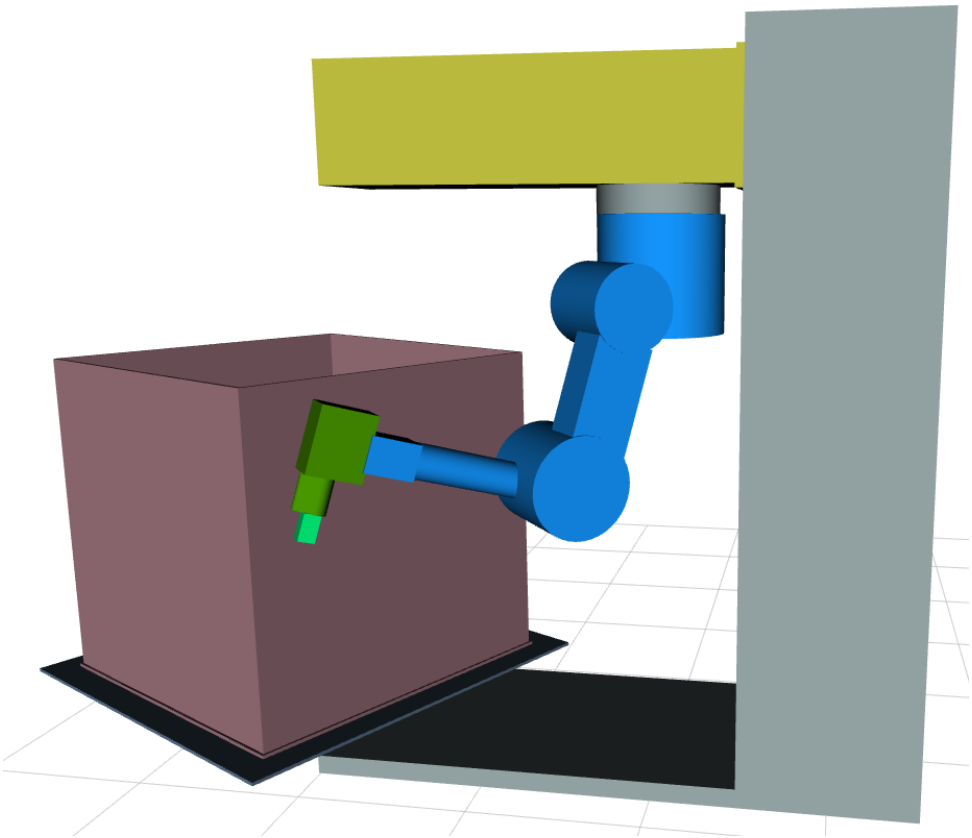
Obrázek 5.1: Vizuální model svařovací buňky

U modelu jsem pomocí nástroje MoveIt! Setup Assistant definoval dva koncové efektory.[27] První je umístěn na špičce extruderu, v místě, ze kterého bude vytlačován roztavený plast do sváru. Druhým je virtuální koncový efektor ve středu otočného stolu, ke kterému budou připevněny nádrže. V jeho souřadnicovém systému budou zadávány všechny pozice svarů.

5.1.1 Kolizní model

Relativně složité tvary robota a extruderu se negativně projevily na výpočetní náročnosti, což se projevilo dlouhým během optimalizačních algoritmů. Z toho důvodu jsem byl nucen zjednodušit kolizní model celého zařízení. Při úpravě modelu jsem se snažil zmenšit množství hran popisující komplikované objekty. Výsledný kolizní model je vyjádřen výhradně jen pomocí kvádrů a válců, viz. obrázek 5.2. Při plánování svářecí trajektorie je potřeba, aby se botka na konci extruderu (světle zelená koncová část extruderu na obrázcích 5.1 a 5.2), ze které je vytlačován roztavený plast, dotýkala nádoby.

Z toho důvodu je důležité pro plánování tohoto typu trajektorie vypnout detekce kolizí mezi botkou a nádobou.



Obrázek 5.2: Kolizní model svařovací buňky

5.2 MoveIt plánování

Původně jsem pro hledání trajektorií robota zamýšlel použít pouze knihovnu MoveIt. Pro plánování pohybu robota mezi jednotlivými svary jsem využil v MoveItu funkci *plan()* implementovanou ve třídě *MoveGroup*.[28]

Pro plánování svarů jsem využil funkci připravenou funkci *ComputeCartesianPath()*. Požadovaný svár je zadán v argumentu funkce jako vektor pozic, které budou po přímkách interpolovány a propojeny. Interpolací vznikne série poloh, kterými musí extruder postupně při pohybu projít. Souřadnice robota jsou pro jednotlivé polohy popořadě počítány jako řešení inverzní kinematické úlohy. Nevýhodou metody *ComputeCartesianPath()* je, že v momentě, když se optimální řešení inverzní kinematiky nachází v kolizi, tak hledání skončí chybou a nezkouší najít přípustné řešení.[28]

V momentě, kdy jsem se snažil naplánovanou trajektorii optimalizovat s cílem nalézt ideální řešení, tak jsem nedokázal překonat omezení MoveItu, který neumožňuje implementaci vlastního optimalizačního kritéria. Na výběr je zde pouze omezený počet již implementovaných kritérií, jako například optimalizace pro zisk co nejkratší trajektorie nebo maximalizace vzdálenosti robota od kolize s okolními objekty. Tato kritéria se spolu s nastavením plánovacího algoritmu specifikují v konfiguračním souboru připraveného rozhraní *OMPL_interface*.[29]

5.3 Napojení OMPL

Výše uvedený problém s optimalizací jsem se rozhodl vyřešit vlastním propojením MoveItu a OMPL. Pro základní funkce jako jsou: testování přípustnosti stavů robota, přepočet souřadnic mezi jednotlivými soustavami nebo simulace naplánované trajektorie, jsem využil hotovou implementaci v MoveItu.

Stavový prostor jsem v OMPL definoval pomocí třídy *RealVectorStateSpace* jako devítistupňový vektor odpovídající kloubovým souřadnicím robotické buňky. Pro testování validace stavů jsem OMPL napojil na třídu MoveItu *PlanningScene*.[25]

OMPL, v případě kdy nejsou známy přesné kloubové souřadnice cílového stavu, reprezentuje cíl pomocí třídy *GoalRegion*. Cílová poloha je vyjádřena v kartézském souřadnicovém systému nádrže a orientace je popsána pomocí kvaternionu.

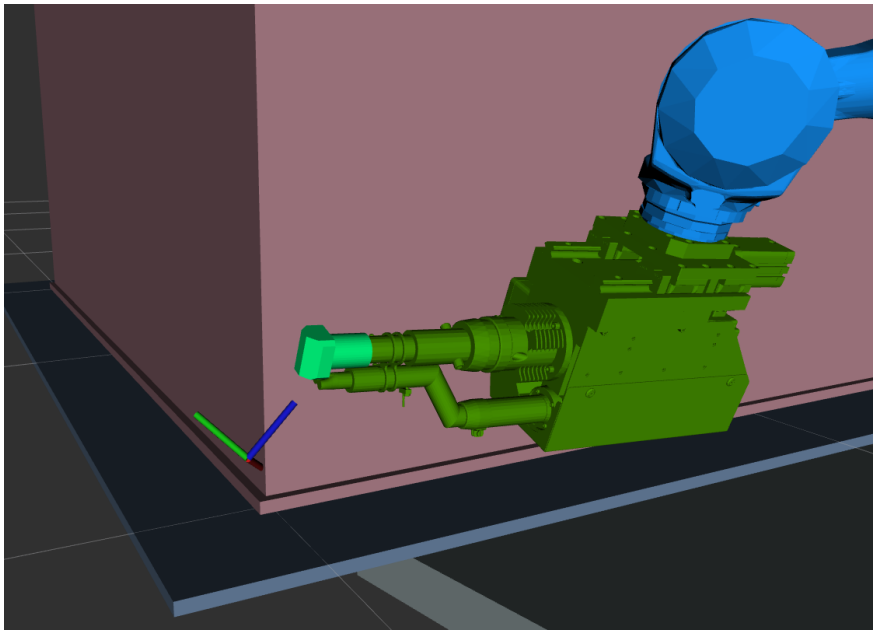
U třídy *GoalRegion* je nutné implementovat funkci *distanceGoal(const State *st)*. Metoda nejdříve pomocí přímé kinematické úlohy ze stavu *St* vypočítá polohu souřadnicového systému extruderu. Následně se vypočítá vzdálenost souřadnicových systémů extruderu a cíle. Návratová hodnota funkce je vzdálenost mezi těmito systémy. Vzájemná vzdálenost souřadnicového systému extruderu a souřadnicového systému cíle zde může být vyjádřena pouze jedním číslem. V mém případě jsem ale potřeboval minimalizovat nejen vzdálenost v posunutí koncového bodu extruderu, ale i relativní orientaci extruderu vůči požadované poloze.[30]

Celkovou vzdálenost jsem tedy definoval jako součet jejich vzájemného posunutí a úhlu vyjadřující jejich relativní rotaci. Rozdíl jednotek metrů a radiánů jsem vyřešil aproximací pomocí vah. Váhy jsem určil na základě experimentů. Úhel vyjadřující vzdálenost v orientaci jsem vynásobil váhou

0.25 na radián a u posunutí jsem použil váhu rovnu 1 na metr. Hranici přípustné vzdálenosti cílového regionu jsem nastavil na 0.03.

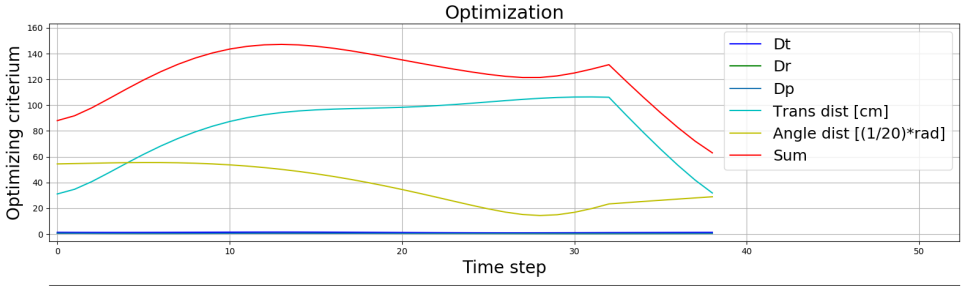
Optimalizační funkci trajektorie jsem definoval pomocí metody *stateCost(State *s)* a třídy *OptimizationObjective()*. Optimalizační kritéria jsem použil ta, která jsou uvedena v kapitole 4.2. Váhy jednotlivých složek jsem nastavil: pro vzdálenost v posunu koncového bodu extruderu od požadované polohy - 100 na metr, relativní rotace v natočení extruderu - 20 na radián, pro kritéria manipulovatelnosti i jsem nastavil váhu na 1 a pro kolizní vzdálenost 0. Při plánování OMPL testuje přípustnost stavů a z toho důvodu jsem si zde mohl dovolit vynechat kritérium kolizní vzdálenosti.[31]

Na obrázku 5.3 je znázorněna poloha cílového bodu pomocí soustavy barevných os. Extruder je na obrázku v pozici, ve které skončil po vykonání provedené trajektorie. Na první pohled je zřejmé, že koncový bod extruderu neodpovídá bodu v počátku barevné soustavy, představující požadovanou cílovou polohu. Tato skutečnost je potvrzena na obrázku 5.4, kde je zobrazen průběh optimalizačních kritérií při vykonávání trajektorie. Na konci grafu v časovém kroku 32 je vidět, že poloha extruderu je posunuta o 30 cm a orientace otočena přibližně o 145°.



Obrázek 5.3: Koncová poloha extruderu při plánování trajektorie pomocí RRT* algoritmu

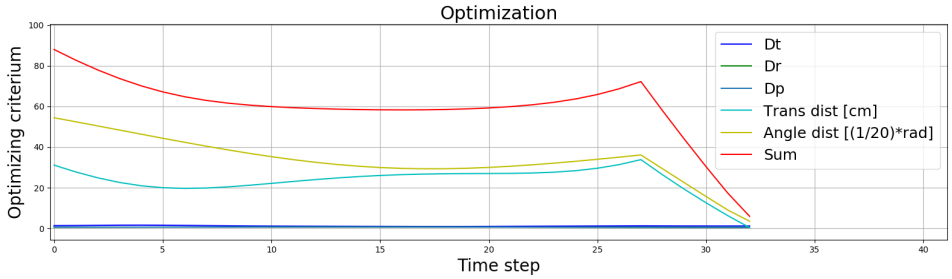
Úpravou parametrů RRT* algoritmu, ani úpravou vah optimalizačních kritérií, jsem nedokázal výsledky uspokojivě vylepšit. Maximální přijatelné odchylky požadované a naplánované polohy byly 1 cm v posunutí a 5° v orientaci.



Obrázek 5.4: Hodnota optimalizačních kritérií v průběhu naplánované trajektorie pomocí algoritmu RRT*

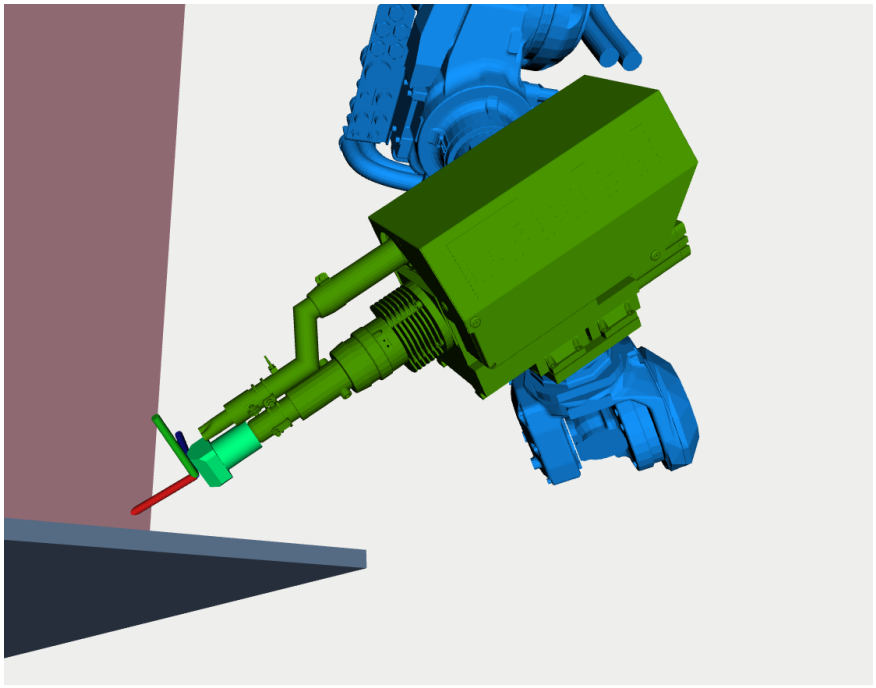
Pro vylepšení přesnosti jsem využil jiný algoritmus RRTConnect. Algoritmus, nejdříve s použitím genetického algoritmu a inverzní kinematické úlohy, nalezne hodnoty kloubů v cílové poloze. Následně se snaží s použitím dvojsměrného prohledávání tyto dva body propojit.

Na obrázku 5.5 je zobrazen graf s hodnotami optimalizačních kritérií v průběhu naplánované trajektorie. Z grafu je možné vidět, že na konci trajektorie se podařilo všechna kritéria minimalizovat téměř až k nule a dosáhnout tak požadované přesnosti.



Obrázek 5.5: Hodnota optimalizačních kritérií v průběhu naplánované trajektorie pomocí algoritmu RRTConnect

Na obrázku 5.6 je již vidět, že koncový bod extruderu, ze kterého je vytlačován roztavený plast, odpovídá požadovanému cílovému bodu v počátku soustavy znázorněné pomocí barevných os. Orientace polohy extruderu také odpovídá orientaci polohy znázorněné barevnou soustavou. Červená osa má směřovat ve stejném směru jako koncová botka extruderu. Zelená osa znázorňuje další směr pohybu při sváření, na extruderu musí stejným směrem směřovat otvor, ze kterého vychází horký vzduch za účelem nahřátí spojovaných plastů.



Obrázek 5.6: Koncová poloha extruderu při plánování trajektorie pomocí algoritmu RRTConnect

5.4 Optimalizace pomocí Ceres-Solver

Pro plánování pohybu k počátečnímu bodu svaru jsem s použitím knihovny OMPL dosáhl dostatečné přesnosti. Ale protože se mi nepodařilo s využitím OMPL dosáhnout svařovací trajektorie s přijatelně malou odchylkou, bylo potřeba k úloze plánování kartézské trajektorie přistoupit jiným způsobem.

Při zadané počáteční poloze manipulátoru jsem následující bod hledal jako optimalizaci 9 parametrů představujících kloubové souřadnice. K optimalizaci jsem použil nástroj Ceres-Solver, v něm bylo potřeba implementovat funkci, která vypočítává cenu jednotlivých stavů. Ceres-Solver se poté snažil optimalizovat hodnoty kloubových souřadnic tak, aby byla cena stavu co nejmenší.[32]

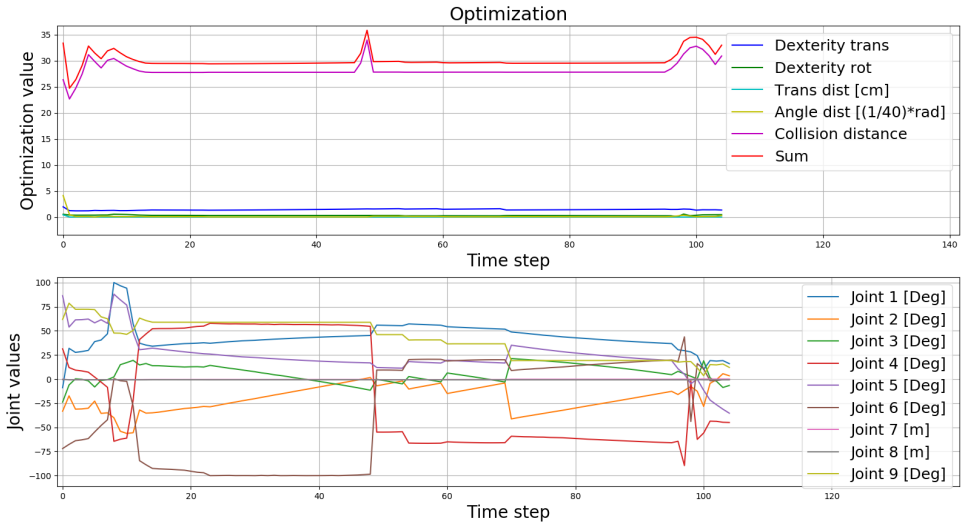
Ani při tomto způsobu jsem ale nedokázal naplánovat pohyb tak, aby se extruder pohyboval po zadané trajektorii. Při simulacích jsem dokázal minimalizovat pohybovat botkou na konci extruderu po vybrané přímce, ale odchylka v orientaci extruderu již přesahovala stanovené limity 5°. V momentě, kdy jsem zvýšil váhu úhlové vzdálenosti extruderu od požadované,

aby extruder udržel správnou orientaci, tak se postupně při pohybu zvyšovala vzdálenost jeho koncového bodu od požadovaného svaru. Ceres-Solver poté již hledal optimální stav robota v menších vzdálenostech od cílové polohy, problém s nepřesnostmi ale pořád přetrvával. Optimalizace vždy skončila v lokálním minimu.

5.5 Optimalizace v prostoru nižší dimenze za použití inverzní kinematiky

Po problémech s přesností jsem byl donucen přístup k řešení plánovacího úkolu ještě jednou pozměnit. Požadovaná poloha extruderu je vyjádřena pomocí 6 parametrů a samotné robotické rameno GP 88 má také 6 stupňů volnosti. Jsme tedy schopni pomocí řešení inverzní kinematické úlohy samotného ramene robota GP 88 jednoznačně nastavit extruder do požadované polohy. Zbylé 3 stupně volnosti otočného stolu a posuvných kloubů můžeme využít k optimalizaci.

K hledání optimálních parametrů jsem opět využil nástroj Ceres-Solver. Pro řešení inverzní kinematické úlohy jsem použil integrované řešení MoveItu, KDL Kinematics Plugin. Z MoveItu poté stačilo zavolat funkci *setFromIK()*. Argumenty metody tvoří: výchozí stav robota v kloubových souřadnicích (aby nedocházelo ke změnám konfigurace robota), požadovaná poloha extruderu a čas poskytnutý k řešení úlohy.[33]



Obrázek 5.7: Trajektorie naplánovaná pomocí optimalizace 3 parametrů a inverzní kinematické úlohy

Na horním grafu obrázku 5.7 jsou znázorněné hodnoty optimalizačních kritérií naplánované trajektorie. Váhy vzdáleností v posunu a rotaci jsem zvolil 0, protože s využitím řešení inverzní kinematiky nedochází k odchylkám polohy extruderu od požadované pozice. Pro kritéria manipulovatelnosti jsem nastavil váhy 1 a pro kolizní vzdálenost 10 na metr.

Naplánovaná trajektorie pomocí inverzní kinematické úlohy se nejblíže přiblížila požadavkům pro pohyb robota při svařování plastů. Jediný problém, který jsem nedokázal vyřešit byl, že při pohybu robota docházelo mezi některými dvojicemi poloh k příliš rychlé změně kloubových souřadnic. Následkem toho robot při pohybu měnil konfiguraci. Tyto rychlé změny kloubů jsou znázorněny ve spodním grafu na obrázku 5.7, kde jsou zobrazené hodnoty kloubových souřadnic v průběhu naplánované trajektorie. Tuto problematiku by mělo být možně vyřešit pomocí vylepšení řešení inverzní kinematické úlohy, u kterého bude kladen větší důraz na zachování jediné konfigurace.

Kapitola 6

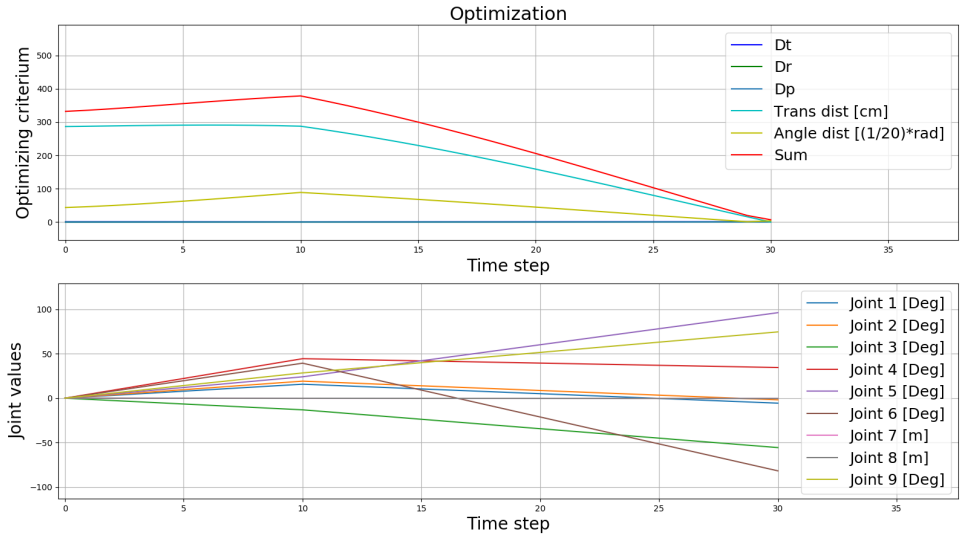
Experimentální část

Následující kapitola je věnována experimentům, na základě kterých jsem otestoval svoji implementaci. Pro experimenty jsem vyzkoušel naplánovat trajektorii pro dva základní sváry, které bude robotická buňka v budoucnu vykonávat.

6.1 Nájezd ke svaru

První část experimentu bylo otestování nájezdu robota k prvnímu svaru. K plánování trajektorie jsem využil knihovnu OMPL napojenou na MoveIt. Poslední bod trajektorie je hledán pomocí genetického algoritmu. Po jeho nalezení je spuštěn algoritmus RRTConnect.

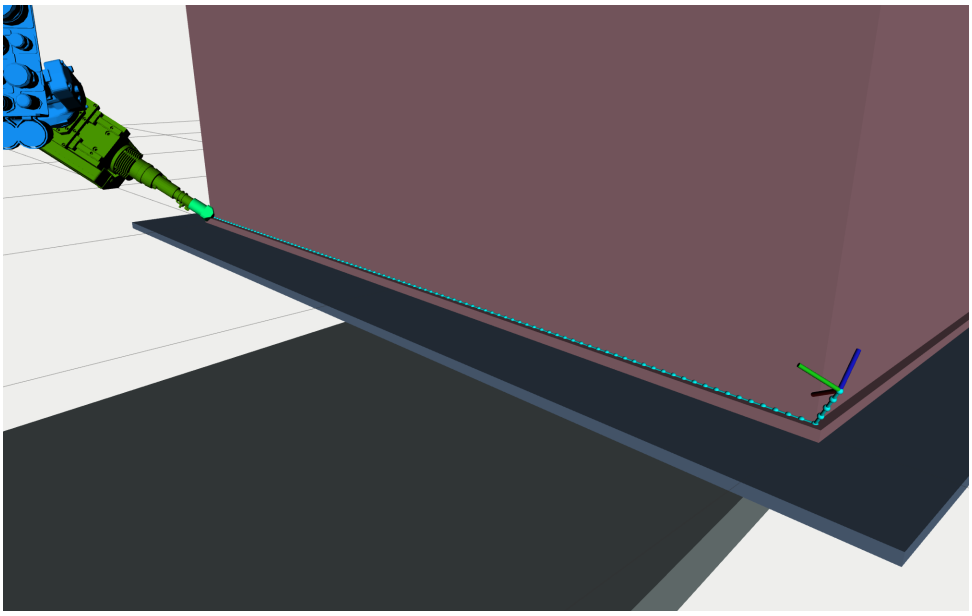
Na obrázku 6.1 jsou zobrazeny dva grafy vyjadřující průběh naplánované trajektorie. Horní graf označuje průběh optimalizačních kritérií během pohybu robota. Z grafu je vidět, že na konci pohybu v časovém kroku 30 jsou jednotlivá optimalizační kritéria minimalizována. Na dolním grafu je znázorněn průběh kloubových souřadnic. Z grafu je zřejmé, že v časovém kroku 10 dochází k ostré změně rychlostí kloubů 1, 4 a 6. Pro řídící systém reálného by bylo vhodné, kdyby byl průběh rychlostí kloubů spojitý. U pohybu znázorněného na obrázku 6.1 není takový požadavek splněn.



Obrázek 6.1: Průběh prvního nájezdu ke svaru

6.2 Svar na dolní hraně nádrže

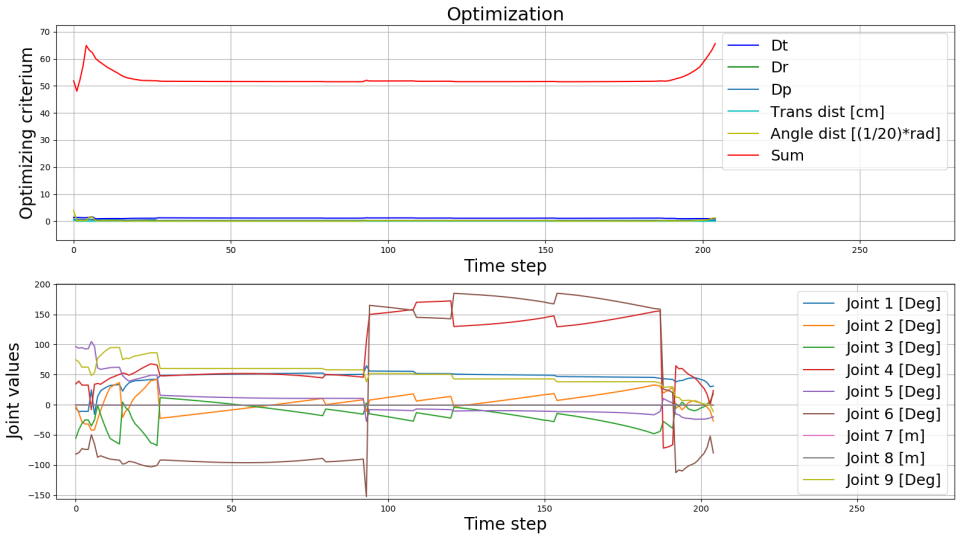
První testovaný svar se nachází z venku nádrže. Pomocí tohoto svaru se spojují podstavná deska a stěny. Robot začíná svůj pohyb kousek od nádrže, kdy se nejdříve po přímce přiblíží ke svaru a následně pak již pokračuje po zadané trajektorii. Svar je zobrazen modrými tečkami a přímkou na obrázku 6.2.



Obrázek 6.2: Dolní svár nádrže

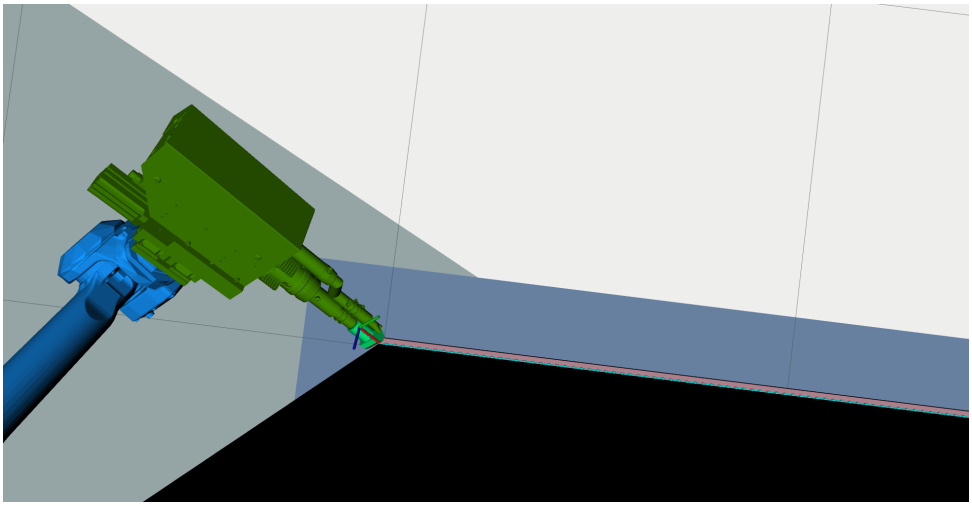
Na obrázku 6.3 jsou zobrazeny grafy průběhu prvního svaru. Na grafu je

vidět, že v průběhu svaru extruder vykonává požadovaný pohyb bez jakýchkoli odchylek. Na spodním grafu je zobrazen průběh kloubových souřadnic robota v průběhu pohybu. Z grafu je zřejmé, že v průběhu pohybu dochází u některých kloubů k rychlým změnám, například v časovém kroku 90. Dochází zde ke změně konfigurace robota a u reálného robota je nutné se těmto změnám vyvarovat.

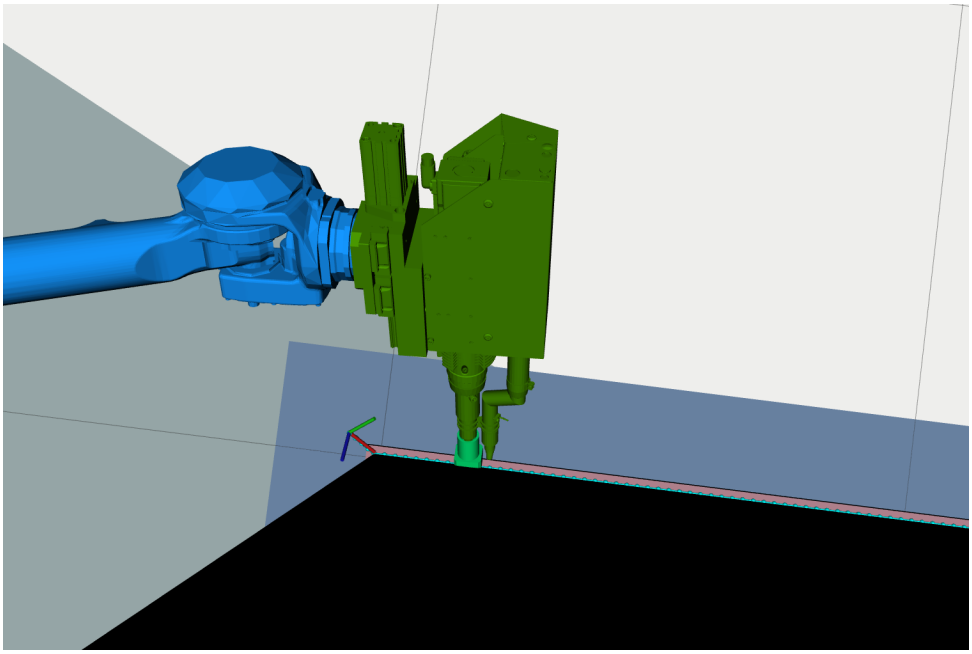


Obrázek 6.3: Svar na dolní hraně nádrže

Extruder se ke svaru přibližuje natočený pod úhlem 45°, viz. obrázek 6.4. V průběhu prvních 20 cm se extruder narovná kolmo ke svaru, viz. obrázek 6.5. Na konci svaru se opět začne natáčet, aby mohl plynule začít svařovat další stěnu.



Obrázek 6.4: Počáteční natočení extruderu



Obrázek 6.5: Poloha extruderu po narovnání

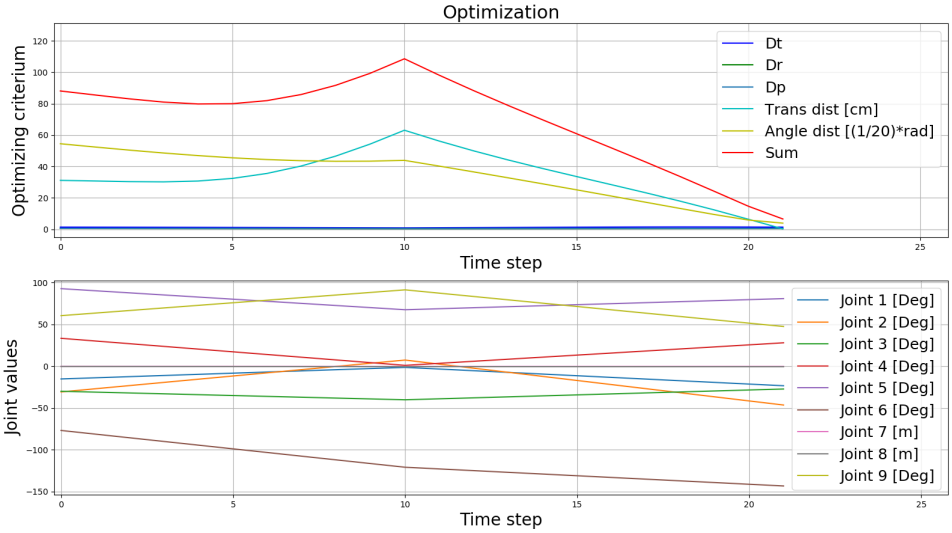
6.3 Přejezd mezi svary

Po provedení dolního svaru následoval přejezd robota ke druhému svaru. Pohyb byl opět naplánován pomocí OMPL a algoritmu RRTConnect. Průběh kloubových souřadnic a optimalizačních kritérií je zobrazen na obrázku 6.6.

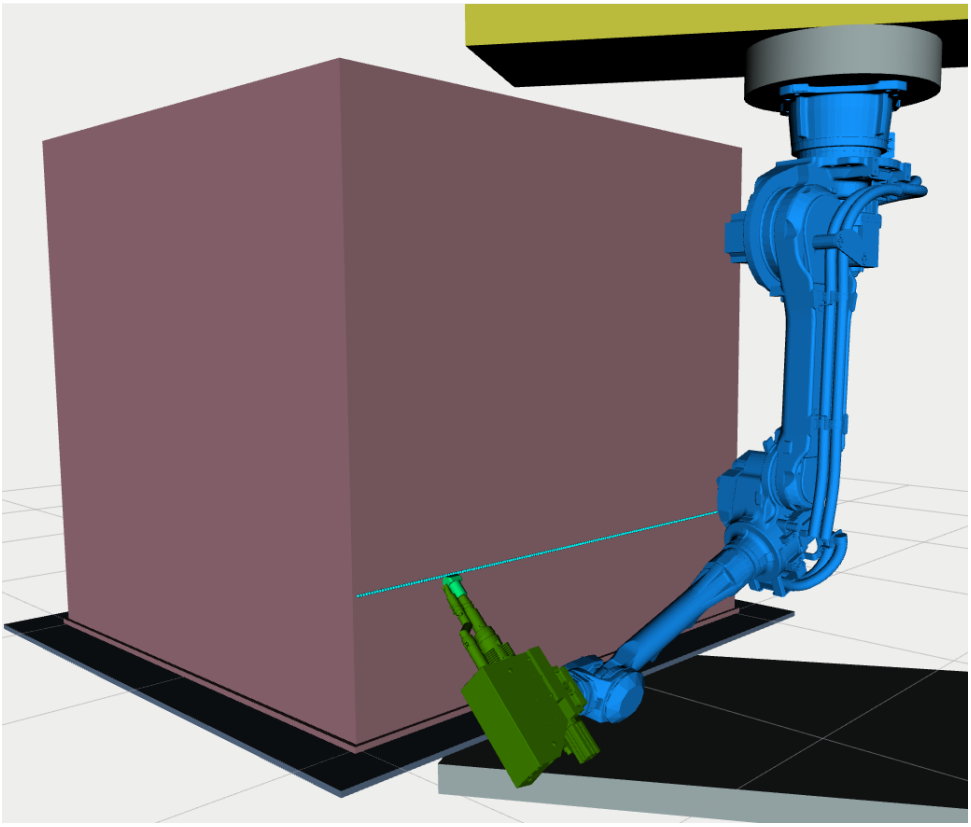
6.4 Sváření žebra

Firma STP plast s.r.o. pomocí přidává na některé nádrže žebra, viz. obrázek 1.1. Účelem těchto žeber je zlepšení mechanických vlastností nádrží. Žebra jsou k nádržím připevněna svařováním. V rámci experimentů jsem jeden takový svar zkusil vykonat.

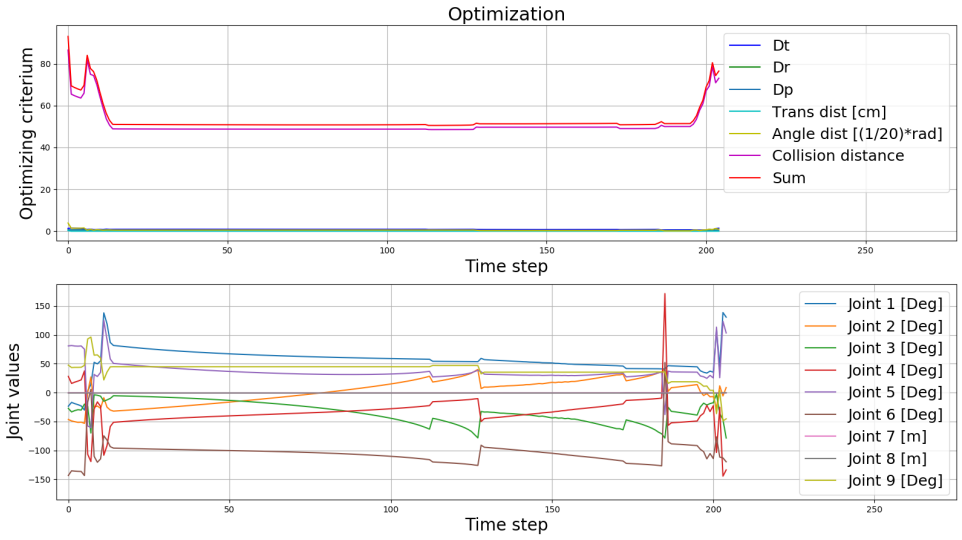
Testovací trajektorii jsem provedl ve výšce 40 cm od spodní strany nádrže. Při tomto svaru je extruder pootočen o 90° oproti svaru na dolní hraně nádrže. Svar včetně polohy extruderu je zobrazen na obrázku 6.7.



Obrázek 6.6: Průběh trajektorie při pohybu mezi svary



Obrázek 6.7: Orientace extruderu při svařování žebra



Obrázek 6.8: Průběh trajektorie při sváření žebra

Na obrázku 6.8 je zobrazen průběh svařovací trajektorie žebra. Z průběhu hodnot optimalizačních kritérií je možné říci, že se poloha extruderu při vykovávání svaru nijak neodchylovala od požadované trajektorie. Z grafu s průběhem kloubových souřadnic plyne, že zde podobně jako v předchozím případě dochází ke změnám konfigurace robota.

6.5 Vyhodnocení experimentů

V rámci experimentů jsem ověřil, že implementované řešení plánování pohybu je možné použít pro nalezení trajektorie robota s dostatečnou přesností. Experimenty ale ukázaly, že implementované řešení není možné bez dalších úprav použít na reálném robotu. V průběhu pohybu robota po naplánované trajektorii dochází k prudkým změnám v rychlostech a zrychleních kloubů.

Problém se spojitostí rychlostí kloubů by mohlo být v budoucí práci možné vyřešit pomocí zavedení dynamických vlastností modelu. Ty vlastnosti je možné specifikovat buď v URDF popisu modelu a nebo definovat přímo v nastavení OMPL plánování.

U naplánovaných svařovacích trajektorií se v průběhu pohybu vyskytují změny v konfiguraci robota. Tento problém by mohl být minimalizován s využitím jiného nástroje k řešení inverzní kinematické úlohy. Použitý nástroj

KDL Kinematics Plugin neumožňuje výběr mezi vícero řešení inverzní kinematické úlohy. Problém s rychlými změnami kloubových souřadnic by současně mohl být eliminován přidáním dalšího optimalizačního kritéria, které by penalizovalo pohyb kloubů na základě potřebné rychlosti.

Kapitola 7

Závěr

V této diplomové práci jsem připravil program pro generování trajektorie pro redundantního robota. V první části jsem nejprve provedl průzkum, které plánovací algoritmy a knihovny se v současnosti využívají k řešení úloh se stejnou tématikou.

V druhé části jsem blíže popsal model robotické svařovací buňky s 9 DoF, na jehož základu bude v budoucnu sestaveno reálné zařízení. V práci jsem dále popsal použité softwarové nástroje, zejména ROS, MoveIt a OMPL. Součást hlavního cíle této práce bylo nalézt optimální trajektorii pro robota s redundantními stupni volnosti. V práci jsou popsána jednotlivá optimalizační kritéria, která jsem využil k hledání ideální trajektorie.

V implementační části jsem nejprve popsal tvorbu vizuálního a kolizního modelu celého pracoviště využitého k simulaci vygenerovaných trajektorií. Dále v práci popisuji jednotlivé kroky, jak jsem při programování postupoval. Nejprve jsem vysvětlil, proč nelze využít téměř připravené řešení MoveItu. Dále jsem vyzkoušel vlastní propojení knihoven OMPL a MoveIt. Po částečně neúspěšném plánování svařovací trajektorie s pomocí knihovny OMPL jsem zvolil jiné řešení. K hledání trajektorie jsem použil optimalizační knihovnu Ceres-Solver a modul připravený k řešení inverzní kinematické úlohy, KDL Kinematics Plugin.

Na závěr práce jsem provedl experimenty, při kterých jsem vyzkoušel naplánování trajektorie pro dva svary a přejezd mezi nimi. Experimenty potvrdily úspěšnost implementace plánování trajektorie, ale ukázaly nedostatky, kvůli kterým není možné bez další práce plánování použít na reálných robotech.

Další práce

- Vylepšení nastavení parametrů genetického algoritmu pro efektivnější vyhledávání cílového stavu
- Vylepšení plánování pomocí inverzní kinematické úlohy tak, aby nedocházelo k skokovým změnám kloubových souřadnic při plynulém pohybu
- Optimalizace výpočetní náročnosti aplikace pro reálné využití

Příloha A

Literatura

[1] STP plast s.r.o. [online]. Stráž pod Ralskem: STP plast. c2020 [cit. 2021-5-1]. Dostupné z:. http://stpplast.cz/.

[2] LEISTER. WELDPLAST 200-i / 600-i. Švýcarsko, c2019, 4 s.

[3] MOTOMAN GP88 [online]. In: . Allershausen (Německo): Yaskawa Europe. c2020, s. 2 [cit. 2021-5-10]. Dostupné z:. https://www.yaskawa.eu.com/Global%20Assets/Downloads/Brochures_Catalogues/Robotics/MOTOMAN_Robots/GP-Series/Flyer_Robot_GP88_E_06.2020.pdf.

[4] Stefano Chiaverini, Giuseppe Oriolo, and Anthony Maciejewski. Redundant robots. pages 221–242, 01 2016.

[5] Lars Larsen, Jonghwa Kim, Michael Kupke, and Alfons Schuster. Automatic path planning of industrial robots comparing sampling-based and computational intelligence methods. *Procedia Manufacturing*, 11:241–248, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[6] Planners Available in MoveIt. MoveIt [online]. [cit. 2021-5-21]. Dostupné z:. https://moveit.ros.org/documentation/planners/.

[7] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics Automation Magazine*, 19(4):72–82, 2012.

[8] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[9] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[10] Manuel Beschi, Stefano Mutti, Giorgio Nicola, Marco Faroni, Paolo Magnoni, Enrico Villagrossi, and Nicola Pedrocchi. Optimal robot motion planning of redundant robots in machining and additive manufacturing applications. *Electronics*, 8(12), 2019.

[11] E.J. Solteiro Pires, P. B. de Moura Oliveira, and J.A. Tenreiro Machado. Manipulator trajectory planning using a moea. 7(3):659–667, June 2007.

[12] A Harish, Chandrashekhar A, and Satish Garlanka. Trajectory planning of a robot manipulator using genetic algorithm. 11 2018.

[13] Ibraheem Bahaa, Bahaa Kazem, Ali Mahdi, and Ali Talib. Motion planning for a robot arm by using genetic algorithm. *Jordan Journal of Mechanical and Industrial Engineering*, 3, 10 0002.

[14] Yu SK. Lin CJ, Chen CS. A gpu-based evolution algorithm for motion planning of a redundant robot. pages 489–494, 2017.

[15] AUSPERGER, Aleš. Technologie zpracování plastů. Svitavy: Code Creator, c2016. ISBN 978-80-88058-77-9.

[16] Technická data pro IRB 4600. ABB [online]. ABB Asea Brown Boveri Zurich. [cit. 2021-5-6]. Dostupné z:. `https://new.abb.com/products/robotics/cs/prumyslove-roboty/irb-4600/irb-4600-technicka-data`.

[17] [cit. 2019-05-12]. Dostupné z: The 3-Clause BSD License [online]. c2019. `https://opensource.org/licenses/BSD-3-Clause`.

[18] urdf [online]. c2019, [cit. 2021-05-7]. Dostupné z:. `http://wiki.ros.org/urdf`.

[19] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. `https://www.ros.org`.

[20] Ioan A. Sucan and Sachin Chitta, MoveIt, [Online] Dostupné z: moveit.ros.org.

[21] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[22] W Khalil and E Dombre. Chapter 5 - direct kinematic model of serial robots. In W Khalil and E Dombre, editors, *Modeling, Identification and Control of Robots*, pages 85–115. Butterworth-Heinemann, Oxford, 2002.

[23] Ricardo Campa and Karla Camarillo-Gómez. *Unit Quaternions: A Mathematical Tool for Modeling, Path Planning and Control of Robot Manipulators*. 09 2008.

[24] Eric W. Quaternion. From MathWorld-A Wolfram Web Resource. Weisstein. `https://mathworld.wolfram.com/Quaternion.html`.

[25] Planning Scene. MoveIt Motion Planning Framework [online]. PickNik Robotics [cit. 2021-5-10]. Dostupné z:. `https://ros-planning.github.io/moveit_tutorials/doc/planning_scene/planning_scene_tutorial.html`.

[26] ROS Melodic Morenia [online]. c2018, [cit. 2021-05-20]. Dostupné z:. `http://wiki.ros.org/melodic`.

[27] MoveIt Setup Assistant. MoveIt Motion Planning Framework [online]. PickNik Robotics [cit. 2021-5-15]. Dostupné z:. `http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html`.

[28] Move Group C++ Interface. MoveIt Motion Planning Framework [online]. PickNik Robotics [cit. 2021-5-10]. Dostupné z:. `https://ros-planning.github.io/moveit_tutorials/doc/move_group_interface/move_group_interface_tutorial.htmlf`.

[29] OMPL Planner. MoveIt Motion Planning Framework [online]. PickNik Robotics [cit. 2021-5-10]. Dostupné z:. `http://docs.ros.org/en/kinetic/api/moveit_tutorials/html/doc/ompl_interface/ompl_interface_tutorial.html`.

[30] Representing Goals in OMPL. The Open Motion Planning Library [online]. Rice University: Kavraki Lab c2019 [cit. 2021-5-10]. Dostupné z:. `http://ompl.kavrakilab.org/goalRepresentation.html`.

[31] Optimal Planning Tutorial. The Open Motion Planning Library [online]. Rice University: Kavraki Lab c2019 [cit. 2021-5-10]. Dostupné z:. `http://ompl.kavrakilab.org/optimalPlanningTutorial.html`.

[32] Modeling Non linear Least Squares. Ceres Solver [online]. c2020 [cit. 2021-5-21]. Dostupné z:. `http://ceres-solver.org/nnls_modeling.html`.

[33] Robot Model and Robot State. MoveIt Motion Planning Framework [online]. PickNik Robotics [cit. 2021-5-10]. Dostupné z:. `https://ros-planning.github.io/moveit_tutorials/doc/robot_model_and_robot_state/robot_model_and_robot_state_tutorial.html`.

Příloha B

Obsah přiloženého CD

- thesis.pdf - tato práce
- Složka se zdrojovými kódy implementované aplikace