VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Proces řízení optimální trajektorie na robotickém pracovišti pomocí digitálního dvojčete

Optimal trajectory control process for robotic workstation using a digital twin

Bc. Daniel Seidel

Diplomová práce
Vedoucí práce: Ing.Zdeněk Macháček, Ph.D
Ostrava 2021/2022

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Daniel Seidel**

Studijní program: N0714A150001 Řídicí a informační systémy

Téma: Proces řízení optimální trajektorie na robotickém pracovišti pomocí digitálního dvojčete
Optimal trajectory control process for robotic workstation using a digital twin

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Rozbor problematiky metod simulace procesu řízení robotizovaných zařízení.
2. Rozbor problematiky optimalizace trajektorií robotizovaných pracovišť.
3. Návrh metodiky řízení optimální trajektorie.
4. Implementace digitálního dvojčete s prvky dynamiky a kinematiky 3D modelu robotizovaného pracoviště.
5. Optimalizace trajektorie dle zvolených kritérií a metod pomocí digitálního dvojčete.
6. Analýza s vyhodnocením shody optimálně realizované trajektorie na reálném pracovišti a digitálním dvojčeti.
7. Zhodnocení dosažených výsledků závěrečné práce.

Seznam doporučené odborné literatury:

[1] KOUBAA, Anis, Hachemi BENNACEUR, Imen CHAARI, et al. Robot Path Planning and Cooperation: Foundations, Algorithms and Experimentations. Springer, Cham. Springer International Publishing, 2018. ISBN 978-3-319-77042-0.

[2] THRUN, Sebastian, Wolfram BURGARD a Dieter FOX. Probabilistic Robotics: Intelligent Robotics and Autonomous Agents. The MIT Press, 2005. ISBN 978-0-262-20162-9.

[3] ODREY, Nicholas, Mitchell WEISS, Mikell GROOVER, Roger NAGEL a Ashish DUTTA. Industrial Robotics -Technology ,Programming and Applications (SIE). 2 edition. McGraw Hill Education, 2017. ISBN 1259006212.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

prof. Ing. Jiří Koziorek, Ph.D.	prof. Ing. Jan Platoš, Ph.D.
vedoucí katedry	*děkan fakulty*

Abstrakt

Práce je o optimalizaci trajektorie v simulačním prostředí třetí strany se zaměřením na samostatnou úpravu a vytvoření optimální trajektorie. Výsledná trajektorie by měla být následně importována do simulačního prostředí výrobce daného robota na kterém je práce prováděna a být schopný komunikovat s prostředím třetí strany tak aby obě robotická ramena vykonávaly stejný pohyb v co nejkratším časovém rozptylu od sebe. Výsledný ověřený program je následně importovatelný do reálného pracoviště, jež bude následně spuštěno s optimalizovaným programem a porovnat shodu se simulačním očekáváním.

Klíčová slova

Robotické rameno; Visual Components; Teachpendant; Robotstudio; Digitální dvojče

Abstract

The thesis is about optimizing the trajectory in a third-party simulation environment with a focus on self-adjustment and creating an optimal trajectory. The resulting trajectory should then be imported into the simulation environment of the robot manufacturer on which the work is performed and be able to communicate with the third-party environment so that both robotic arms perform the same movement in the shortest possible time dispersion from each other. The resulting verified program is then importable into a real workplace, which will then be launched with the optimized program and compare compliance with simulation expectations.

Key words

Robotic arm; Visual Components; Teachpendant; Robotstudio; Digital twin

Poděkování

Rád bych poděkoval Ing. Zdeněk Macháček Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce, dále také kolegovi ze společnosti Elvac a.s Ing. Karel Wija.

Obsah

Seznam zkratek

2D	**Dvou dimenzionální**
3D	**Tří dimenzionální**
Add-on	**Doplněk**
CAD	**Computer aided design**
CIRC	**Circular motion**
DoF	**Degrees of freedom**
LIN	**Linear motion**
OLP	**Off line programování**
PTP	**Point to point motion**
PnP	**Plug ‘n play**
SCARA	**Selective Compliance Assembly Robot Arm**
TCP	**Tool center point**
VC	**Visual compoennts**

Seznam obrázků

Seznam tabulek

Úvod

Cílem této práce je zabývat se porozumění pohybového principu robotických ramen využívaných ve výrobě, potravinářství, zdravotnictví či výzkumu. Při počátku jsou zkoumány pohybové akce robotických ramen, a to jak vytváření pohybových akcí, specifikace požadavků pro nejlepší a nejvýhodnější typy pohybů tak i specifikace pro volby těchto pohybů. Ozvláštnění celého procesu zajišťoval simulační proces, který je co nejvěrnější a nejidentičtější reálnému sestavení tak aby nemohlo dojít k destruktivním pokusům při počátečním testování průchozích cest prostředím.

Při výběru simulačního softwaru je brán ohled hned na několik kvalit, následný výběr tedy dopadl tak aby se v softwaru dalo pracovat pohodlně a také se v něm daly dělat úpravy které by zajišťovaly co nejvěrohodnější a nejpřesnější srovnání simulovaného a reálného prostředí. Prvotní plány tedy souvisí s výběrem vhodného prostředí, základní práce v tomto prostředí a modelování prostředí které bude použito reálně. Po tomto kroku se soustředím na ovladatelnost, rychlost, trajektorie, stabilitu a orientaci simulovaného pracoviště jakožto vzor, ze kterého pak je možné čerpat data pro vývoj reálné aplikace. Jsou zde také zahrnuty různé testované variace jak robotické orientace, tak typy velikostí tras, rychlosti průjezdů mezi jednotlivými body a také typy pohybů.

Díky takovémuto vývoji je možno vytvořit v reálném pracovišti program, jenž je už dostatečně připraven k využití bez větších obav o kolizní stavy s prostředím, přílišné zatížení servopohonů ať už způsobeno přebytečnou rychlostí, nepřiměřenou a nevhodnou orientací, nebo nedosažitelností požadovaných pozic.

V první kapitole je rozebrán popis problematiky metod virtualizace procesu řízení, ve kterém se soustředím na jednotlivé výběry virtualizačních softwarů, dále jejich využití v oblasti automatizace a zhodnocení výhod a vad oproti ostatním vybraným softwarům. Také se zde zabývám kompletními výhodami a nedostatky virtualizace jakožto celkového oboru ve firmě, kdy je tento proces výhodný a kdy zase tento proces může být zbytečný.

Ve druhé kapitole pojednávám o problematice optimalizování procesu řízení, do této oblasti spadá jak správná volba robotické operace, sběr užitečných dat, optimalizace řízení trajektorií, optimalizace polohového nastavení a orientace robotického ramene.

V třetí kapitole, ve které se zabývám návrhem metodiky optimalizace procesu řízení se snažím o popis a shrnutí vývoje co nejkratší trajektorie, které budou co nejméně náročné pro servopohony a s co nejpřesnější opakovatelností a znuvupoužitelností.

Ve čtvrté kapitole připravuji a upravuji 3D model k pracovnímu postupu, to se týká jak rozdělení modelu na jednotlivé části, tak nastavení jednotlivých pohyblivých komponent ke zprovoznění a ovladatelnosti, jenž se nadále bude využívat pro kontrolu pohybu.

V páté kapitole se zabývám problematikou řízení trajektorií, pozic, rychlostí a orientací na základě rozboru tématiky v druhé kapitole. Díky stanoveným kritériím a metodám tedy v tomto bodě bude sestaven co nejefektivnější řídící proces jež bude aplikován jak ve virtuálním prostředí, tak v reálném.

Šestá kapitola obsahuje analýzu s vyhodnocením shody optimálního řízení reálného pracoviště a virtuálního modelu. Tady zde budou porovnávány výsledky rozdílů mezi průběhy v jednotlivých operacích a zaznamenaných dat při průběhu splňování sestavení nejvalidnějšího řízeného procesu.

Sedmá kapitola je poslední část této diplomové práce a zabývám se v ní zhodnocením kompletních výsledků jež jsou zaznamenány v průběhu vytváření této diplomové práce a následně interpretovány, tato kapitola se také odkazuje na poznatky, chyby a zjištění ze všech předchozích kapitol.

1 Rozbor problematiky metod simulace procesu řízení robotizovaných zařízení

Název simulace je používáno hned v několika odvětvích novodobých technologií, původní význam definoval simulace jako napodobení reálného objektu, stavu či procesu, a to s co největší přesností. Mezi odvětví využívající simulace ve svůj prospěch patří například simulace pro výcvik a výuku, zdravotnické simulátory nebo i simulace digitálního životního cyklu[1].

- Zvýšená přesnost: Simulační modely umožňují analyzovat systémy a hledat řešení kde tradiční metody selžou. Je to z velké části proto, že simulační model může brát v úvahu více komplexní data, vzájemnou závislost a chování systému v průběhu času.
- Jednoduchost použití: Jakmile vyberete přiměřenou úroveň abstrakce, vývoj simulace modelu je přímočařejší než analytické modelování. Obvykle to vyžaduje méně intelektuálních úsilí, je škálovatelné a modulární.
- Kooperace: Struktura simulačního modelu přirozeně odráží strukturu skutečného systému. Simulace modelu jsou vyvíjeny převážně s využitím vizuálních jazyků, je snadné sdělit vnitřní funkčnost ostatním.
- Měřitelnost: V simulačním modelu je možné změřit jakoukoliv hodnotu, sledovat jakoukoliv entitu. Měření a statistickou analýzy je možné přidat kdykoliv.
- Vizualizace: Schopnost přehrát a animovat systémové chování je jednou z nejlepších výhod simulace. Animace je často využívána nejen jako demo verze ale také pro ověřování a ladění.
- Přesvědčování: Simulační modely jsou mnohem více přesvědčivé nežli tabulky. Pokud zákazníkovi přineseme a spustíme simulaci, určitě budeme mít výhodu oproti těm kteří přinesou pouze čísla a tabulky
- popis typů robotů s ohledem na jejich možnosti pohybů a tvoření samotné trajektorie

1.1 Využití simulace

Velmi dobrým způsobem, jak učit dovednosti a postupy, které jsou nebezpečné nebo nákladné je právě využití simulací. Vzdělávané osoby mohou cvičit v bezpečném prostředí a získat důvěru bez obav z podstatných důsledků.

Výhody dle studií

Ve výzkumu bylo zjištěno že zaměstnanci se nejvíce, a to ze 70 %, učí ze zkušeností, 20 % ze sociálních interakcí a 10 % z tradičních zdrojů učení. Simulace jsou nástroje, které vytvářejí uměle generované reálné pracovní prostředí právě proto, aby se s ním mohli zaměstnanci ztotožnit a seznámit. Tyto simulace zajišťují zážitkové učení bez možnosti reálných škod, jež by mohly při seznamování s reálným pracovištěm vyskytnout. Tyto znalosti a zkušenosti s prostředím je často možné vytvořit ještě před sestavením reálného pracoviště, tudíž budeme mít proškolené pracovníky kteří budou schopni aplikovat koncepty naučené v realistickém, kontrolovaném prostředí hned jak to bude potřebné[2].

Simulace řídí změnu chování

Řízení změny v chování u lidí je velký výkon, protože zpochybňuje, jak se tělo pravidelně chová. Aby bylo možné změnit chování vzdělávaných osob a toto chování uchovat na stabilní bázi, musíte je přimět opakovaně praktikovat nové chování. Simulace učení jsou nákladově efektivní nástroje pro standardizaci obsahu, jehož cílem je změnit chování vzdělávaných osob.

Tyto vzdělávané osoby mohou opakovaně zkoušet své dovednosti v kontrolovaném prostředí ve kterém mohou udělat chyby bez jakýchkoliv následků a následně se z těchto chyb poučit, testovat nové metody řešení stejného problému a nalezení efektivnějších metod pro jejich řešení, než byly naučeni. Jakmile jsou si jisti tím, co jsou schopni udělat, jsou připraveni na změnu chování [2].

Simulace jsou ideální způsob, jak trénovat větší kolektiv

Pro společnosti, které potřebují vzdělat větší množství osob může být běžné zaučení nákladné v celé organizaci. Problém nemusí obsahovat jen velké množství školených osob ale také s jejích geografickým rozptylem. Tímto způsobem by byli nuceni utrácet nemalé částky za místo konání, cestování a ubytování, jen aby všechny tyto osoby zaučili na jednom místě.

Oproti tomu simulovaná výuka může být konána za pomoci digitálních platforem. Pokud budou mít vzdělávané osoby přistup k internetu, pak mohou být zaškoleny kdekoliv. Kromě toho, vzdělávané osoby nového věku poreferují být vzdělávány z mobilních zařízení, tím se zaměstnanci mohou učit na svých pracovištích, pokud to jejich rozvrh dovolí místo toho, aby si museli vyhradit speciální čas na školení ve třídě[2].

Simulace jsou bez rizika

Pracovní simulace umožnují vzdělávaným osobám provádět dané úkony stejným principem jako když je budou dělat v reálném prostředí. Výhodou je, že pokud při trénování udělají špatný pohyb nebo akci pak zde nejdou žádné reálné následky, které by mohly ovlivnit, poškodit nebo zničit prvky v reálném pracovním prostředí, jelikož se učí v kontrolovaném prostředí.

Zaměstnanci nejsou vybízeni k experimentování na jejich reálných pracovištích, jelikož by mohli ublížit stroji, firmě ale hlavně i jiným zaměstnancům. V simulovaném prostředí ale všechny tyto negativa odpadají a neměli by se bát zkusit nové postupy či své nápady, jak zlepšit jejich pracovní postup, i když selžou tak je jim poskytnuto množství opakování pro změnu jejich rozhodnutí. Zaměstnanci jsou také ihned informováni zpětnou vazbou, tato zpětná vazba pomáhá vzdělávaným osobám získat důvěru při řešení jejich práce v reálném světě[2].

Simulace zlepšují uchovávání znalostí

Zapamatovat si zábavný zážitek je jednodušší. Pokud si užívají to, co dělají rádi pak si tuto zkušenost zapamatují na delší dobu, simulace dělají učení zábavnější a vstřícnější. Vzdělávané osoby si zapamatují naučené informace po delší dobu a také budou schopni si připomenout naučené koncepty které použijí ve své práci[2].

Simulace nabízejí okamžitou zpětnou vazbu

Pracovní simulace často nabízejí okamžitou zpětnou vazbu od instruktorů stejně jako od platformy, na které se učí. Jako výsledek mají možnosti zjistit jejich nedostatky a slabiny na kterých mohou následně zapracovat a vylepšit je. Také je však poukázáno na jejich silné stránky a jsou povzbuzeni, aby na těchto stránkách vydělali. Konstruktivní zpětná vazba by měla být nabízena co nejrychleji aby se zaučované osoby nedopouštěly dělání stejných problémů v budoucnu. Díky zpětné vazbě se také vzdělavatel ujistí že nebudou tyto vzorce spatného chování využívány v budoucnu, namísto toho vstřebají správné vzorce a následně je využijí[2].

1.2 Simulační programy

U simulačních programů se často využívá off-line programování, nejčastěji právě u robotických linek, kde výrobci různých typů robotů, často přední výrobci, mají vlastní software, který obsahuje knihovnu jejich robotů, jejich programovací jazyk a také možnost se k těmto robotům vzdáleně připojit a ovládat je bez osobní přítomnosti. Často se tedy daná sestava prvně vytváří v tomto off-line módu, kde jsou odzkoušeny různé varianty umístění robotů, plánovaných tras nebo práce s plánovanými objekty.

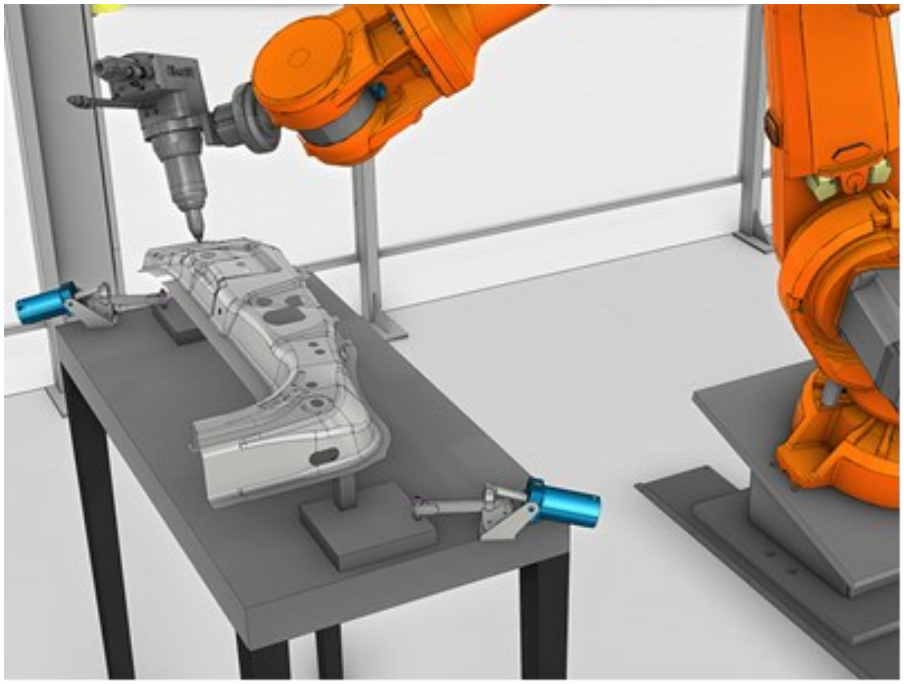
Využívat se však nemusí pouze softwary těchto výrobců, ale také se zde nabízí možnosti pro softwary třetích stran. Ty se využívají z principu často větší knihovny robotických zástupců od různých firem, a proto umožňují porovnávat jaký typ robota a od kterého výrobce je tato volba nejvhodnější.

1.2.1 Off-line softwary třetích stran

Mezi simulační softwary třetích stran pracujících v offline režimu patří Octopuz, RobotDK, Famos,Robotmaster, AUTOMAPPS a Robotics.

Octopuz

Kanadská firma OCTOPUZ Inc. se 15 let specializuje na design, vývoj, implementaci a přizpůsobení softwaru pro offline programování průmyslových robotů. Firma nabízí vývojové prostředí OCTOPUZ 3.0. Software OCTOPUZ umožňuje automatizaci úloh, které jsou pro pracovníky opakující se, zdlouhavé, nebezpečné nebo nevhodné, a to díky snadnému programování a přeprogramování průmyslových robotů. Programování robotů je pro každého výrobce v prostředí stejné až do exportu kódu pro danou značku robota. Možnost programovat aplikace s více roboty. Nabízí tvorbu svařovacích, nástřikových, obráběcích, řezacích a aditivních operací. U svařovacích aplikací software nabízí implementaci mnoha druhů svařovacích postupů, jako jsou více vrstvové svary, bodové svařování, snímání dotyku, sledování svaru a další. OCTOPUZ dokáže automaticky napočítat optimální místa pro snímání dotyku, nebo ručně vytvářet jedno, dvou nebo tří bodové snímání dotykem. Simuluje a analyzuje tloušťku naneseného nástřiku. Podpora výrobců robotických ramen ABB, FANUC, KUKA, YASKAWA a mnoho dalších[3].



Obr. 1 Prostředí Octopuz

Dále OCTOPUZ nabízí možnost výrobcům a integrátorům vytvářet ve virtuálním prostředí přesně své robotické buňky. Jednotlivé komponenty lze vkládat do prostředí z rozsáhlé knihovny. Virtuální prostředí nabízí testování složitých mechanických dílů, optimalizaci operací a manipulací s díly. Nastavit komunikaci mezi vstupy a výstupy jednotlivých komponent[3].

RobotDK

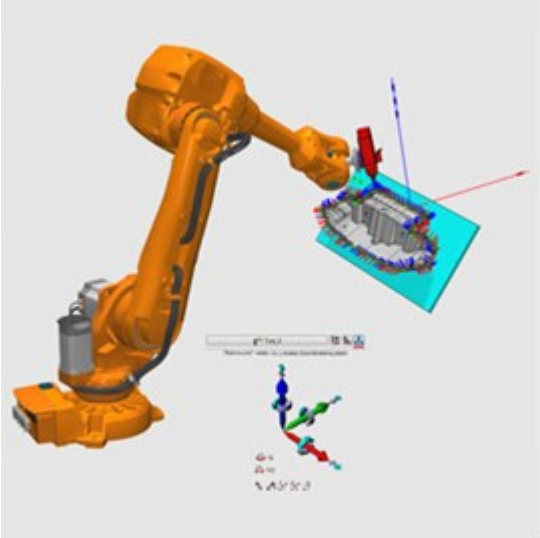
Firma RobotDK nabízí stejně jmenovaný SW, firma vznikla jak spin-off v roce 2015 v Kanadě. RobotDK nabízí rozsáhlou knihovnu průmyslových robotických ramen od 40 různých výrobců (ABB, KUKA, Fanuc, Universal Robots a další). Knihovna podporovaných průmyslových robotických ramen je dostupná online na stránkách výrobce softwaru. RobotDK nabízí konfiguraci až tří externích os, jako jsou lineární kolejnice, otočné talíře. Další funkcí softwaru je možnost modelování a synchronizování dalších os. Podporovaný import 3D modelů vlastních nástrojů, importované formáty STL, STEP a další. Simulování drah robotů nabízí možnost vytvářet obráběcí, pick & place, nástřikové, vrtací, laserové, svařovací, 3D tiskové operace. RoboDK API umožňuje programovat libovolného robota pomocí jednoho z podporovaných programovacích jazyků, jako je C \#, Python nebo C ++. Opakované úkoly můžete také automatizovat pomocí rozhraní RoboDK API. RoboDK podporuje virtuální realitu[4].

Obr. 2 Prostředí RobotDK

RobotDK je nabízen ve čtyřech verzích. Ve všech prodejních verzích je dostupná celá knihovna průmyslových robotických ramen. První verze je zdarma ke stažení se všemi funkcemi po dobu 30 dní. Omezené je generování programu na 50 řádků kódu. Druhá "Education" verze je dodávaná pouze do akademického prostředí. Tato licence nabízí všechny funkce offline programování, robotické obrábění, 3D tisk, neomezené generování programu, přístup ke kontrolérům robotů. Cena této licence je 145 euro. Druhá "Professional" verze nabízí navíc oproti education licenci možnost simulace více robotů najednou, podporu externích os a synchronizaci až 12 os. Cena této licence je 2995 euro. Poslední verze calibration & performance testing, nabízí navíc oproti professional kalibraci a testování výkonu. Cena této licence je na vyžádání[4].

Famos

Ve svém vlastním softwarovém oddělení vyvíjí Carat robotické inovace GmbH FAMOS robotic jako softwarový systém pro offline programování, simulaci a optimalizaci procesů průmyslových robotů. CAD modely lze importovat v různých formátech a snadno je použít pro generování trajektorií. Cesty lze vytvářet například podél souvislých křivek a hran, na plochách a průsečících. Všechny souřadné systémy (nástroje, TCP, základny) přesouvat a otáčet relativně ke každému jinému souřadnicovému systému v projektu. Hodnoty lze měnit postupně nebo je zadat přímo. Orientace se zobrazují automaticky ve formátu příslušného robota. FAMOS podporuje mít ve stejném simulačním projektu použití několik robotů, kteří mohou být od různých výrobců. Počet robotů v projektu je omezen pouze výkonem počítače. Roboty v projektu lze vyměnit za jiné i v běžící simulaci, vyměňovat lze i roboty od jiných výrobců. Program podporuje hlídání kolizí mezi roboty a pracovním prostorem a řízení externích os. Knihovna robotů nabízí okolo 500 modelů a podpora značek robotů ABB, KUKA, FANUC a další. Firma nabízí možnost vytvoření modelu, když se nenachází v knihovně. FAMOS lze využít v aplikacích jako je leštění, broušení, lepení, řezání a další[5].



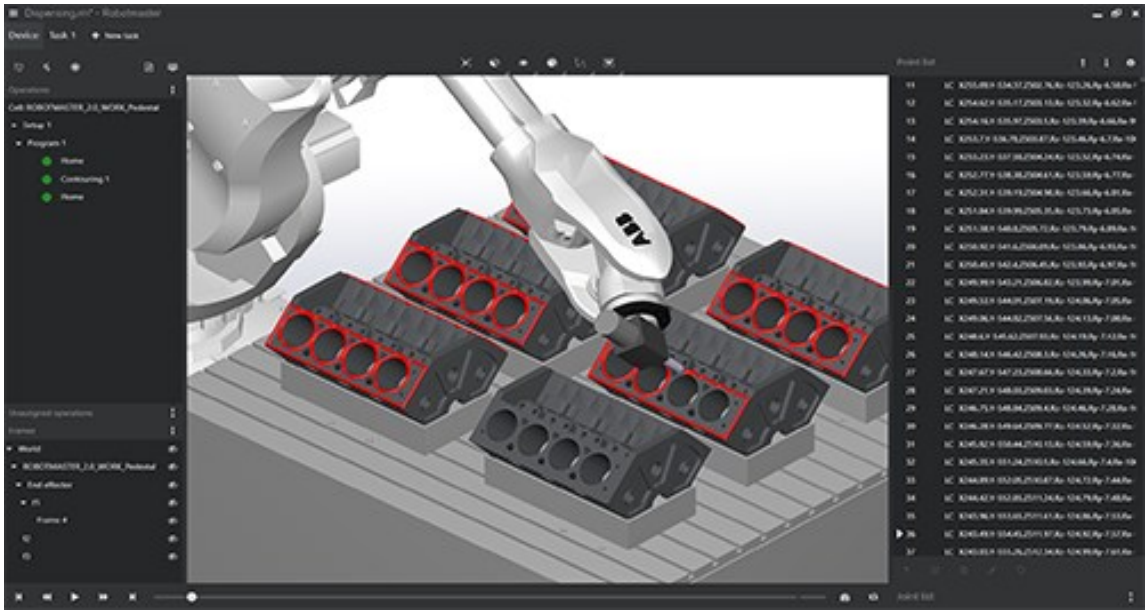
Obr. 3 Prostředí Famos

FAMOS nabízí psaní skriptů, které umožňují vytvářet vlastní funkce. Pro programování skriptů se využívá jazyk pascal. Pomocí skriptů lze vytvářet automatické úkoly, které se neustále opakují. Tímto je myšleno tvoření duplicitních cest a generování nových cest. Import a export bodů tras z textového soubor ve vlastním formátu[5].

Robotmaster

Robotmaster vznikl v roce 2001 jako myšlenka, která se chtěla oprostit od klasického programování robotů a využít možnost offline programování. Nyní Robotmaster patří pod společnost Hypertherm. Software poskytuje nástroj pro programování, vizualizaci a optimalizaci robotických procesů. V programu Robotmaster lze parametrizovat rotaci nástroje, náklonu nástroje a polohy kolejnice a rotace. V prostředí lze vytvářet aplikace v oblastech ořezávání, řezání, robotického obrábění, odstraňování otřepů, svařování, leštění, broušení, lakování a další. Integrace parametrů specifických pro proces uživatele je zjednodušena pomocí upravitelných obrazovek, které definují interakci, terminologii a nastavení řízení jedinečné pro aplikaci. Robotmaster podporuje značky robotů jako je ABB, KUKA, FANUC, YASKAWA a mnoho dalších[6].

Robotmaster nabízí při programování průmyslového robota pro různé aplikace (svařování, obrábění, řezání atd.) možnost orientovat nástroj automaticky v závislosti na optimalizaci s minimálním, schopnost zpravovat vstupy a výstupy, simulování a programování více nástrojů pro robota a kalibrační nástroje kompenzující skutečné nastavení. Robotmaster podporuje snadnou tvorbu aditivních výrobních procesů, jako je 3D tisk, laserové tavení kovů, laserové nanášení kovů a další[6].

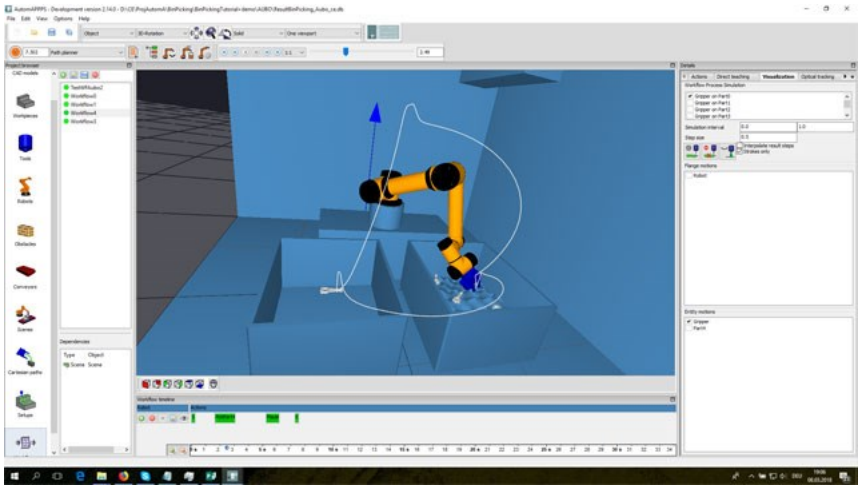


Obr. 4 Prostředí Robotmaster

AUTOMAPPPS

Společnost Convergent Information Technologies se sídlem v Rakousku nabízí na trh produkt AUTOMAPPPS, který je souhrnem nástrojů pro programování průmyslových robotů. Softwary řeší problematiku offline programování, bin pickingu, automatického programování robotů a robotické moduly. Software pro programování robotů zahrnuje rychlé a snadné OLP a automatické generování kódu. Programovací nástroje robotů AUTOMAPPPS a plánování pohybu robota podporuje více než 15 výrobců robotů (ABB, KUKA, Omron, Universal Robots, a další). Podpora formátů JT, STEP, STL a další[7].

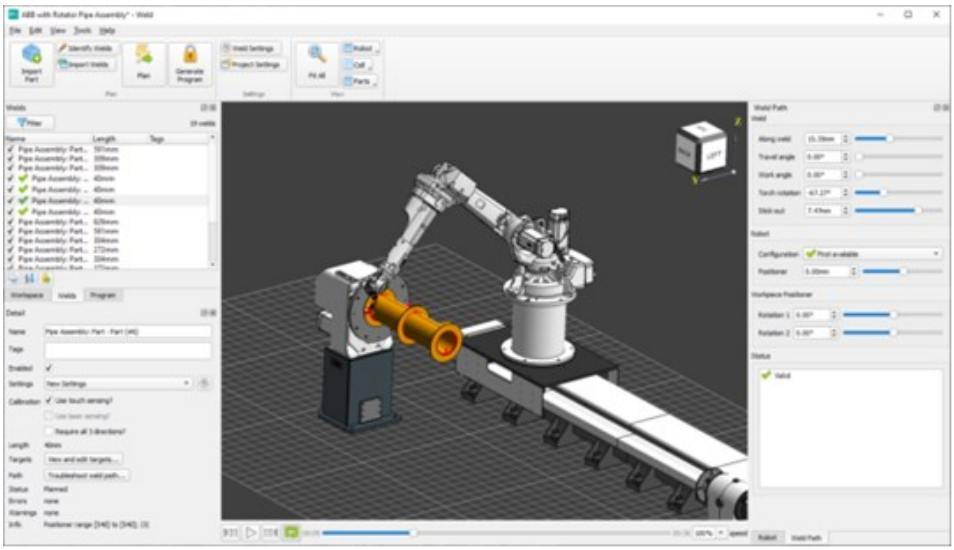
SW obsahuje programování s nízkým a žádným kódem, kromě toho je vytvořeno automatické předcházení kolizím a automatické plánování. AUTOMAPPPS nabízí automatické určování polohy robotického ramene v buňce a ověřování dostatečné velikosti buňky pro vytvořené operace. AUTOMAPPPS má plnou podporu pro svařování, obrábění, pískování, řezání, aditivní výrobu a další. Software podporuje koordinací více robotů se stejným pracovním prostorem. Dále nabízí automatickou optimalizace sekvencí pomocných os[7].



Obr. 5 Prostředí AUTOMAPPPS

Verbotics

Společnost Verbotics řeší nabízí inteligentní software pro generování robotických svařovacích programů přímo z informací CAD modelů. Software pomocí algoritmů z importovaných modelů navrhnout svary, generování není ovlivněno tvarem a složitostí modelu. Navrhnuté svary automaticky parametrizují úhly natočení svařovacího nástroje. Svary lze také importovat z tabulek CSV. Vygenerované svary se dají jednoduše modifikovat. Dráhy svařovacího nástroje jsou automaticky plánovány a optimalizovány. Software podporuje externí osy (portály a rotační) a 7osé roboty. Robotické operace jsou optimalizovány pro nejkratší dobu cyklu. Verbotics Weld nabízí automatickou dotykovou nebo laserovou kalibraci. Software v současné době podporuje offli programování pro robotická ramena od výrobců ABB, FANUC a Yaskawa Motoman's. Software Verbotics Weld je na webových stránkách výrobce nabízen ve zkušební verzi, která je zdarma ke stažení[8].



Obr. 6 Prostředí Verbotics

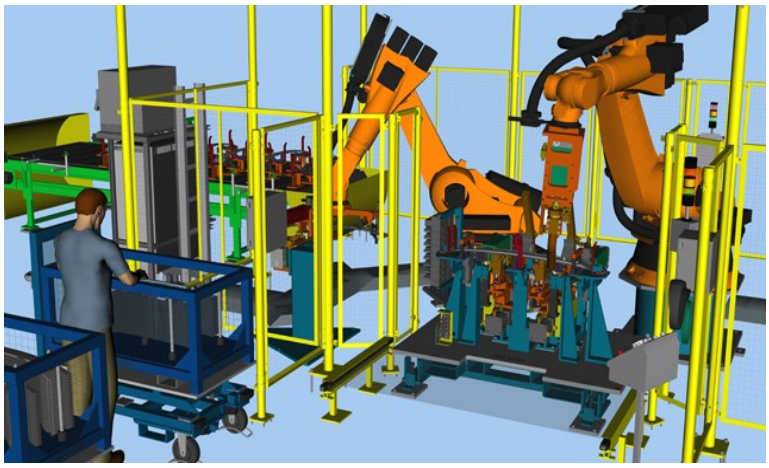
1.2.2 Softwary s nástroji pro digitální dvojče

Mezi softwary obsahující nástroje pro práci s digitálním dvojčetem spadá Tecnomatix, Visual components, Ciros, FASTSUITE, a WinMod.

Tecnomatix

Tecnomatix je seskupení produktů, které se zabývají kompletně digitalizovanou výrobou, které pomáhají digitalizovat výrobu. Se SW Tecnomatix lze dosáhnout synchronizaci mezi produktovým inženýrstvím, výrobním inženýrstvím, výrobou a servisními operacemi[9].

Process Simulate je digitální simulační řešení pro ověření výrobního procesu s využitím robotů v 3D prostředí. Process Simulate je nástroj, který řeší několik úrovní simulace robotů a vývoje pracovních stanic, od stanic s jedním robotem až po kompletní výrobní linky a zóny. Pomocí tohoto nástroje lze zlepšit komunikaci a koordinaci mezi výrobními disciplínami a umožnit tak chytřejší rozhodování. Software Process Simulate nabízí možnost offline programování robotických ramen. Při tvorbě digitálních dvojčat disponuje mnoha nástroji pro jejich přesné zpracování, např. tvorbu dopravníků, tvorbu materiálů potřebných pro výrobu, práci s mnoha druhy senzorů, tvorbu kinematiky, tvoření logických funkcí pro jednotlivé objekty a další. Software Process Simulate nabízí možnost simulování lidských pohybů při vykonávání jejich práce, tyto data lze analyzovat a upravit jejich pracoviště tak, aby byla jejich práce méně náročná a došlo například i k zrychlení výroby. K virtuálnímu zprovoznění nabízí Process Simulate připojení signálu k reálnému PLC s využitím OPC serverů anebo připojení k simulovanému PLC. Nejnovější verze Process Simulate nabízí simulaci AGV, díky které lze detekovat kolize, upevňovat nástroje na konstrukci vozidel, a plánovat efektivnější trasy[10][11].



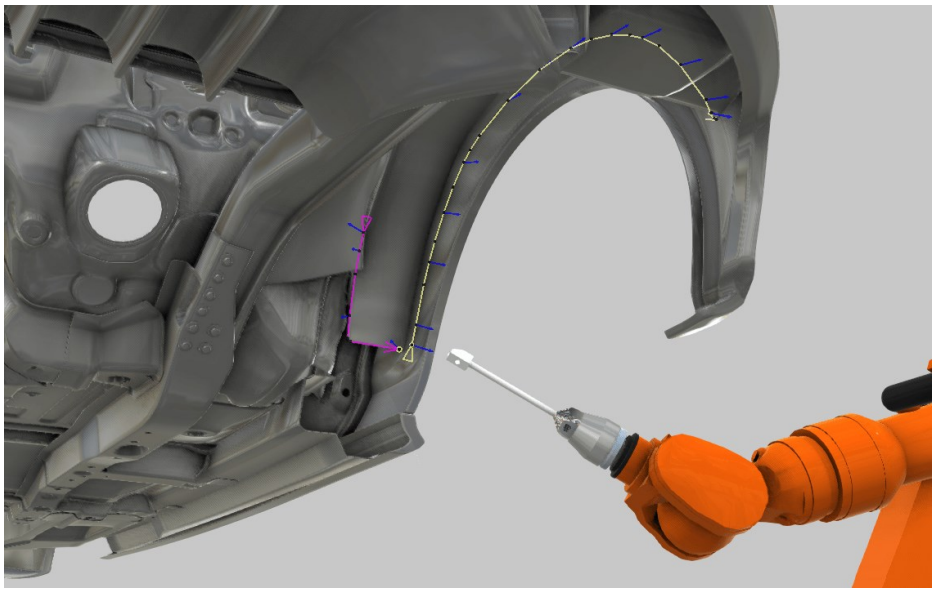
Obr. 7 Prostředí Tecnomatix Process Simulate

Robotická část Process Simulate pokrývá širokou škálu robotických aplikací od Pick\&Place, balení a montáže, svařovací a nástřikové operace. Ke generovaným cestám robota lze přidat podrobné informace, včetně atributů pohybu a procesu, pro vytvoření kompletních programů stažených do skutečného kontroléru. Process Simulate nabízí automatické generování trajektorií na základě vytvořených křivek. Tvorbu svařovacích operací, svařovacích nástrojů a diagnostiku

svarů. Process Simulate dokáže kontrolovat dosah robota, umístit robota do prostoru podle potřeby a lze provádět jednoduché úpravy již vytvořených operací. Samozřejmostí je možnost nahrávat vytvořené programy do robotů, a naopak program z robotu může být přímo načten do Process Simulate, kde může být libovolně upravován. Díky využití virtuálních kontrolérů robotů je dosažena maximální přesnost simulace, přičemž program robotu je pak generován bez nutnosti úprav. Process Simulate nabízí kontroléry od mnoha firem zabývajících se výrobou průmyslových robotů např. KUKA, ABB, Fanuc, Mitsubishi, a mnoho dalších[11].

Visual Components

Visual Components je předním vývojářem softwaru a řešení pro simulaci 3D výroby. Společnost byla založena v roce 1999 týmem odborníků na simulace. Platforma Visual Components byla navržena tak, aby podporovala pokročilé aplikace 3D simulace výroby. Vše pohání open-source fyzikální engine NVIDIA PhysX, v simulačním prostředí jsou objekty a funkce ovlivněny gravitací, vlastnostmi materiálů a kolizí. Software dále nabízí možnost tvorby vlastních API s využitím skriptovacího programovacího jazyka Python. Visual Components je nabízen ve třech verzích a to Essentials, Professional a Premium[12].

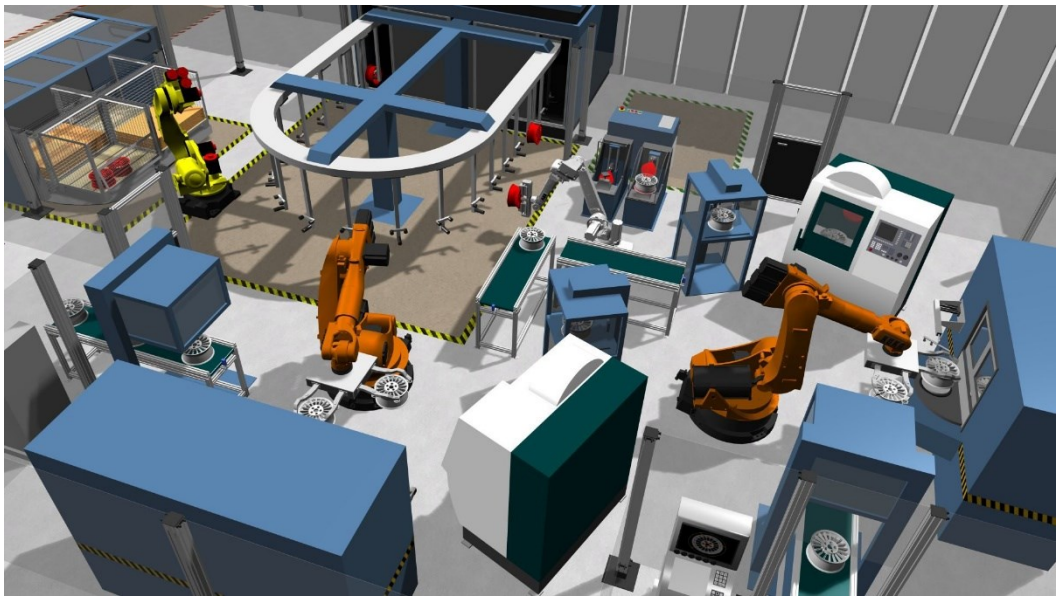


Obr. 8 Prostředí Visual Components

Z pohledu OLP nabízí Visual Components programování robotických úloh ve virtuální prostředí. Nabízí nástroje pro tvorbu křivek, který je schopný automaticky vytvořit kód pro robotickou operaci, dle vytvořené křivky. Visual Components při importu CAD modelů analyzuje model a vytvoří strukturovaná data povrchů. Import dat z katalogu, který obsahuje mnoho předem vytvořených komponent. Umožňuje rychlý návrh a tvorby robotických buněk. Pro výrobce robotických ramen Universal Robot je vyvinut speciální plugin pro připojení ke kontroléru a lze spouštět simulovaného robota s realistickými pohyby. Podobný plugin je vytvořen i pro roboty Stäubli, ve kterém lze vytvářet a upravovat pozice robotických programů a ověřit činnost programů a doby cyklů[13].

Ciros

CIROS Studio je software pro tvorbu digitálních dvojčat v 3D prostředí. V programu CIROS Studio uživatelé modelují rozvržení a procesy, simulují robotické pracovní buňky a automatizované výrobní závody a vizualizují složité sekvence. CIROS Studio spojuje plánování, design, elektrických rozvodů, vývoj procesů, uvádění zařízení do provozu, prodej a marketing. CIROS Studio nabízí plánování a optimalizaci času cyklů, automatický výpočet trajektorií pro zpracování povrchů obrobků, a i následné ruční zpracování. Při virtuálním uvedení do provozu lze využít skutečné PLC. CIROS Studio se například hodí do odvětví automobilového, automatizace, výroby strojů, strojírenství a obrábění. CIROS Studio nabízí k základní aplikaci SW balíčky, které rozšiřují základní SW o problematiku, kterou balíček obsahuje, např. robotika, import modelů atd[14].



Obr. 9 Prostředí Ciros

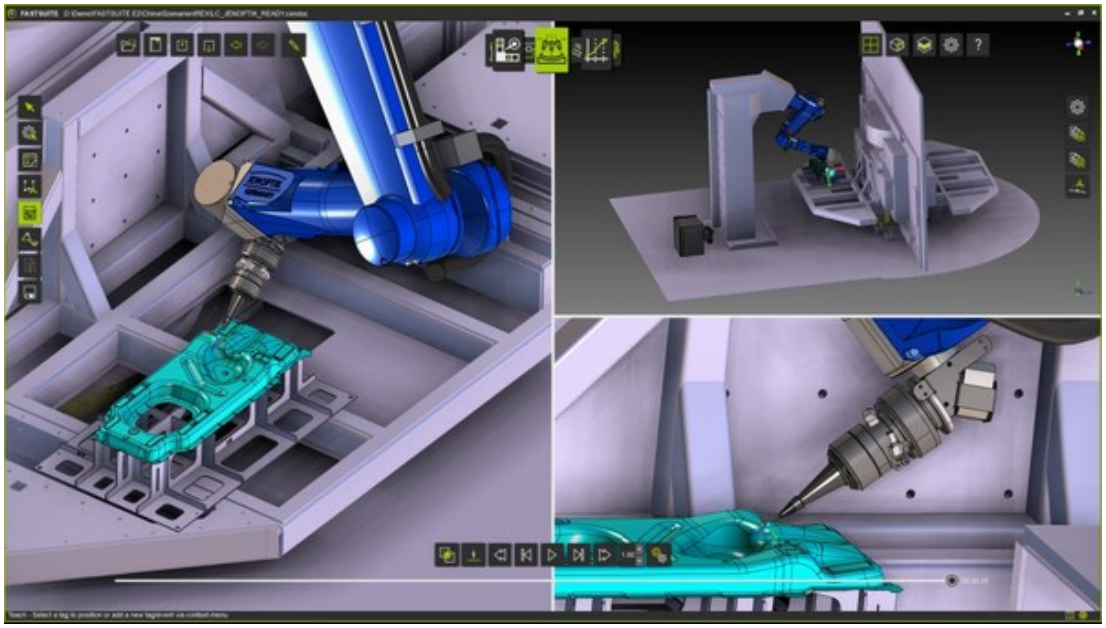
CIROS Studio nabízí v problematice offline programování rozsáhlou knihovnu průmyslových robotů od 17 výrobců. Podporuje spolupráci více kooperaci průmyslových robotů. Dále program nabízí snadné modelování uživatelsky definované kinematiky. Balíček robotiky nabízí virtuální kontrolér robota, průmysloví roboti různých výrobců, virtuální učení pohybů, správu TCP bodu, vizualizaci pracovního prostoru, analýzu dosažitelnosti v závislosti na kinematice, rotační a lineární přídavné osy. Dále nabízí detekci kolizí pro libovolné prvky (roboty, chapadla, uchycené prvky v chapadlech a další). Kolize lze zapisovat do protokolu, každá kolize generuje zprávu s časovým razítkem pro pozdější analýzu. Další robotické balíčky se zaměřují na výrobce

průmyslových robotických ramen, např. ABB s programovacím jazykem RAPID a knihovou modelů robotů, KUKA s programovacím jazykem KRL a knihovnou modelů robotů a další[14].

Další balíčky se zaměřují na mechanismy, import a práci s modely, uvádění do provozu, tvorba videí, virtuální osoby, virtuální reality a mnoho dalších[14].

FASTSUITE

FASTSUITE OLP je založeno na softwarové architektuře a konceptu, který umožňuje uživatelům snadno programovat roboty nebo obráběcí stroje s využitím procesní geometrie definované na obrobcích a dostupných technologických balíčcích. FASTSUITE nabízí dva produkty FASTSUITE pro V5, který se integruje do CATIA/DELMMA V5 (DELMIA V5 je software, který se specializuje na digitální výrobu a simulaci výroby). Druhý SW, který firma nabízí FASTSITE Edition 2 je nezávislá platforma, která slouží k tvorbě digitální továrny[15].



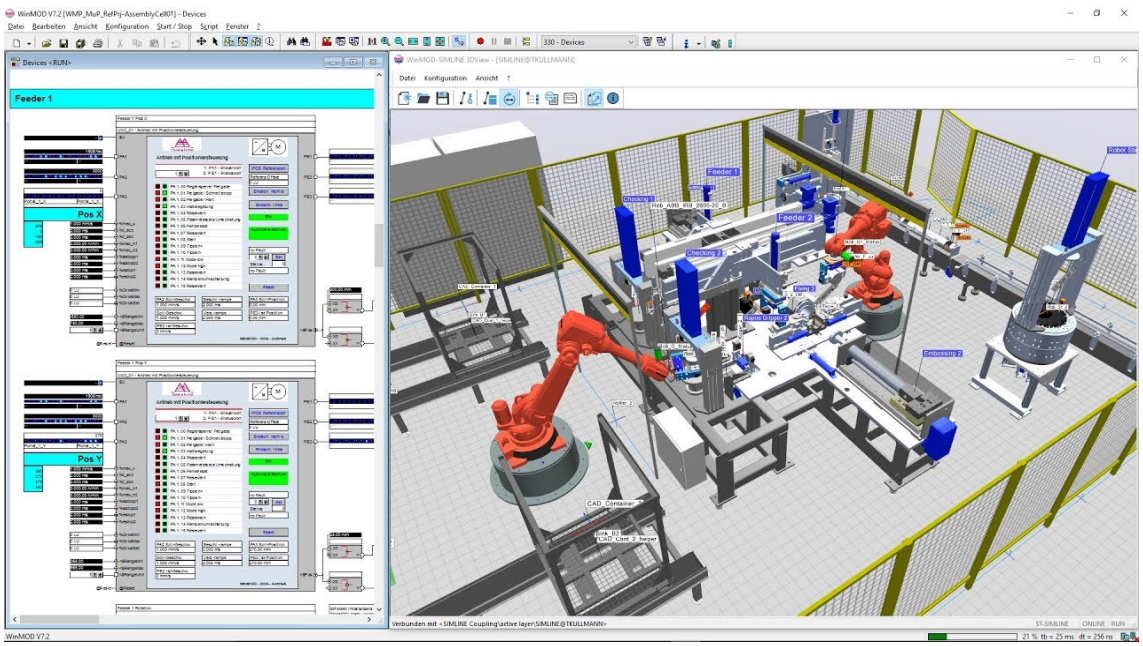
Obr. 10 Porstředí FASTSUITE

FASTSUITE Edition 2 automaticky vybere odpovídající technologický balíček na základě nástroje namontovaného na robotu. Různé balíčky technologií poskytnou GUI přizpůsobené přímo pro konkrétní technologii a vypočítají dráhu nástroje způsobem, který je již optimalizován pro danou technologii, tj. Jako je obloukové svařování nebo laserové řezání a který bude obsahovat již všechny specifické procesy příkazy pro ovladač. FASTSUITE nabízí OLP robotických operací jako bodové a obloukové svařování, laserové řezání, nástřik a pick&place. Při tvorbě pick&place operací je podpora více TCP, události k ovládání operací uchopení a uvolnění, virtuální určení pozic robotů a kontrola poloh robota prostřednictvím panelů[15].

Laserové svařování nabízí speciální integraci technologie, která byla dokonale přizpůsobena robotům TruLaser Robot 5020, TRUMPF TruLaser Weld 5000 nebo Amada FLW 4000 M3 a dalším systémům, nabízí FASTSUITE optimální podporu pro efektivní používání, provoz systému a získání vysoce kvalitních laserových svařovacích programů. Softwarová podpora výběru správných WPS z přizpůsobitelné databáze schválených parametrů svaru. FASTSUITE Edition 2 umožňuje výrobcům vytvářet, simulovat a analyzovat nástřikové aplikace a vylepšovat vzory stříkání, pokrytí povrchu, prodloužení rozprašovacího kužele a rychlosti nanášení. Simulace umožňuje optimalizaci trajektorií robotů tak, aby se rozdělilo správné množství stříkaného materiálu a dosáhlo se rovnoměrného usazování po povrchu, což zajišťuje pokrytí a zamezení plýtvání[15].

WINMOD

WinMOD je založený na modelování komunikací, zařízení, strojů a zařízení řízených automatizačními systém. Z dnešního pohledu je třeba řídit digitalizaci systémů. WinMOD pomáhá při realizaci automatizačního systému ve virtuálním světě v reálném čase. Vizualizace signálů, reakcí a časových průběhů činí chování transparentním a vnímatelným. Nahrazení systému virtualizovanými systémy lze provést zcela nebo zčásti na automatizačním systému. S využitím WINMOD lze provádět virtuální zprovoznění.



Obr. 11 Ukázka WINMOD

WINMOD nabízí možnost připojení se k jednomu nebo více automatizačním systémům a připojení k jednomu nebo více reálným nebo simulovaným subsystémům (např. Robotickým ovládacím prvkům). Dále nabízí připojení k dalším inženýrským nástrojům pro výměnu dat. WINMOD nabízí připojení k Tecnomatix Process Simulate.

WINDMOD se skládá z mnoha částí, které si může uživatel poskládat.

Systém WinMOD se obvykle skládá z:

- Systémový software WinMOD jako základní software pro inženýrské a běhové prostředí.
- Konfigurace WinMOD pro propojení s automatizačními systémy.
- Doplňky WinMOD pro speciální požadavky, jako je automatizované inženýrství, distribuovaná simulace, záznam signálu atd.
- Knihovny WinMOD k rozšíření schopností systémového softwaru WinMOD. Jsou strukturovány podle cílů použití.
- WinMOD-SIMLINE, pro 3D kinematickou vizualizaci, pro simulaci toku materiálu s knihovnami WinMOD-SIMLINE
- Knihovny WinMOD-SIMLINE – pro virtualizaci skutečných dopravních systémů
- WinMOD CoSimulation – rozšiřují možnosti systémů WinMOD tím, že je kombinují se simulačními nástroji jiných výrobců.

1.3 Možnosti simulačních programů

Knihovny – Popisují, jestli SW má knihovny robotických ramen a dalších nástrojů.
Podpora značek – Popisuje, jestli SW podporuje přední výrobce (ABB, KUKA, FANUC, atd. a i menší výrobce.
Podporované procesy – Parametr popisující, které podporované procesy lze vytvářet při programování robota
Prodejní verze – Parametr označující v jakých verzích je SW prodáván.
Cena – Cena SW

Tabulka 1 Offline programy třetích stran

SW	Knihovny	Podpora značek	Podporované procesy	Prodejní verze	Cena
Octopuz	ANO	Velká podpora předních výrobců i menších	Svařovací, nástřikové, obráběcí, řezací, pick&place, aditivní a další	Jedna	Není uvedena
RobotDK	ANO	Velká podpora předních výrobců i menších	Svařovací, nástřikové, obráběcí, řezací, pick&place, a další	Essential, Proffesional, Premium	Trial – Zdarma Education - 145€ Proffesional - 2995€ Calibration&Performance Testing – Na vyžádání
Famos	ANO, ale pouze robotů	Podpora předních výrobců	Svařování, leštění, lepení, nástřik,	Trial, Education, Profesional,	Demo – Zdarma Klasická – Není uvedena

			obrábění, řezáín, a další	Calibration&Performance Testing	
Robot master	Není uvedeno	Podpora předních výrobců	ořezávání, řezání, robotického obrábění, odstraňování otřepů, svařování, leštění, broušení	Jedna	Není uvedeno
AUTOMAPPS	Není uvedeno	Podpora předních výrobců	svařování, obrábění, pískování, řezání, aditivní výrobu a další	Jedna	Není uvedeno
Verbotics	Není uvedeno	ABB, FANUC, Yaskawa	Svařování	Jedna	Není uvedeno

SW	Knihovny	Podpora značek	Podporované procesy	VR	Prodejní verze	PLC	Cena
Tecnomatix	Ano, ale pouze roboty, ale ke stažení (jsou mimo aplikaci)	Velká podpora předních výrobců i menších	Svařovací, nástřikové, nanášecí, pick&place, a další, Komplexní tvorba výrobních procesů	ANO	Process Simulate – Podpora serverového řešení Teamcenter Robot Expert – Standalone verze	ANO	Není uvedena
Visual Components	ANO	Velká podpora předních výrobců i menších	Svařovací, nástřikové, nanášecí, pick&place, a další, Komplexní tvorba výrobních procesů	ANO	Essential, Proffesional, Premium, Demo	ANO	Essential – Není uvedeno Proffesional – Není uvedeno Premium – Není uvedeno Demo – Zdarma na vyžádání

Circos	ANO	Velká podpora předních výrobců		ANO	Trial, Education, Profesional, Calibration&Performance Testing	ANO	Na vyžádání

Tabulka 2 shrnutí SW pro tvorbu digitálních dvojčat

Knihovny – Popisují, jestli SW má knihovny robotických ramen a dalších nástrojů.
Podpora značek – Popisuje, jestli SW podporuje přední výrobce (ABB, KUKA, FANUC atd. a i menší výrobce.
Podporované procesy – Parametr popisující, které podporované procesy lze vytvářet při programování robota
VR – Podpora virtuální reality
Prodejní verze – Parametr označující, v jakých verzích je SW prodáván.
Cena – Cena SW
PLC – Možnost připojení programovatelného automatu k testování simulace

1.4 Základní typy a popis robotických soustav

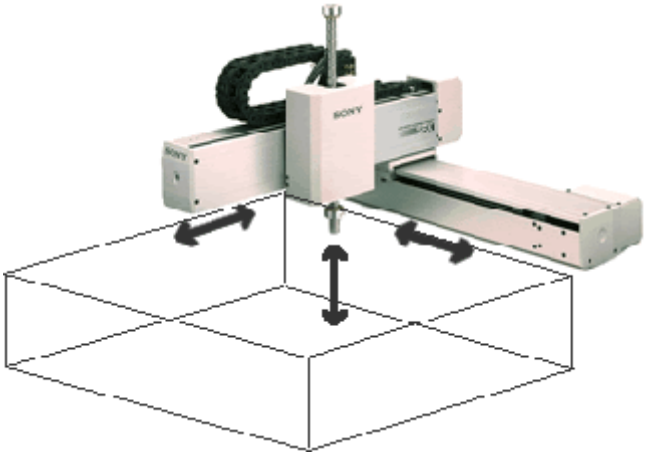
Robotická ramena jsou stroje naprogramovány tak, aby prováděly konkrétní úkol nebo práci rychle, efektivně a extrémně přesně. Obvykle jsou poháněny motorem a jsou používány pro svůj rychlý a konzistentní výkon těžkých a/nebo vysoce opakovaných postupů po dlouhou dobu. Jsou zvláště ceněny v průmyslové výrobě, obrábění a montáži. Typické rameno průmyslového robota obsahuje řadu kloubů a manipulátorů, které spolupracují, aby co nejvíce připomínaly pohyb a funkčnost lidské paže z čistě mechanického hlediska. Programovatelné robotické rameno může být kompletním strojem samo o sobě, nebo může fungovat jako samostatná robotická část většího a složitějšího zařízení. Velké množství menších robotických ramen používaných v bezpočtu průmyslových odvětví a aplikací na pracovišti je dnes umístěno na pracovním stole a ovládáno elektronicky. Větší verze mohou být namontované na podlaze, ale v každém případě mají tendenci být vyrobeny z pevného a odolného kovu (často oceli nebo litiny) s 4 až 6 kloubovými spoji. Opět z mechanického hlediska jsou klíčové klouby na robotické paži navrženy tak, aby co nejvíce připomínaly hlavní části jeho lidského ekvivalentu – včetně ramene, lokte, předloktí a zápěstí. S takovou rychlostí a výkonem, s jakou mohou ramena průmyslových robotů pracovat, je naléhavá potřeba při jejich programování a používání dbát na maximální bezpečnost. Při správném nasazení však mohou výrazně zvýšit produkční rychlost a přesnost umisťování a vychystávání, stejně jako provádět těžké zvedání a přemisťování, které by nebylo možné provádět ani pro skupiny více lidských pracovníků jakýmkoli tempem. Jak technologie pokročila a výrobní náklady robotických komponent v průběhu let klesaly, za posledních zhruba deset let došlo k velmi rychlému rozšíření dostupnosti a cenové dostupnosti robotů a robotických ramen v celé řadě průmyslových odvětví. To znamená, že se s nimi mnohem častěji setkáme v menších provozech, než tomu bylo dříve, protože již nepředstavují pouze ekonomicky životaschopnou možnost pro velké výrobní linky, které produkují velmi velké objemy produktů.

Na dnešním trhu je k dispozici mnoho různých typů robotických ramen, z nichž každé je navrženo s důležitými základními schopnostmi a funkcemi, díky nimž jsou různé specifické typy

zvláště vhodné pro konkrétní role nebo průmyslová prostředí. Většina robotických ramen má až šest kloubů spojujících sedm sekcí, z nichž většina nebo všechny jsou poháněny různými formami krokových motorů a řízeny počítačem. To umožňuje neuvěřitelně přesné umístění koncové efektorové části paže, což je ve většině průmyslových použití obecně nějaký druh specializovaného nástroje nebo nástavce, navržený k provádění vysoce specifické akce nebo opakovatelné série artikulací. Z velké části spočívá klíčový rozdíl mezi různými druhy robotických paží ve způsobu, jakým jsou jejich klouby navrženy tak, aby artikulovaly – a následně v rozsahu pohybů a funkcí, které jsou schopny vykonávat – a také v typu konstrukce, kterou mají. podporovány a půdorysem, který vyžadují pro instalaci a provoz[18].

Karteziánský

Systém, který se pohybuje ve třech ortogonálních osách (X, Y a Z) podle kartézských souřadnic. Kartézský robot provádí koordinovaný pohyb os prostřednictvím společného ovladače, kde tyto osy jsou vyrobeny z nějaké formy lineárního pohonu, a to například zakoupením jako předem sestavený od výrobce, vytvořeným na zakázku nebo koncovým uživatelem z komponent lineárního vedení pohonu.



Obr. 12 Kartézský robot

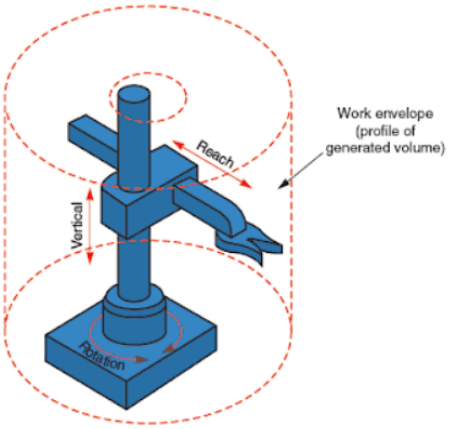
Jednou výraznou výjimkou, která nepatří pod kartézské roboty je lineární systém, který má konfiguraci 2X-Y nebo 2X-Z, tento typ konfigurace spadá do portálových robotů. Primární rozdíl mezi portálovými a karteziánskými roboty je v tom, že kartézský robot používá jeden lineární pohon na každé ose, zatímco portálový robot je vždy konstruován se dvěma základními (X) osami, přičemž druhá (Y) osa je překlenuje.

Druhý typ víceosého lineární systém, který se nebude spadat pod definici kartézského robota je tabulka XY. Rozdíl mezi kartézskými roboty a stoly XY spočívá v uspořádání montáže a nakládání. U kartézského robota je druhá nebo třetí osa (Y nebo Z) konzolová a je podporována pouze na jednom konci osou pod ní.

Kartézské roboty se v některých technických specifikacích překrývají se SCARA a 6osými roboty ale kartézské roboty mají oproti SCARA a 6osým typům několik výhod. Kartézský design poskytuje obdélníkový pracovní prostor, ve kterém je významné procento plochy robota využito jako aktivní pracovní plocha. Naproti tomu typy SCARA a 6osé mají kruhové nebo oválné pracovní obálky, které často vedou k velkému mrtvému prostoru, zvláště když je požadovaný zdvih nebo dosah velmi dlouhý. Kartézské roboty mohou být konstruovány z prakticky jakéhokoli typu lineárního pohonu s různými pohonnými mechanismy – řemenem, kuličkovým nebo vodicím šroubem, pneumatickým pohonem nebo lineárním motorem. To znamená, že mohou mít a často mají lepší přesnost polohování a opakovatelnost než typy SCARA a 6osé. Kartézské roboty mají také výhodu snadného použití z hlediska programování, protože jejich kinematika je jednodušší[19].

Válcový

Osy tvoří válcový souřadnicový systém a využívá cylindrického souřadnicového systému. Pohyb hlavního ramene je nahoru a dolů. Robot může tento pohyb provádět vysunutím válce, který je zabudován v rameni. U většiny válcových robotů je pohyb nahoru a dolů zajišťován pneumatickým válcem a rotace je obecně zajišťována motorem a ozubenými koly.



Obr. 13 Válcová pracovní obálka

Válcové konfigurace však mají některé nevýhody. Jejich celková mechanická tuhost je snížena, protože roboty s rotační osou musí při otáčení překonávat setrvačnost předmětu. Jejich opakovatelnost a přesnost je rovněž snížena ve směru rotačního pohybu. Válcová konfigurace vyžaduje sofistikovanější řídicí systém než kartézská konfigurace. Aplikovatelnost těchto typů robotů se vztahuje na strojní nakládání a vykládání, investiční lití, dopravníkové přepravy palet, slévárny a kovárny, balení masa anebo tlakové lití[20].

SCARA

Roboty SCARA, známé také jako Selective Compliance Assembly Robot Arm nebo kloubové robotické rameno, které jsou jednou z nejběžnějších forem průmyslových robotů v dnešní průmyslové oblasti. Roboty SCARA jsou vhodné pro mechanickou automatizaci v mnoha

průmyslových oblastech. Pohyb ramene je prováděno za pomoci několika tuhých tyčí a otočného nebo pohyblivého kloubu, což je otevřený prstencový kloub jedním koncem upevněný na základně, na druhém konci je volně namontovaný koncový aktuátor. Dráha pohybu konce robotického manipulátoru je prostorová křivka, v současnosti běžně používané metody plánování tratí zahrnují prostorovou společnou interpolaci a karteziánské územní plánování[21].

Obr. 14 4osý SCARA robot

Mezi výhody robotů SCARA spadá kompaktní konstrukce, která umožňuje velký provozní rozsah a malou instalační plochu, vysokou dostupnost, nízkou energetickou spotřebu díky požadovanému nízkému momentu. Mezi další výhody patří i fakt že tyto roboty, stejně jako u všech dalších neživých nástrojů, lze využít pro práci které by lidskému dělníkovi byly zdraví škodlivé.

6osý

6osý robot, také známý jako kloubový robot je hojně využíván ve flexibilní automatizaci díky své flexibilitě, síle a dosahu. Jejich typický pohyb je složen z X, Y a Z roviny, dále však také dokáží provádět akce vybočování, stoupání a přetáčení. Velká část tohoto typu robotů je spojována s lidskou paží a jejím principem pohybu, v literatuře se tedy často tyto robotická ramena nazývají právě kloubová ramena. Každý kloub obsahuje servo motor, který zajišťuje jeho pohyb a všechny tyto serva jsou řízeny kontrolerem jež se obvykle nachází v základně ramene.

Obr. 15 6osé robotické rameno

Takto využívána robotická ramena mohou mít různé nosné váhové kategorie od 5 kg až po stovky kg, dále také dosahové rozmezí a také dizajn který je vhodný pro specifická pracovní prostředí jako jsou čisté prostory až po slévárny.

2 Rozbor problematiky optimalizace trajektorií robotizovaných pracovišť

V této kapitole se budeme zabývat rozborem problematiky optimalizace trajektorií, do tohoto tématu tedy bude spadat popis co tedy vlastně trajektorie je, definice transformačních principů robotických ramen, důvody využití programování bez nutnosti zastavení stroje, a tedy tak zvané off-line programování, metody využívané pro návrh optimalizované trajektorie a také způsoby srovnání simulovaného a reálného prostředí.

Pro začátek si definujeme, co je trajektorie. Trajektorie je také nazývána jako pohybová křivka a jedná se o geometrickou čáru která se nachází v prostoru a je vytvořena opisováním pohybu tělesa nebo hmotného bodu v daném prostoru, je to tedy souhrn všech pozic, v nichž se naše těleso či hmotný bod v daném časovém úseku nacházel. Tvarem trajektorie je obecně křivka a podle tvaru této křivky dělíme trajektorie na přímočaré a křivočaré. Trajektorie nám tedy následně popisuje oblast, ve které se dané těleso nacházelo.

V robotice se tyto trajektorie upravují, aby se dosáhlo co nejoptimálnějšího a nejefektivnějšího pohybu robotického ramene bez srážky s okolními překážkami.

2.1 Definování transformačních principů pohybů robotických ramen

Transformační principy vycházejí pro robotická ramena z dopředné kinematiky. Dopředná kinematika se vztahuje k procesu získávání polohy a rychlosti koncového efektoru, daných známými kloubovými úhly a úhlovými rychlostmi. Matice transformace spojů pro každý spoj se získávají podle Denavit-Hartenbergovy (DH) konvence. Manipulátor průmyslového robota se skládá z řady spojů a článků kde anatomie robota se zabývá studiem různých kloubů a vazeb manipulátoru společně s dalšími aspekty fyzické konstrukce. Robotický spoj zajišťuje relativní pohyb mezi dvěma články robota a každý spoj zde poskytuje určitý stupeň volnosti pohybu (DoF). V mnoha případech je s každým kloubem spojen pouze jeden stupeň volnosti, z toho vyplývá že složitost robota lze klasifikovat podle celkového počtu stupňů volnosti, které má kde každý spoj je připojený ke dvěma článkům[23].

Při použití Denavit-Hartenbergovy konvence na nový manipulátorový mechanismus je potřeba dodržet následujících postupů.

- osy spojů a přiřaďte osu Z směřující podél osy i-tého spoje.
- Identifikujte společné kolmice os spojů. V bodě, kde se společná kolmice setkává s i-tou osou, přiřaďte počátek spojovacího rámce.
- osu X_i směřující podél společné kolmice.
- Přiřazení osy Y_i k dokončení pravého souřadnicového systému
- Přiřaďte rám {0} tak, aby odpovídaly rámu {1}, když je první proměnná spoje nulová.

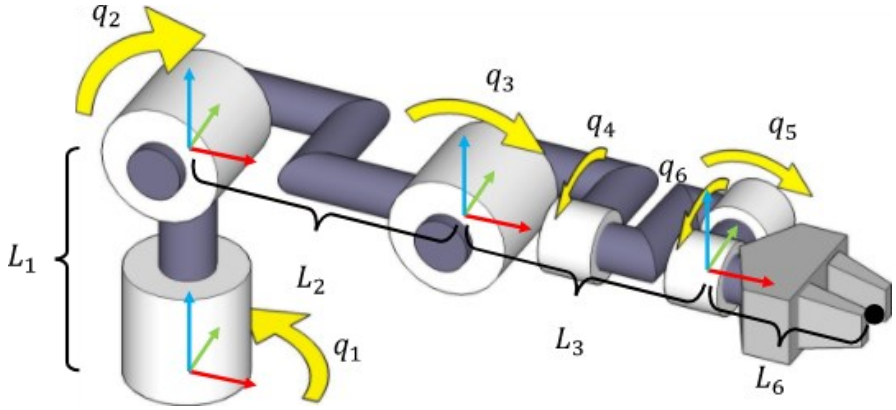
– Pro poslední snímek {n} zvolte umístění počátku a směr X tak, aby se co nejvíce parametrů vazby stalo nulovým. Poslední metoda se nazývá pravotočivý souřadnicový systém (RHCS – Right Handed Coordinate System).

Obecná transformační matice pak vypadá takto:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1}, & \cos\theta_i \cos\alpha_{i-1}, & \sin\alpha_{i-1} & \sin\alpha_{i-1}\, d_i \\ \sin\theta_i \sin\alpha_{i-1}, & \cos\theta_i \sin\alpha_{i-1}, & \cos\alpha_{i-1} & \cos\alpha_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1) $$

Tato transformace bude funkcí všech společných proměnných. Z hodnot parametrů linky (DH-tabulka) lze vypočítat jednotlivé matice transformace spojení ${}^{i-1}T_i$[23].

Specifikace a znalost kloubové konfigurace při potřebných pohybech je nutná pro co nejefektivnější pohybové principy, sestavování trajektorií a práci s objekty. Každý výrobce robotických ramen má své vlastní popisy, které lze použít. V základu se tyto principy shodují však ve 3 kloubech které při rotaci změní celkovou kloubovou sestavu a jejich náklony s ponecháním koncového bodu sestavy na stále stejné světové pozici.



Obr. 16 Typická konfigurace pro 6R manipulátor (Rameno“ má 2 stupně volnosti, „loket“ má 1 a „zápěstí“ má 3.)

V koordinaci na Obr. 16 Můžeme vidět klouby zarovnány podle os Z, Y, Y, X, Y, Z pro klouby od základny po koncový efektor s tím že všechny referenční orientace odpovídají světovému souřadnicovému rámu. Souřadnicové rámy jsou také orientovány tak, že kladné úhly spoje se otáčejí proti směru hodinových ručiček vzhledem k jejich lokálním souřadnicovým rámům, a proto můžeme konkrétněji uvést osy spojů jako +Z +Y +Y +X +Y +Z[17].

Všechny relativní transformace mají jako součást rotace matici identity. Transformace odkazu 1 se posune o L1 jednotek ve směru Z, spojka 3 směny o L2 jednotek ve směru X a spojte 4 posuny o L3 jednotky X směr, zatímco spojky 2, 5 a 6 nevyvolávají žádný posun[17].

Transformační matici mezi dvěma po sobě následujícími snímky můžeme formulovat následovně:

$$ {}^{0}T_1=\begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{1}T_2=\begin{bmatrix} s_2 & c_2 & 0 & a_2 \\ 0 & 0 & 1 & 0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{2}T_3=\begin{bmatrix} c_3 & -s_3 & 0 & a_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}T_4=\begin{bmatrix} c_4 & -s_4 & 0 & a_4 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{4}T_5=\begin{bmatrix} -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{5}T_6=\begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

(2)

Nyní tvoříme ${}^{0}T_6$ násobením matic jednotlivých matic spojů. Začneme vynásobením ${}^{4}T_5$ a ${}^{5}T_6$, výsledek je ${}^{4}T_6$, který se vynásobí ${}^{3}T_4$ a tak dále, dokud nedostaneme ${}^{0}T_6$. Horní index označuje prvotní kloub a dolní index označuje sekundární kloub, tímto tedy ${}^{3}T_4$ je vazba mezi 3 a 4 kloubem.

$$ {}^{B}T_W = {}^{0}T_6 = {}^{0}T_1.{}^{1}T_2.{}^{2}T_3.{}^{3}T_4.{}^{4}T_5.{}^{5}T_6 $$

(3)

Kde

$$ {}^{B}T_W = {}^{0}T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

(4)

Kde Px, Py a Pz jsou globální souřadnice označující prostorovou polohu koncového efektoru a výsledky pak tedy jsou:

$$ n_x = c_6 \cdot (c_5 \cdot c_1 \cdot c_{23} - s_5 \cdot (s_1 \cdot s_4 + c_4 \cdot c_1 \cdot s_{23})) + s_6 \cdot (c_4 \cdot s_1 - s_4 \cdot c_1 \cdot s_{23}) $$

$$ n_y = c_6 \cdot \left(c_5 \cdot s_1 \cdot c_{23} + s_5 \cdot (c_1 \cdot s_4 - c_4 \cdot s_1 \cdot s_{23})\right) - s_6 \cdot (c_1 \cdot c_4 + s_4 \cdot s_1 \cdot s_{23}) $$

$$ n_z = -c_6 \cdot (c_5 \cdot s_{23} + c_4 \cdot s_5 \cdot c_{23}) - s_4 \cdot s_6 \cdot c_{23} $$

$$ o_x = c_6 \cdot (c_4 \cdot s_1 - s_4 \cdot c_1 \cdot s_{23}) - s_6 \cdot (c_5 \cdot c_1 \cdot c_{23} - s_5 \cdot (s_1 \cdot s_4 + c_4 \cdot c_1 \cdot s_{23})) $$

$$ o_y = -s_6 \cdot \left(c_5 \cdot s_1 \cdot c_{23} + s_5 \cdot (c_1 \cdot s_4 - c_4 \cdot s_1 \cdot s_{23})\right) - c_6 \cdot (c_1 \cdot c_4 + s_4 \cdot s_1 \cdot s_{23}) $$

$$ o_z = s_6 \cdot \left(c_5 \cdot s_{23} + c_4 \cdot s_5 \cdot c_{23}\right) - c_6 \cdot s_4 \cdot c_{23} $$

$$ a_x = -s_5 \cdot c_1 \cdot c_{23} - c_5 \cdot (s_1 \cdot s_4 + c_4 \cdot c_1 \cdot s_{23}) $$

$$a_y = c_5 \cdot (c_1 \cdot s_4 - c_4 \cdot s_1 \cdot s_{23}) - s_5 \cdot s_1 \cdot c_{23}$$
$$a_z = s_5 \cdot s_{23} - c_4 \cdot c_5 \cdot c_{23}$$
$$P_x = a_2 \cdot c_1 + a_4 \cdot c_1 \cdot s_{23} + d_4 \cdot c_1 \cdot c_{23} + a_3 \cdot c_1 \cdot s_2$$
$$P_y = a_2 \cdot s_1 + a_4 \cdot s_1 \cdot s_{23} + d_4 \cdot s_1 \cdot c_{23} + a_3 \cdot s_1 \cdot s_2$$
$$P_z = a_3 \cdot c_2 + a_4 \cdot c_{23} - d_4 \cdot s_{23}$$

(5)

Kde označení c_n *je zkratkou pro* $cos(\theta_n)$ *a* s_n *je zkratkou pro* $sin(\theta_n)$

Tyto rovnice poskytují dopřednou kinematiku navrženého ramenního robota. Znalost proměnných robota (θ1, θ2, θ3, θ4, θ5, θ6) pak bude identifikována 0T6 a je známa poloha a orientace zápěstí robota vzhledem k základnímu rámu.

2.2 Off-line a online programování

Robotické aplikace ve výrobě vedou ke zkrácení doby výroby a zlepšení kvality. V průmyslovém prostředí existují dvě hlavní metody programování robotů online programování a off-line programování (OLP). Online programování nevyžaduje žádný další hardware a software kromě těch, které se používají pro výrobní proces. Vygenerovaný program je však velmi nepružný. Jestliže se provádí úpravy do již stávajícího procesu je potřeba zastavit výrobu, nebo daný proces.

Offline programování robota pomocí CAD softwarů má vytvořit vizuální představení robota při plnění jeho úkolu a ve fázi plánování eliminovat problémy s dosahem robota, přístupností, kolizí, načasováním, hledání ideálních trajektorií, zrychlování procesů atd. Programování robotického systému pomocí komerčního OLP softwaru dodaný výrobcem, nebo jinou firmou zabývající se touto problematikou. Další nesmírnou výhodou OLP je opakovatelnost použití kódu, flexibilita, menší prostoje během programování systému. OLP je obvykle ziskovější ve složitých aplikacích, jako je svařování, ořezávání, řezání laserem, odstraňování otřepů, termické stříkání, lakování, nanášení lepidla a další. OLP se nevyplácí u jednodušších aplikací jako je paletizace, balení, pick and place a další.

Obr. 17 Princip offline programování

2.3 Popis pohybových instrukcí robotického ramene

Pro pohyb mezi jednotlivými definovanými body se využívá u všech robotických pracovišť pohybových instrukcí, které definují trajektorii mezi těmito body, celkově jsou využívány 3 typy pohybových instrukcí a u každého výrobce se může lišit jak množství těchto instrukcí, tak způsob

jejich využívání. Tyto instrukce jsou nazývány jako Point-to-Point motion (PTP), Linear motion (LIN) a Circular motion (CIRC).

Při vytváření pohybových instrukcí se musí dbát na splnění základních požadavků, a to při vytváření cesty a následného definování trajektorie. Tyto dvě synonyma se často zaměňují, a tak je zde popsána jejich jednotlivá funkce. Cesta je sestavena z volně ležících bodů v prostoru, které jsou uspořádány a robot by je měl následovat, cesta nám poskytuje čistě geometrický popis pohybu a je často plánována globálně přičemž bereme ohled na možné překážky, způsob průchodu nebo složitá bludiště. Kdežto trajektorie je cesta obohacená o rychlost v každém bodu a tím pádem trajektorie nepotřebuje globální informace, je specifikována a navržena lokálně (založeno na principu „rozděl a panuj“) přičemž části cesty jsou řešeny jednotlivými trajektoriemi. Je požadováno, aby trajektorie mezi sebou přecházeli co nejplynuleji což znamená, že každá trajektorie přidává do svého návrhu pouze své sousedící trajektorie.

Různé robotické úkoly vyžadují různé typy pohybů robota. Některé robotické úlohy, jako je manipulace nebo bodové svařování, vyžadují pohyby bod po bodu (pohyby PTP), zatímco některé jiné úlohy vyžadují pohyby po matematicky definovaných drahách (nepřetržitá dráha, řízená dráha, pohyby CP).

PTP

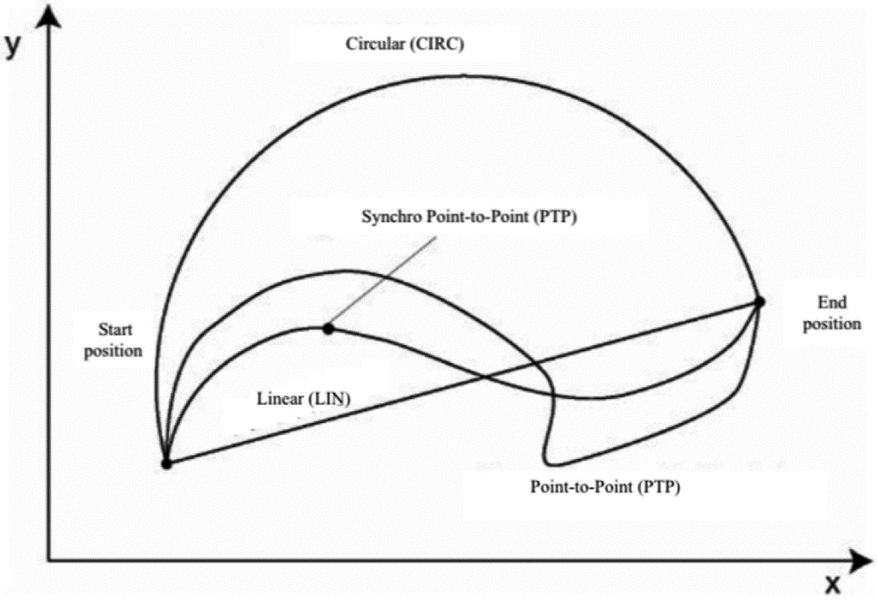
Pohyb PTP neboli z bodu do bodu, je typ pohybu, který zajišťuje nejmenší namáhavost servo pohonů při pohybu. Tato trajektorie nevede po přímce a využívá základních kruhových pohybů které se v kloubech navzájem prolínají. Robot si vypočítá podle své aktuální pozice a natočení všech os nejideálnější trajektorii a díky tomu může být pohyb proveden efektivně. Při pohybu se osy robotů pohybují synchronně z aktuálního do cílového bodu, což má za následek zakřivenou trajektorii koncového efektoru. Mezi základní nevýhody tohoto pohybu patří nemožnost odhadnout trajektorie tohoto pohybu při prvním průjezdu, díky tomu může nastat kolize s prvky jež jsou v blízkosti ramene nebo i kolize s prvky u kterých bychom neočekávali kolizi, ale které stejně zasahují do pracovního prostoru robotického ramene. Tento pohyb je znázorněn v porovnání s LIN pohybem na Obr. 18

LIN

Lineární pohyb, jak už název napovídá, je pohyb po přímce. Při využívání tohoto pohybu je přesně definována trajektorie kudy koncový bod robotického ramene projede. Nevýhodou tohoto pohybu však je vyšší namáhání kloubů a také možnost dostat se do maximálních úhlových stavů jednotlivých kloubů, jelikož při tomto pohybu se nevykonávají transformační změny, pokud je zjištěno, že se kloub dostane do maximální hodnoty a je možný ještě jiný přístup který umožní se do požadované pozice dostat. Tento pohyb je znázorněn v porovnání s PTP pohybem na Obr. 18

CIRC

Kruhový pohyb, jak už název napovídá je pohyb mezi dvěma body přes minimálně jeden bod pomocný pro definování trajektorie pro část kruhu, jež toto robotické rameno bude vykonávat a robot automaticky vypočítá poloměr. Při jeho využití se s množstvím potřebného prostoru počítá a využívá se při pohybu jen jednoho kloubu. Kruhový pohyb vykonává špička koncového efektoru a ten při tomto pohybu opisuje část kružnice jež byla vypočítána. Výsledná trajektorie tohoto pohybu je ukázána na Obr. 18



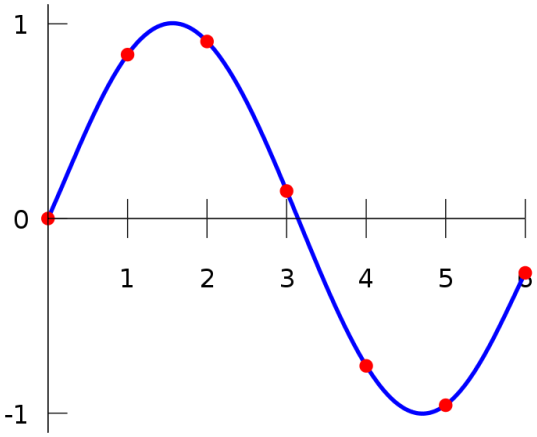
Obr. 18 Typy pohybových instrukcí[22]

2.4 Metody optimalizace trajektorie

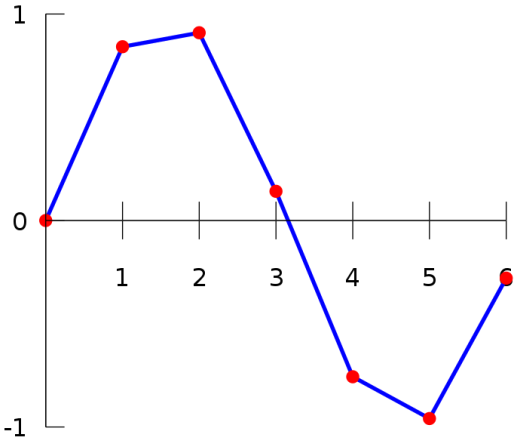
V této části kapitoly budou popsány numerické metody interpolace a aproximace pro využití optimalizace trajektorie robotického ramene a základní porozumění využití těchto numerických metod.

2.4.1 Interpolace

Úkolem interpolace je nalezení funkce v určitém intervalu, pokud známe hodnoty bodů v mezích intervalu námi neznámé funkce. V geometrii je toto znázorněno prokládáním křivky mezi jednotlivými námi známými body této funkce, na rozdíl od aproximace se interpolace liší tím, že všemi známými body přesně prochází.



Obr. 19 Interpolace polynomem 6.stupně[24]



Obr. 20 Lineární interpolace[25]

Lagrangeův interpolační polynom

Polynom naší hledané funkce dostaneme z množství sítě interpolačních uzlů, kdy n je stupeň polynomu a $n+1$ je počet uzlů (dále označovány jako x_i) a množinu funkčních hodnot v těchto uzlech $f(x_0), \cdots, f(x_n)$. Tato interpolační funkce nám potom poslouží k získání polynomu procházejícím všemi body na intervalu $\langle x_0, x_n \rangle$.

Pak tedy funkci L_n můžeme vyjádřit ve tvaru

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \cdot l_i(x),$$

kde funkce l_i jsou polynomy stupně n, které tvoří bázi prostoru všech polynomů stupně n.

Kde $l_i(x_k) = 0 \; pro \; k \neq i, \; l_i(x_i) = 1$ a jsou zadány předpisem

$$l_i(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

Newtonův interpolační polynom

Newtonův interpolační polynom je zápisem podobný Lagrangeově interpolačnímu polynomu, na rozdíl však od něj má výhodu v jednoduchosti výpočtu, pokud přidáme další průchozí bod, kdy není zapotřebí vypočítat celou funkci znovu ale jen hodnoty pro přidaný bod.

Jeho základní tvar je tento:

$$\mathrm{f(x)} = \mathrm{a_0} + \mathrm{a_1(x - x_0)} + \mathrm{a_2(x - x_0)(x - x_1)} + \dots + \mathrm{a_n(x - x_0)(x - x_1)\dots(x - x_n)}$$

A zkrácený zápis má tuto rovnici:

$$f(x) = \sum_{i=0}^{n} a_i n_i(x)$$

Kde $n_i(x)$ je:

$$n_i(x) = \prod_{j=0}^{i-1} (x - x_j)$$

Speciální vlastností Newtonova polynomu je že koeficienty lze určit pomocí jednoduchého matematického postupu. Ku příkladu, mějme $a_i(x_i, y_i) f(x_i) = y_i$.

$$f(x_0) = a_0 = y_0$$

A po přeskupení dostaneme $f(x_1) = a_0 + a_1(x_1 - x_0) = y_1$:

$$a_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

A pokud vložíme bod (x_2, y_2) můžeme spočítat a_2 do formy:

$$a_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}$$

Racionální lomená funkce

Pro využití této funkce předpokládáme že bude vhodnější využít racionální lomenou funkci než výpočet pomocí polynomických funkcí. Je definována takto.

Je dáno *m+1* bodů, pak lze zadanou funkci aproximovat pomocí racionální lomené funkce.

$$R_{i,j}(x) = \frac{p_0 + p_1 x + p_2 x^2 + \ldots + p_i x^i}{q_0 + q_1 x + q_2 x^2 + \ldots + q_j x^j}$$

Kde $i+j = m$ a lze $q_0 = 1$. Pokud je počet bodů sudý pak $i, j = m/2$, v případě lichého počtu bodů je $i = j-1$.

Interpolační racionální lomenou funkci počítáme rekurentní algoritmem, který je obdobou Nevillova algoritmu, navrženým Stoerem a Bulirschem.

2.4.2 Aproximace

Aproximace si klade za cíl nahradit danou funkci f z prostoru funkcí $\mathcal{F}$ nějakou jednodušší aproximační funkcí ϕ z prostoru aproximačních funkcí Φ. Na rozdíl od interpolační funkce však nevyžadujeme, aby se aproximační funkce ϕ shodovala v předem daných bodech x_i, $i = 0, \ldots, m$ s původní funkcí f. Důležitou otázkou v aproximační úloze je tedy otázka „míry vzdálenosti" aproximační funkce ϕ od původní funkce f. Čím je tato „vzdálenost" menší, tím zřejmě lepší aproximační vlastnosti získáváme.

Bézierovy křivky

Jedná se o křivku, která se používá k vytvoření vektorové grafiky. Ze začátku se skládá z minimálně 2 bodů kdy první a poslední bod slouží jako začátek a konec křivky a mezilehlé body definují zakřivení této křivky. Existuje hned několik typů bézierových křivek podle množství mezilehlých bodů, křivka neobsahující žádný mezilehlý bod (tedy pouze 2 bodová) se nazývá lineární křivka a křivka s jedním mezilehlým bodem se nazývá křivka kvadratická.

Tvar bézierovy křivky se vypočítá pomocí interpolace, metody aproximace dráhy přímky mezi každým kontrolním bodem.

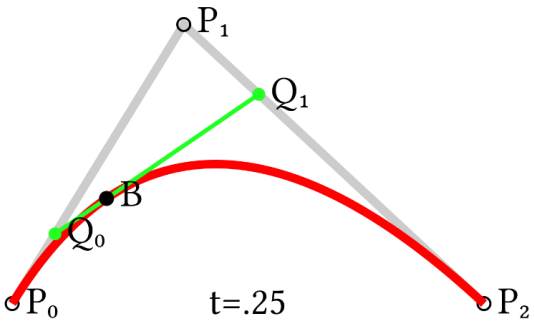
Obecná definice:

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) \cdot P_i$$

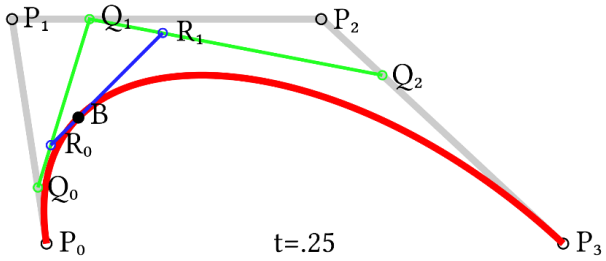
Přičemž $B_{i,n}(t)$ je i-tý Bernsteinův polynom *n*-tého stupně:

$$B_{i,n}(t) = \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i}$$

Kde $t \in \langle 0,1 \rangle$ a *n-tý* stupeň pro *n+1* kontrolních bodů které tvoří řídící polygon.



Obr. 21 Kvadratická bézierova křivka[26]



Obr. 22 Kubická bézierova křivka[27]

Toto byly zjednodušené Bézierovy křivky, pokud chceme racionální křivku, u které je možné upravit parametry váhy bodu tak tato rovnice vypadá následovně:

$$C(t) = \frac{\sum_{i=0}^{n} B_{i,n}(t) \cdot P_i \cdot \omega_i}{\sum_{i=0}^{n} B_{i,n}(t) \cdot \omega_i}$$

Kde ω_i definuje váhu daného bodu

B – spline

V numerické analýze je B-spline nebo také „basis spline“ splajn funkce která má minimální podporu na daný stupeň, hladkost a rozdělení domény. Jakákoliv splajn funkce lze být definována jako lineární kombinace B-spline stejného stupně. Oproti bézierově reprezentaci má 2 hlavní výhody, prvním je že změnou polohy kontrolního bodu měníme křivku pouze lokálně (nemění se celá křivka ale pouze oblast ovlivněná tímto kontrolním bodem) a druhý při kterém zvyšování počtu kontrolních ne nutně roste stupeň křivky.

Křivka je určena počtem *n+1* řídících bodů, stupněm *p* který určuje stupeň jednotlivých oblouků B-spline křivky a uzlovým vektorem který určuje jednotlivé napojení oblouků B-spline křivky.

Definice:

$$C(t) = \sum_{i=0}^{n} P_i \cdot N_{i,n}(t)$$

Kde P_i jsou řídící body a $N_{i,n}$ jsou bázové funkce pro B-spline křivku stupně p.

3 Návrh metodiky řízení optimální trajektorie

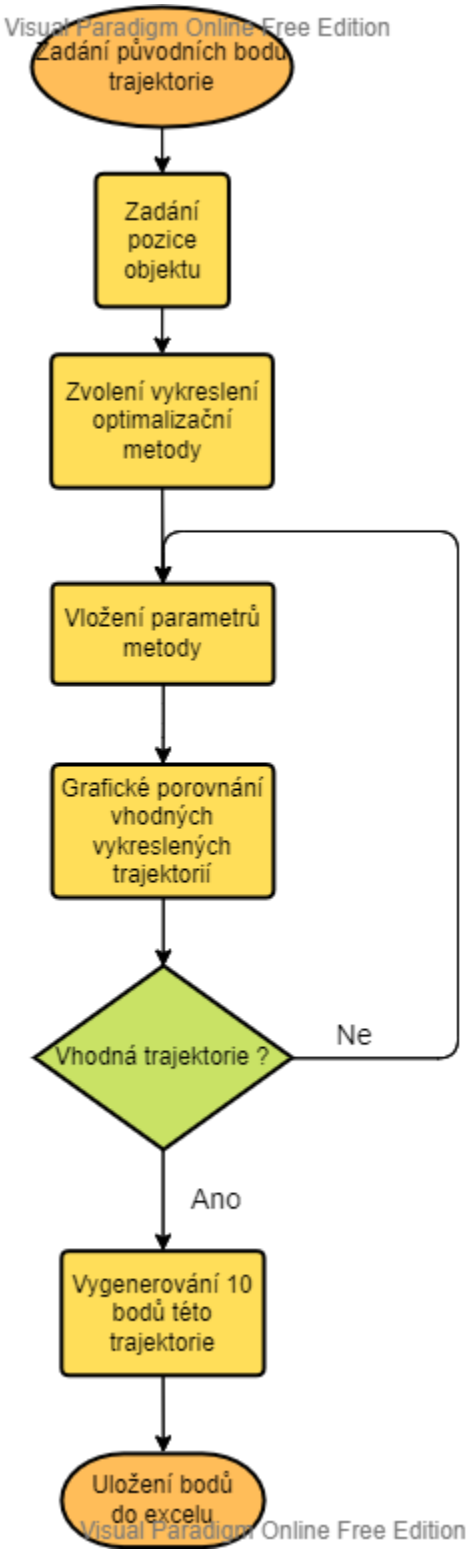
Při návrhu řízení optimální trajektorie je potřeba se seznámit s pracovištěm které bude využíváno pro následné řízení robotického ramene které je vybráno dle požadavků, které vyplývají ze simulování technologie řízení pro nejideálnější trajektorii. Pro simulaci řízení a seznámením s možnostmi je využíváno simulační prostředí firmy Visual Components od které software nese stejnojmenný název, nadále je na tento software odkazováno zkratkou VC.

3.1 Návrh koncepce řízení optimální trajektorie

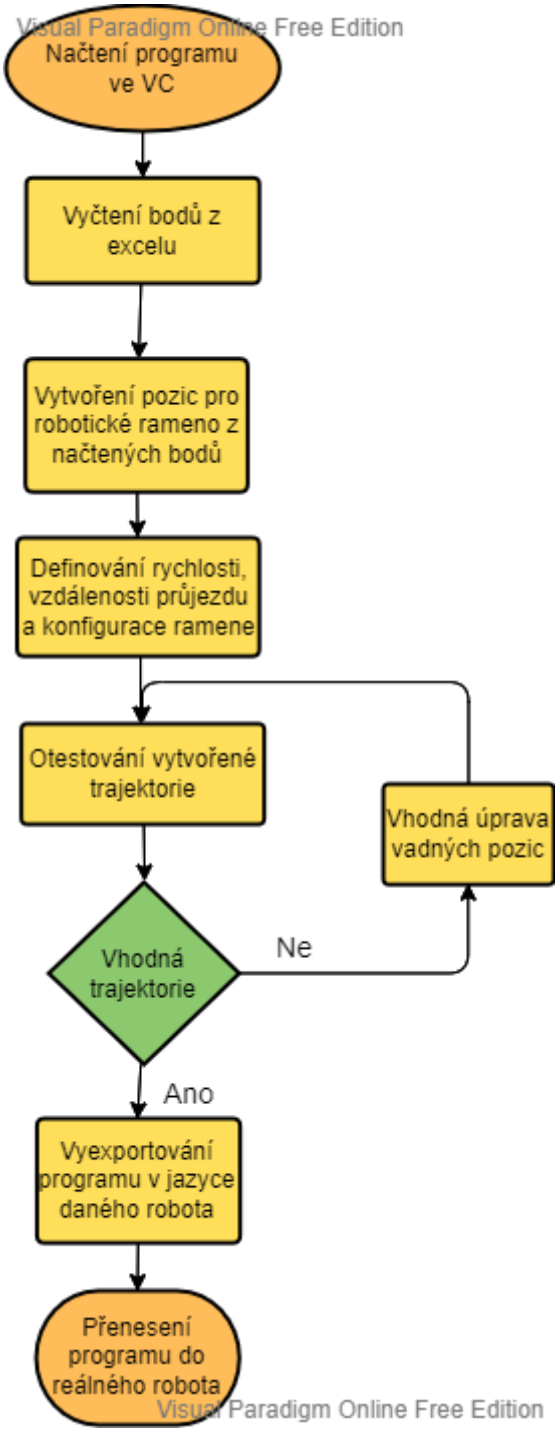
Pro návrh koncepce optimální trajektorie byl zvolen způsob zobrazený na následujícím diagramu Obr. 23, ve kterém je zkráceně ukázán postup práce a návrh optimální trajektorie oproti trajektorii základní. Jsou vytvářeny trajektorie za pomoci interpolace a aproximace v základních zvolených bodech. Pro základní trajektorii vygenerovanou simulačním softwarem platí, že prochází všemi body, a tedy bude následně porovnávána s interpolací generovanou vytvořeným programem. Pro aproximaci bude platit že bude generována s několika typy vah v meziprůchozích bodech a budou tedy jednotlivé trajektorie srovnávány v oblasti rychlosti průjezdu celou definovanou trasou a následně i délkou dráhy kterou absolvují. Ke konci bude zaměřen i pohled na jednotlivé klouby robotického ramene a budou porovnávány grafy jednotlivých průjezdových trajektorií ve kterých budou tyto klouby a jejich hodnoty vyznačeny.

Při popisování rozdílů jednotlivých trajektorií bude právě brán i pohled na pozice jednotlivých kloubů a nutná velikost jejich rotací a pohybů, může se stát, že z hlediska délky trajektorie bude výhodná specifická trajektorie ale z hlediska kloubových pohybů by byla náročnější nebo neproveditelná oproti ostatním a z tohoto důvodu bychom ji nakonec mohli vyloučit z výběru vhodných trajektorií.

Diagram na Obr. 23 Popisuje postup práce při generování graficky znázorněných návrhů trajektorií, ze kterých se následně vybírá. Postupem je zadání bodů původní trajektorie, kterou chci optimalizovat a následně zadám pozici překážky se kterou nechci, aby došlo ke kolizi. Následně vyberu testovanou optimalizační metodu, které nastavím počáteční parametry vah, které budu chtít znázornit. Výsledky se mi následně vykreslí v grafu, ve kterém je možné graficky odvodit nejvhodnější trajektorii kterou následně použít pro další práci. V případě, že žádná z trajektorií není vhodná zadám nové počáteční hodnoty pro vygenerování nových trajektorií, případně pozměním i zvolenou metodu. Následné váhové hodnoty nejvhodnější trajektorie využiji pro generování 11 bodové trajektorie která mi tyto body vloží do excelu pro další práci.



Obr. 23 Grafický výběr optimální trajektorie

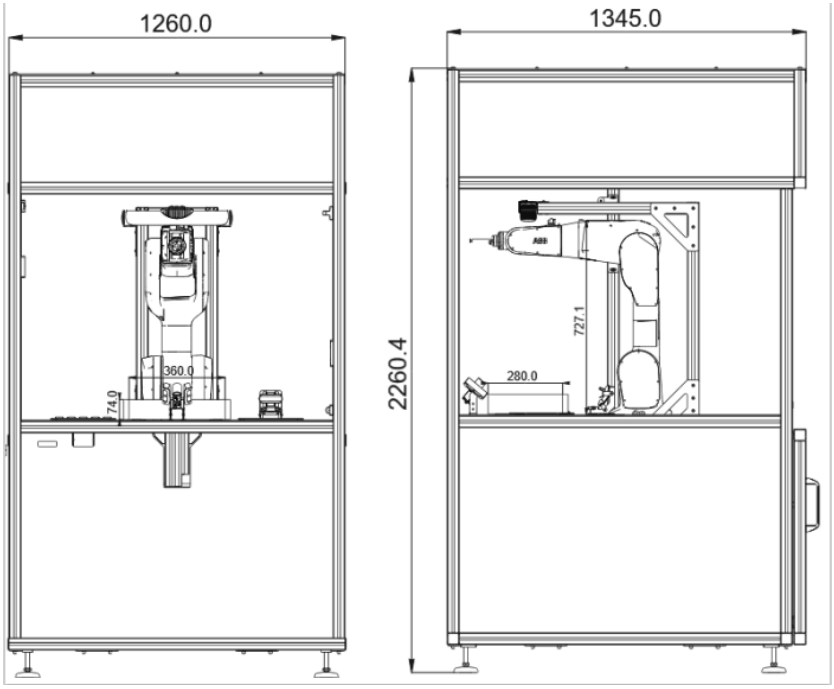


Obr. 24 Ověření trajektorie v simulaci

3.2 Návrh hardwarového řešení robotického pracoviště

Využívané pracoviště má sloužit k většímu množství využití pro přenos a úpravu prvků, aktuální model je definován jako ukázkový pro vykládání bloků čokolády nasypané v plastové

krabici na středu podpůrné plochy a možností úpravy otočení těchto bloků čokolády za pomoci menšího kovového modelu ze kterého je následně možné robotickým ramenem tento blok odejmout. Celá sestava se skládá z konstrukce, robotického ramene firmy ABB verze 1200/09 řízený kontrolérem IRC5 compact, toto robotické rameno je programovatelné za pomoci softwaru s názvem RobotStudio od výše zmíněného výrobce ABB. Dále tento model obsahuje 3D Skenovací hlavu Photoneo Phoxi M, prvky pro kompletní funkčnost obsahují Vision kontroler, Bin picking studio a komunikaci mezi photoneo a robotem řešenou přes ethernet za pomoci socketů. Dále se v pracovišti nachází nástroj (efektor) pro vyzvednutí daných bloků čokolády a zde je využíváno přísavky která vytvoří po dosednutí na rovnou stranu čokolády podtlak.



Robotické rameno ABB 1200/0.9

Robot ABB IRB 1200-5/0.9 je rychlejší a více kompaktní než obdobná robotická ramena v této třídě, jeho maximální nosnost je 5 kg a maximální dosah 901 mm, je ideální pro manipulování s využívaným materiálem a také pro obsluhu strojních zařízení. Má standartní hodnocení IP40 a je také snadno integrovatelný díky svým 4 vzduchovým kanálům, 10 nastavitelným signálům a Ethernetovým schopnostem.

Tabulka 3 Fyzické specifikace robota

Robotické specifikace	
Os:	6
H-Dosah:	901 mm
Usazení:	Podlaha, Obrácený, Úhel
Zatížení:	5 kg

Opakovatelnost:	±0.06mm
Hmotnost:	54 kg
Struktura:	Kloubový

Tabulka 4 Kloubové specifikace robota

	Rychlost jednotlivých kloubů	Rozsah pohybu robota
J1	288°/s (5.03 rad/s)	± 170°
J2	240°/s (4.19 rad/s)	130°, -100°
J3	300°/s (5.24 rad/s)	70°, -200°
J4	400°/s (6.98 rad/s)	± 270°
J5	405°/s (7.07 rad/s)	± 130°
J6	600°/s (10.47 rad/s)	± 360°

Obr. 25 Robotické rameno ABB IRB 1200/0.9

Kontroler IRC5

Oproti běžnému kontroléru je tento kontrolér menší a kompaktnější se stejným výkonem jako má běžný kontrolér. Díky menší velikosti má jednodušší uvedení do provozu, a to díky jednofázovému příkonu, další výhodou jeho kompaktnosti je ušetření místa, které může být následně využito pro další přístroje využívané při práci. Jednou z jeho posledních výhod je 16 vstupů a výstupů pro vnější připojení všech typů signálů, ty jsou umístěny v rozšiřitelném modulu.

Tabulka 5 Parametry kontroléru

Vstupní napájení	Single phase 220/230 V, 50-60 Hz
Rozměry	320 x 449 x 442 mm
Váha	28.5 kg

Okolní teplota	0-45 °C
Relativní vlhkost	Max. 95 % non condensing
Ochranný stupeň	IP20

Obr. 26 Kontrolér IRC5 Compact

Efektor

3.3 Návrh softwarového řešení

Z uvedených možných simulačních softwarů v první kapitole byl vybrán software třetí strany s názvem Visual Components, mezi důvody volby tohoto softwaru spadají volně dostupné add-ony, již naimplementované komunikační protokoly pro možnost propojení simulovaného prostředí s reálnými roboty od specifických výrobců nebo také propojení s komplexní simulací motorů, pístů a měničů (kde příkladem je propojení s WinMOD).

Pro základní práci tedy převedeme náš 3D model pracoviště do tohoto prostředí a z katalogu vybereme námi používané a výše zmíněné robotické rameno, tímto způsobem vytvoříme simulované prostředí, které se bude podobat tomu reálnému. Následně se zde nadefinují akce, jež budou ramenem prováděny a budou se také testovat jednotlivé zvolené trajektorie u kterých budou měněny jejich vlastnosti jako je rychlost, vzdálenost průjezdu od následujícího bodu nebo zvolená konfigurace pro aktuální pozici.

V daném simulovaném prostředí se také budou nacházet překážky kterým je zapotřebí se vyhnout. Toto si musíme během plánování trajektorií uvědomit a testovat jednotlivé pozice, které se nacházejí poblíž těchto překážek. Testování bude prováděno v simulačním prostředí s možností zvýrazněním kolizí a také zastavením, pokud se kolize vyskytne. Čistě PTP program, čistě LIN pohyb, popsat rozdíly.

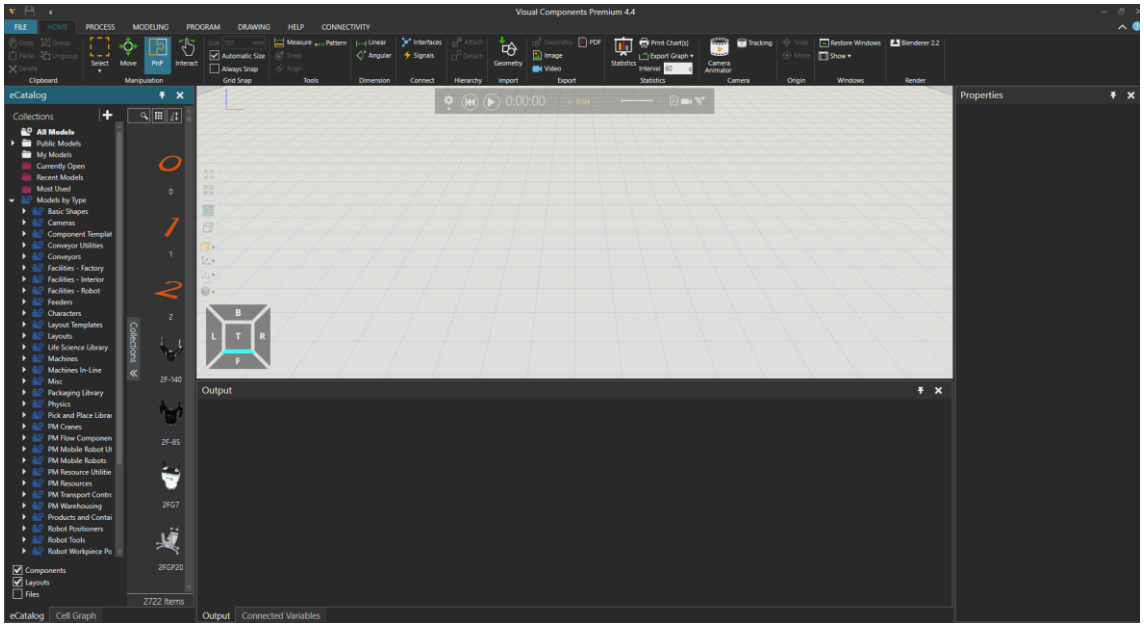
4 Implementace digitálního dvojčete s prvky dynamiky a kinematiky 3D modelu robotizovaného pracoviště

Daný model mi byl poskytnut ve firmě, ve které také vypracovávám diplomovou práci. Model je předán jako soubor s koncovkou .stp což je jeden z typických koncovek CAD modelů. Tento soubor budu vkládat importovat do simulovaného prostředí ze softwaru Visual Component ve kterém budu následně vytvářet množství kinematických drah pro experimentální nalezení těch nejvhodnějších s porovnáním dat získaných z grafů pro měření hodnot jednotlivých kloubů robotického ramen a dobu průjezdu těchto tras.

4.1 Vkládání a úprava 3D modelu

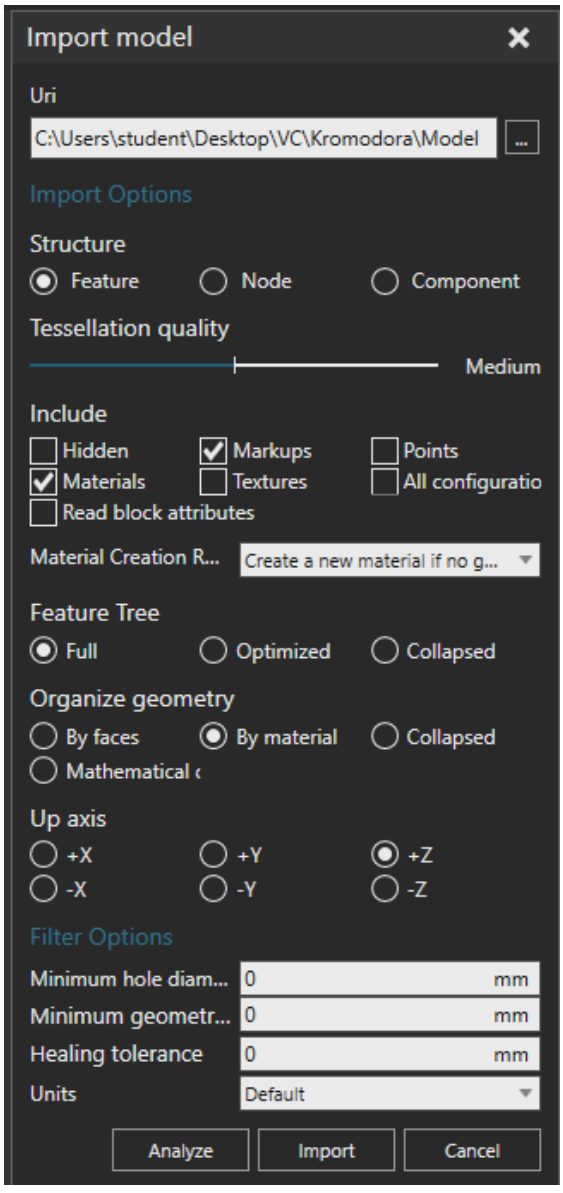
Pro práci s Visual Components je ze základu vhodné ale nikoliv nutné podívat se na stránky výrobce kde lze nalézt množství video ukázek práce s jednotlivými komponenty, programy, ukázkami pracovních ploch či ukázek manipulace s kompletními stanovišti.

Při otevření se nám zobrazí základní plocha skládající se z několika záložek, u těch záleží, jakou máme verzi VC, jelikož verze Essential neobsahuje nic víc než 4 záložky které slouží k vkládání modelů, hlavní záložce s knihovnou, záložku obsahující pomůcky a „Help“. První důležitou složkou je knihovna, ta působí jako zdroj modelů s již připravenými vlastnostmi, které lze použít v naší simulaci, díky této neustálé vyvíjející se knihovně je možné mít stále propracovanější simulační prostředí s různými typy prvků od různých výrobců. Mezi tyto prvky patří například robotická ramena, dopravníky, modely pracovníků, techniky anebo pracovních objektů.



Obr. 27 Pracovní plocha VC

Pro možnost vkládání vlastních objektů je zde importování 3D modelů které mohou být nejen importovány z mnoha různých softwarů ale také exportování těchto modelů pro možnost otevření v jiných softwarech. Pro importování modelu je možné využít 2 typy přístupu, první přístup je v jednoduchém a nenáročném přetažení modelu do projektu kdy daný model se nám následně po načtení dat zobrazí. Druhým přístupem je využití importování modelu s odkazem na jeho umístění, při tomto výběru se nám nabídne více možností úpravy daného modelu, který nám při správném použití a velkých modelech pomůže ušetřit výpočetní výkon potřebný pro tento model, všechny tyto nastavení lze vidět na Obr. 21.



Obr. 28 Nastavení importování modelu

Mezi otevřetelné a importovatelné modely patří například přípony .stp, .step, .3ds, .dwg, .dae, .3dm.

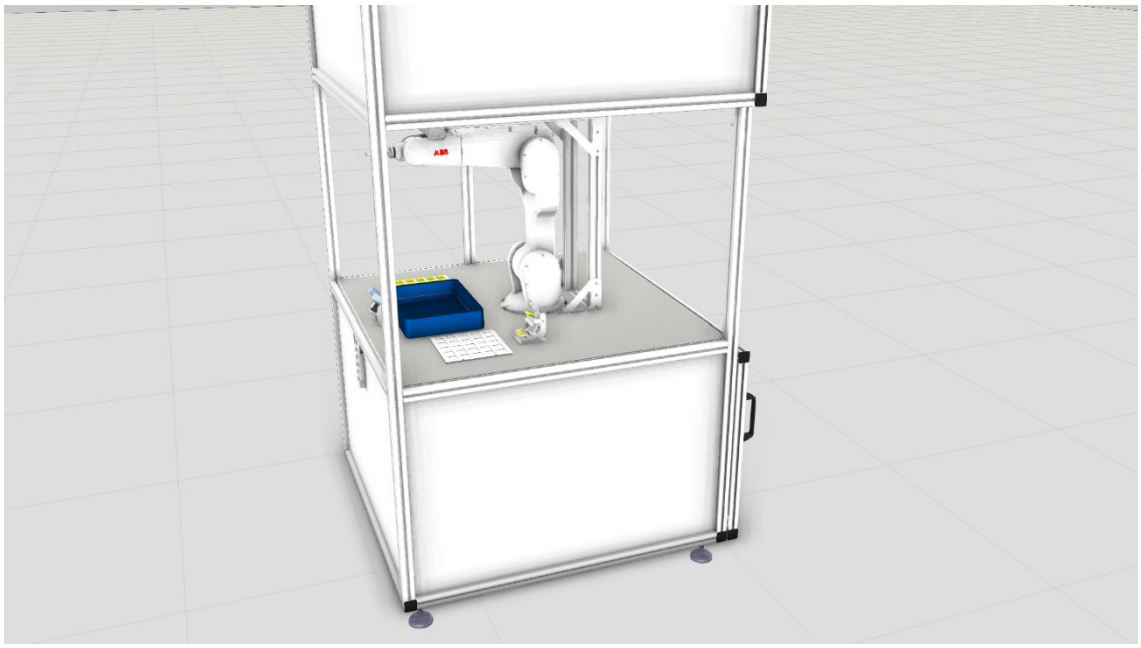
3D Manufacturing Format	1.2.3	.3mf	✓	✓
3D Studio	All	.3ds	✓	✓
ACIS	Up to 2020	.sat, .sab	✓	✗
ASCII Point Cloud file	All	.xyz, .pts, .xyzrgb	✓	✗
Autodesk FBX	FBX ASCII: 7100 to 7400. Binary: all.	.fbx	✓	✗
Autodesk Inventor	Up to 2022	.ipt, .iam	✓	✗
Autodesk RealDWG	AutoCAD 2000-2019	.dwg, .dxf	✓	✓
Binary point cloud point	All	.bxyz	✓	✗
CATIA V4	Up to 4.2.5	.session, .dlv, .exp	✓	✗
CATIA V5	Up to V5_6R2021	.CATDrawing, .CATPart, .CATShape, .cgr	✓	✗
CATIA V6	Up to V5-6 2019	.3dxml	✓	✗
COLLADA	Any	.dae	✓	✗
Creo	Pro/Engineer 19.0 to Creo 8.0	.asm, .neu, .prt, .xas, .xpr	✓	✗
GL Transmission Format	2.0 only	.gltf, .glb	✓	✗
I-deas	Up to 13.x (NX5) and NX I-deas 6	.mf1, .arc, .unv, .pkg	✓	✗
IFC2x	2 to 4	.ifc, .iczip	✓	✗
IGES	5.1 to 5.3	.igs, .iges	✓	✗
Igrip/Quest/VNC	All	.pdb	✓	✓
JT	Up to v10.5	.jt	✓	✓
Parasolid	Up to v33.1	.x_b, .x_t, .xmt, .xmt_txt	✓	✗
PRC	All	.prc	✓	✓
Revit	2015 to 2021	.rvt	✓	✗
Robface	All	.rf	✓	✗
Rhino	from 4 to 7	.3dm	✓	✗
Solid Edge	V19-20, ST-ST10, 2021	.asm, .par, .pwd, .psm	✓	✗
SolidWorks	From 97 up to 2021	.sldasm, .sldprt	✓	✗
Name	**Version**	**Extension**	**Import**	**Export**
Stereo Lithography (ASCII and Binary)	All	.stl	✓	✓
U3D	ECMA-363 1st, 2nd and 3rd editions	.u3d	✓	✓
Unigraphics (Siemens PLM software NX)	V11.0 to v18,NX to NX12.0, and NX1847 to NX1980 Series	.u3d	✓	✓
VDA-FS	1.0 and 2.0	.vda	✓	✗
VRML	1.0 and 2.0	.wrl, .vrml	✓	✓
Wavefront	All	.obj	✓	✓

Obr. 29 Seznam podporovaných modelových programů

Jakmile máme tento model naimportován do našeho prostředí, můžeme s ním okamžitě začít pracovat, od verze VC Professional se nám otevírá záložka modeling, ve které se nacházejí prvky na úpravu těchto modelů. V této záložce navíc také otevřeme, z jakých prvků se daný model skládá a můžeme je začít upravovat, mazat, měnit velikosti, ořezávat, spojovat anebo i nechat „explodovat“. Díky této záložce tedy můžeme prvek upravit tak aby byl vhodný na práci. Mezi další prvky, které tato záložka obsahuje jsou také různé typy vlastností, napojení signálů a chování které se našim modelům dají přidružit. Ve výsledku je každý model objekt, jenž má své specifické vlastnosti a chování které definují k čemu tento model může sloužit a jakým způsobem se bude v prostředí chovat.

4.2 Rozčleňení modelu

Kompletní model pracoviště je označován jako pracovní buňka, tato buňka se skládá z úložné části s dvířky, ve které je uložen kontrolér pro robotické rameno a následně z průhledné oblasti ve které je uprostřed umístěno naše rameno a před ním ležící odběrová pozice.



Obr. 30 3D model

Na modelu robotického ramene je také umístěn model využívaného nástroje, ze základu je tento model také bez jakýchkoliv vlastností ale díky vnitřní funkci VC v záložce modeling → Wizards nalezneme Component wizard End effector který nám na vybraný komponent přidá vlastnosti nástroje. Mezi tyto vlastnosti patří Tool container ve kterém se nachází všechny koncové body pro nástroje, Grasp container který slouží k uskladnění vyzvednutých komponent a poslední je Mouninterface ve kterém se nastaví pozice a směrování pro PnP (Plug ‘n Play). Díky právě zmíněnému PnP můžeme snadno tento nástroj přichytit k robotickému rameni a následně pro pohyb i zvolit tento koncový nástroj.

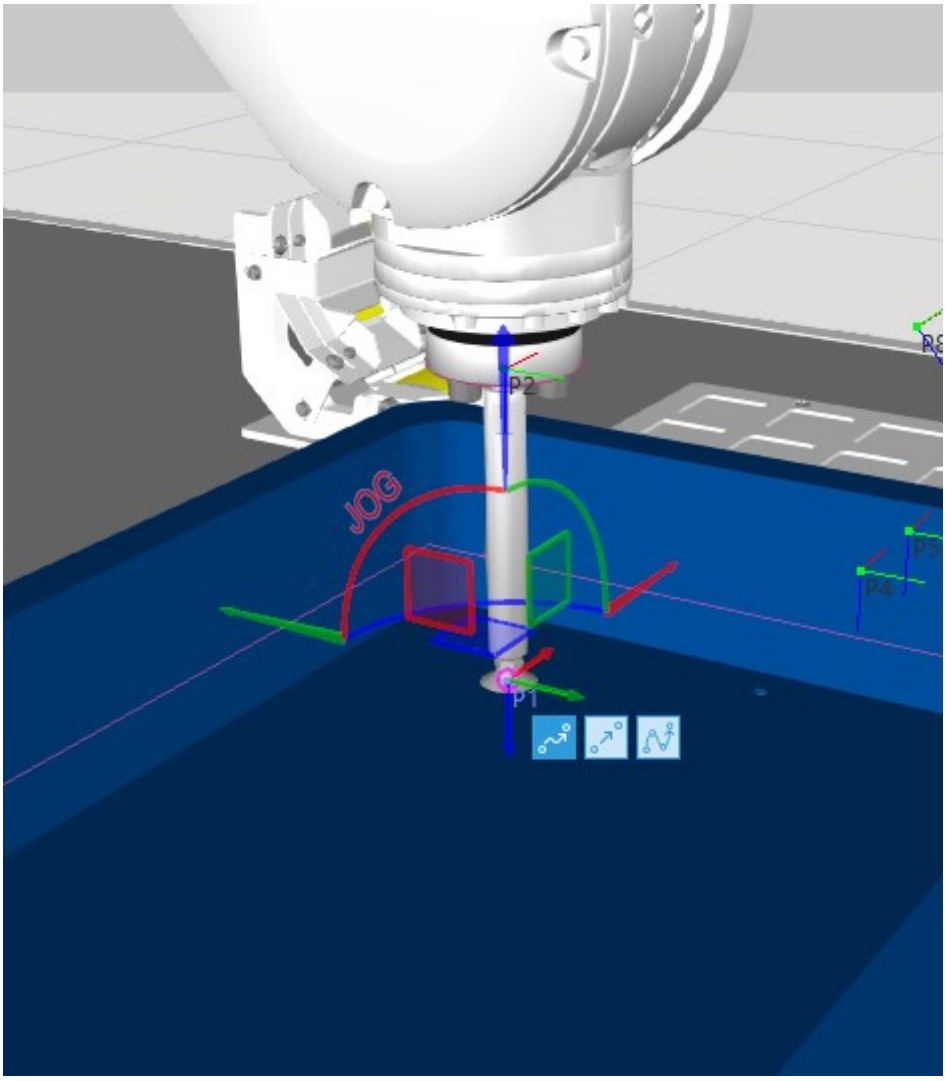
Obr. 31 Robotický nástroj

4.3 Definování počátečních a koncových bodů

V této kapitole budou nadefinovány body, které budou následně využity pro začátek a konec trajektorie. Tyto body budou ukázány v prostoru modelu a budou popsány proč byly tyto body tímto způsobem zvoleny. Body v ose X budou natočen o 180° z důvodu směřování nástroje směrem dolů a body Y jsou v nulovém úhlu, jediný rozdíl tedy v natočení by mohl být v ose Z za kterou je následně zodpovědný kloub 6 který rotuje s nástrojem.

Počáteční bod

Počáteční bod je zvolen uvnitř dané krabice, která může následně obsahovat různé díly nebo ukázkové prvky. Byl definován jako základní výjezdový bod, ze kterého bude rameno odcházet po uchopení prvku z krabice a zároveň je bodem ze kterého přejde do uchopovacího módu po odložení prvku mimo box.

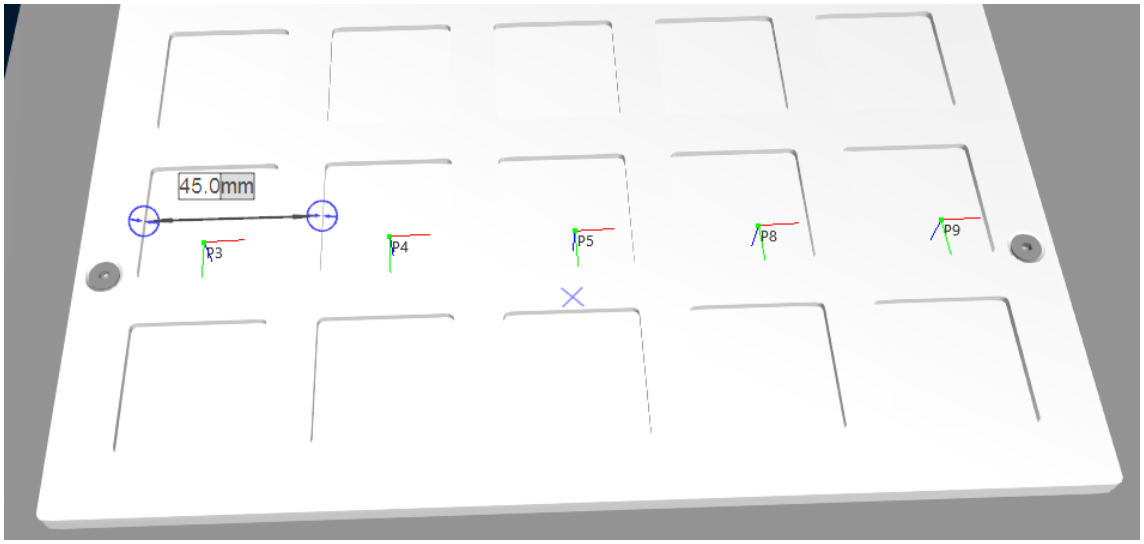


Obr. 32 Počáteční bod trajektorie

Na Obr. 32 můžeme vidět 2 XYZ soustavy, mohutnější XYZ soustava se směrovými šipkami je posuvná osa pro světovou koordinaci (X – červená, Y – zelená, Z – modrá), sekundární a menší XYZ soustava u které můžeme vidět směrování osy Z do boxu je definování pozice pro daný nástroj který se v této pozici právě nachází (označován jako TCP což je zkratkou pro „Tool Center Point“).

Koncový bod

Koncových bodů je zvoleno několik, a to pro možnost porovnání většího množství vypočítaných trajektorií, tyto koncové body se liší pouze vzdáleností na ose X, tato vzdálenost je 45 mm mezi jednotlivými body. Všechny tyto body mají stejnou hodnotu v Y ose pro zjednodušení výpočtů na 2D prostor a také se všechny nacházejí ve stejné výšce nad odkládacím prostorem.

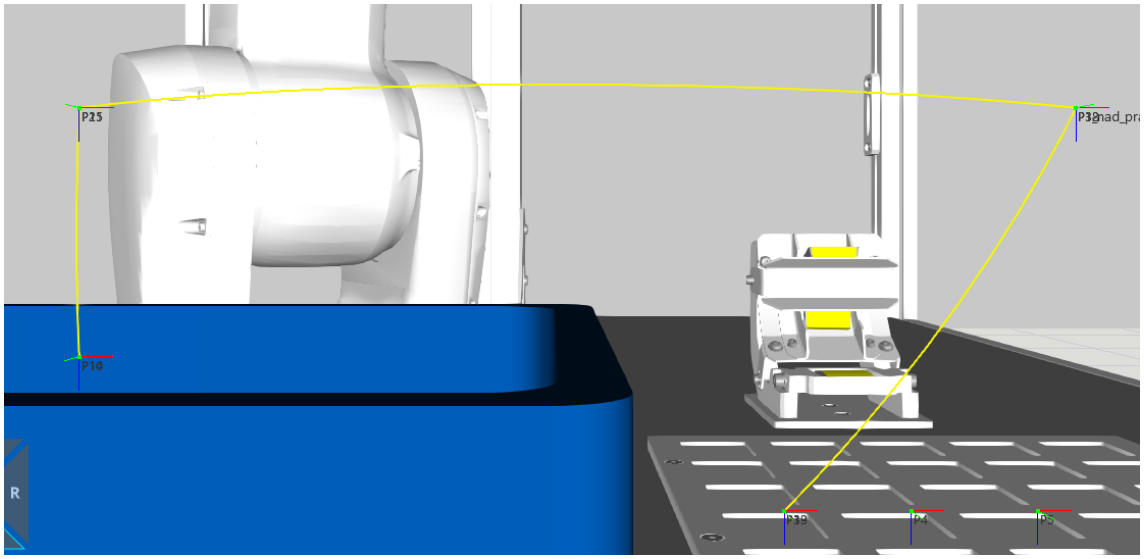


Obr. 33 Koncové body

Pro každý z těchto bodů bude vytvořena jedna trajektorie, která bude i následně optimalizována vůči počáteční trajektorii navrženou pouze spojnicí bodů.

Úvodní zaznamenané trajektorie

Úvodní trajektorie budou vytvořeny pro pohyb z boxu na každý ze dvou odkládacích ploch a následně i přesun mezi těmito odkládacími boxy, tyto trajektorie se každá bude skládat z počátečního a koncového bodu a následně ze 2 dalších průchozích bodů. Následné trajektorie budou vykresleny ve VC pomocí funkce Trace.



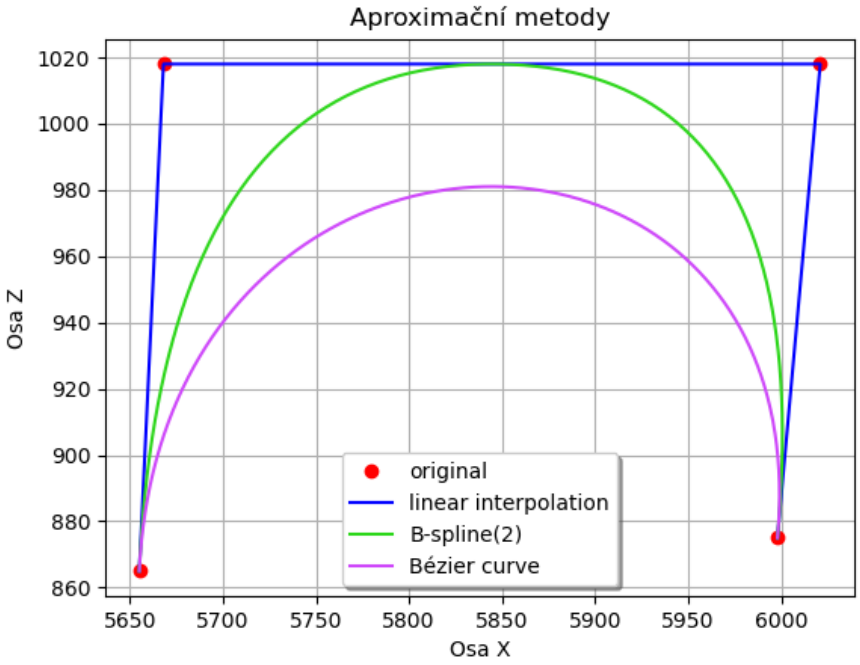
Obr. 34 Trace trajektorie pro přesun z boxu nad pravý odklad

5 Optimalizace trajektorie dle zvolených kritérií a metod pomocí digitálního dvojčete

V této kapitole byly testovány jednotlivé vybrané aproximační metody, pomocí kterých byly definovány trajektorie, tyto trajektorie byly hodnoceny dle zvolených kritérií a následně popsány uvedené výsledky z celkového návrhu. Navrhované trajektorie jsou také vyneseny do grafů pro grafické zhodnocení a vizuální porovnání rozdílů mezi trajektoriemi jednotlivých metod tak porovnání rozdílů mezi využitými metodami.

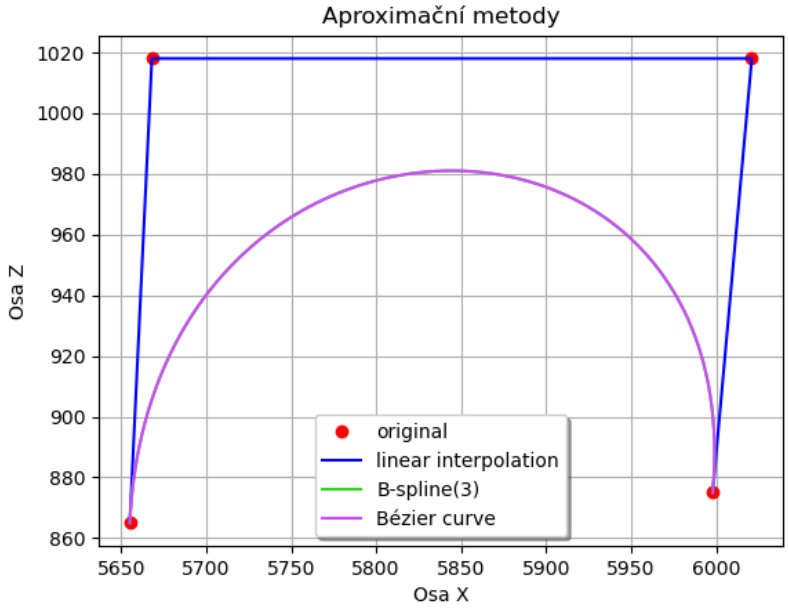
5.1 Prvotní návrh trajektorií

V prvotním návrhu těchto trajektorií byly zvoleny parametry bez možných změn. Toto odpovídá využití Bézierových křivek bez zadávání váhové úpravy a také křivky B-spline druhého řádu. Na Obr. 35 můžeme vidět vykreslení této bézierovy křivky, B-spline křivky a původní trajektorie mezi zadanými body, zároveň osy na těchto grafech již znázorňují světové souřadnice simulace, jelikož se konečný kód a jeho data budou přímo převádět do simulačního prostředí a samostatně generovat body pro průchozí trajektorii.



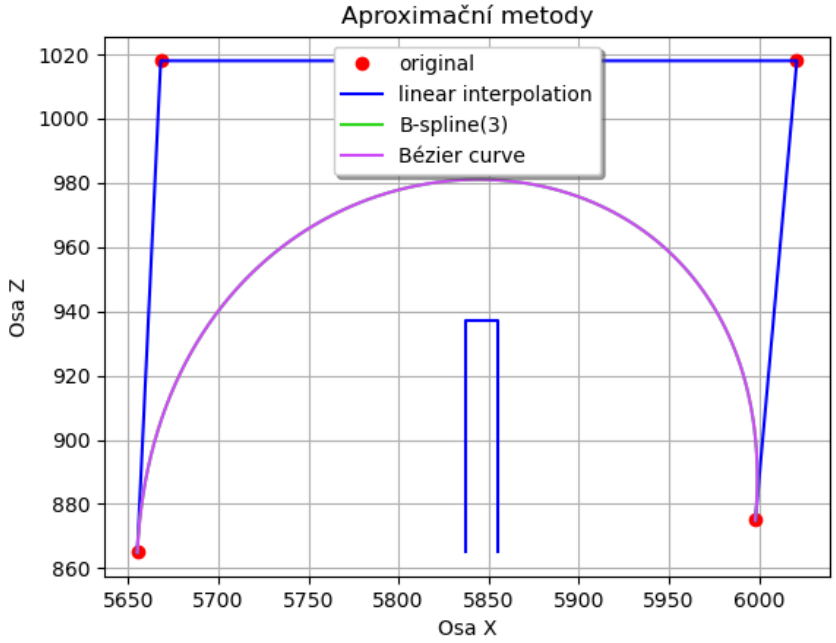
Obr. 35 Základní trajektorie pro vybrané metody

Pro možnost vidět rozdíl mezi jednotlivými řády funkce B-spline byl měněn její řád mezi hodnotami 2-5 ale již při třetím řádu se podle grafického zobrazení shodovala s vykreslenou bézierovou křivkou a se zvyšováním řádů se již skoro vůbec nelišila, třetí řád je ukázán na Obr. 36.



Obr. 36 Shodné trajektorie pro základní bezierovu křivku a B-spline 3 řádu

Pro následné grafické zobrazení byl přidán náš objekt, který je zobrazen na Obr. 37. Je přidán do předchozího grafu, který znázorňuje oba typy aproximačních funkcí a následně se tedy s tímto objektem bude počítat, tento objekt je ve své podstatě okraj bedny, ve které budou umístěny uchopitelné prvky.

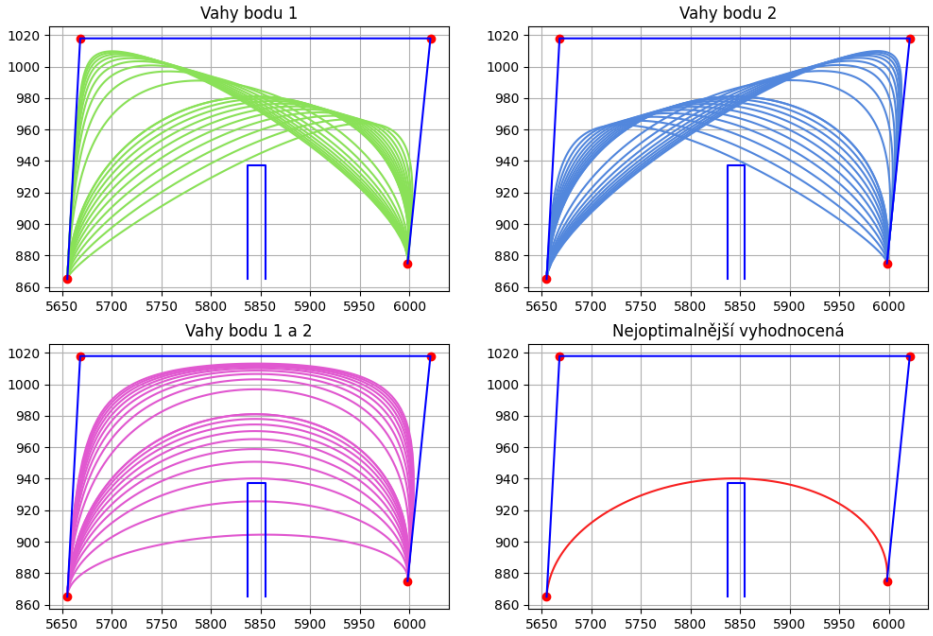


Obr. 37 Grafické zobrazení objektu v prostoru

5.2 Zvolená metoda

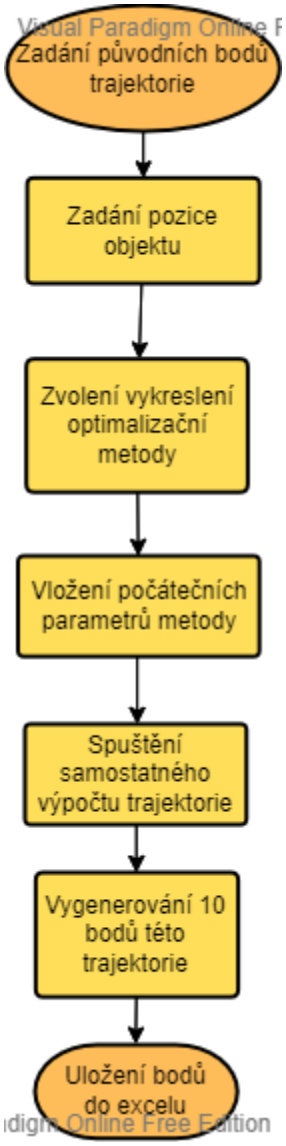
Jakožto zvolenou metodu pro návrh trajektorie byla vybrána metoda Bézierových křivek, hlavním důvodem byla možnost úpravy jednotlivých vah bodů, díky které můžeme zakřivit trajektorii pouze v dané oblasti a nemusíme měnit váhu celé křivky.

Díky této metodě jsem mohl otestovat několik váhových kategorií pro jednotlivé body. Prvotním úkolem bylo vytvořit grafické zobrazení jednotlivých vah v jednom grafu, tyto grafy jsou na Obr. 38 , celkem se zde nacházejí 4 grafy kde 3 z těchto grafů jsou ukázkou a znázorněním jednotlivých vypočtených trajektorií s různými váhami pro specifické body, jelikož naše vstupní trajektorie obsahovala 4 body tak máme ukázku 3 grafů, a to z důvodu úpravy váhy pro samostatný bod 2, úpravu váhy pro samostatný bod 3 a následně stejné váhové změny pro bod 2 a 3. V každém z takto vygenerovaných grafů je 20 trajektorií kde 10 z nich se nachází v rozmezí vah 0.1–1.0 s krokem 0.1 a dalších 10 trajektorií u kterých se váha pro dané body nachází v rozmezí 1-10 s krokem 1.



Obr. 38 Grafické vykreslení vícero trajektorií zvolené metody

Pro graf 4 byla vytvořena vlastní funkce která vytvoří prvotní téměř přímou trajektorii mezi počátečním a koncovým bodem a následně porovnává, jestli body této trajektorie procházejí daným objektem, výsledkem této funkce je první trajektorie, která již tímto objektem neprochází a vrátí nám i hodnotu jednotlivých vah. V této funkcí se také jednotlivé váhy přizpůsobují dle pozice okrajů daného objektu, tímto mi může daná funkce vrátit i hodnoty, ve kterých se váha jednoho bodu liší oproti bodu druhému.



Obr. 39 Diagram samostatného nalezení optimální trajektorie

Následně je možné graficky pozorovat která trajektorie by mohla být vhodná, v programu je také implementována funkce, která vypočítá celkovou dráhu těchto trajektorií a ve výsledku je vytisknuta délka nejkratší dráhy která se mine s objektem a také vrátí i váhy pro tuto trajektorii.

Ve výsledku ke grafům v Obr. 38 jsou vyvedeny hodnoty s délkou trajektorie a váhami dané trajektorie.

Tabulka 6 Nejvhodnější trajektorie jednotlivých návrhů

Graf	Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)
Změna váhy bodu 1	[1, 0.1, 1, 1]	439.78

Změna váhy bodu 2	[1, 1, 0.1, 1]	428.08
Změna váhy bodu 1 a 2	[1, 0.3, 0.3, 1]	397.52
Optimální vyhodnocená	[1, 0.3, 0.3, 1]	397.52

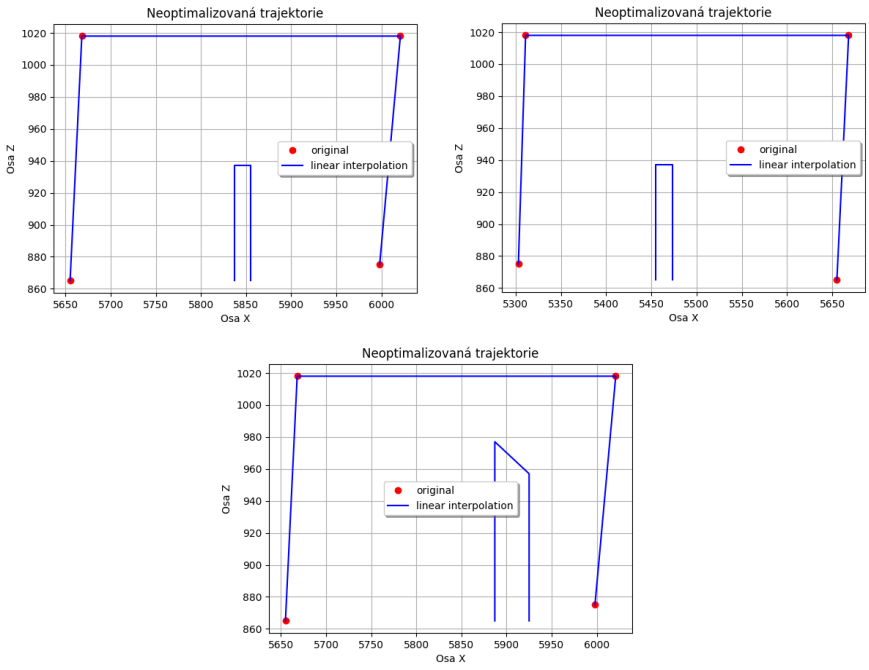
Výpočtem je zjištěna optimální trajektorie s váhami a její délkou, tyto údaje jsou následně předány do dalšího kódu, který připraví trajektorii o 10 bodech, ty jsou následně uloženy do Excel souboru, ze kterého bude následně čerpat Visual Components python kód.

6 Analýza s vyhodnocením shody optimálně realizované trajektorie na reálném pracovišti a digitálním dvojčeti

V této kapitole se zabývám implementací optimalizované trajektorie do digitálního dvojčete společně s otestováním této trajektorie a pozorováním chování robotického ramene. Následně je vyexportován vygenerovaný vhodný robotický program do programu v jazyce daného robotického ramene za pomoci volně dostupného Add-onu pro simulační program Visual components. Tento program je následně nahrán do simulačního prostředí Robotstudio, porovnán se simulací ve Visual components a také nahrán do reálného robota s následným otestováním programu a propojením Visual components s reálným robotem, aby Visual Components sledoval hodnoty jednotlivých kloubů a následně tyto hodnoty zapisoval do robotického ramene v simulaci.

6.1 Příklady navrhovaných trajektorií

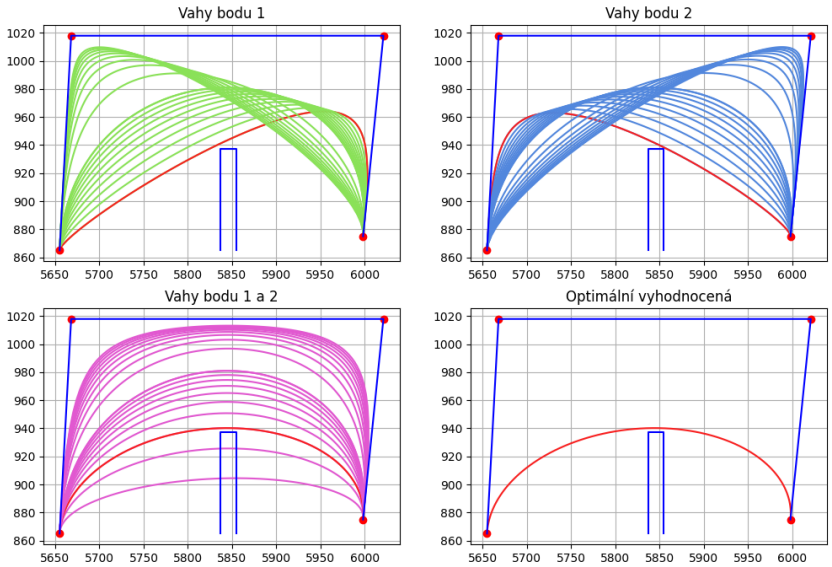
Z důvodu testování jak samotného programu, tak získání více dat pro možnost porovnání rozdílů mezi jednotlivými trajektoriemi jsou navrženy 3 příklady pro vyhodnocení optimalizace trajektorie, prvotní dvě trajektorie jsou si podobné, jelikož obě vycházejí ze stejného počátečního bodu a následně každá směřuje do koncového bodu který je umístěn nad odkladištěm. Třetí trajektorie je testována s jiným než obdélníkovým tvarem překážky, aby se ověřil vytvořený program pro samotný návrh trajektorie.



Obr. 40 Příklady neoptimalizovaných trajektorií

Příklad 1: Trajektorie z boxu nad odkladiště 1

Tato trajektorie byla sestavena tak aby minula okraj krabice s ohledem na co nejkratší trasu. Jsou zde znázorněny jak grafické výběry jednotlivých trajektorií, tak samostatný výpočet nejkratší vhodné trajektorie.



Obr. 41 Trajektorie z boxu nad odkladiště 1

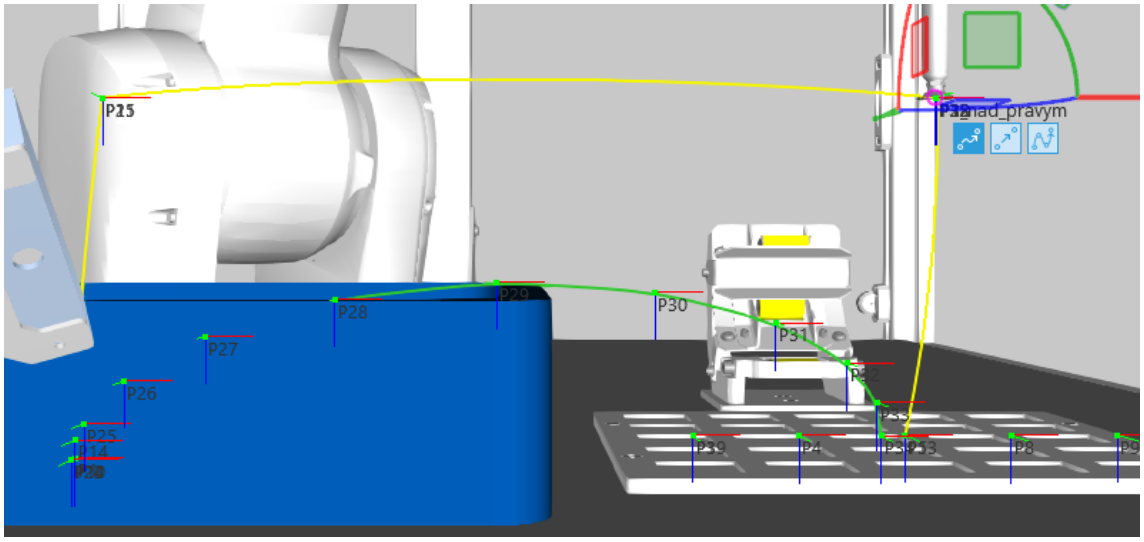
Jak můžeme na obrázku výše vidět, tak se vypočtená trajektorie shoduje s jednou z graficky určených trajektorií, a to s trajektorií která má upravené oba průchozí body. Všechny vhodné trajektorie jsou navíc vyznačeny červenou barvou v daném grafu, aby byly snadno rozpoznatelné. Hodnoty délky těchto trajektorií a váhových úprav jsou uvedeny v tabulce níže. Hodnoty délky byly spočítány jako suma vzdáleností jednotlivých bodů křivek kde každá z křivek obsahuje 100 bodů.

Tabulka 7 Délky a váhy jednotlivých grafických optimálních trajektorií

Graf	Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)
Změna váhy bodu 1	[1, 0.1, 1, 1]	439.78
Změna váhy bodu 2	[1, 1, 0.1, 1]	428.08
Změna váhy bodu 1 a 2	[1, 0.3, 0.3, 1]	397.52
Optimální vyhodnocená	[1, 0.3, 0.3, 1]	397.52
Nejdelší trajektorie	[1,10,10,1]	551.24

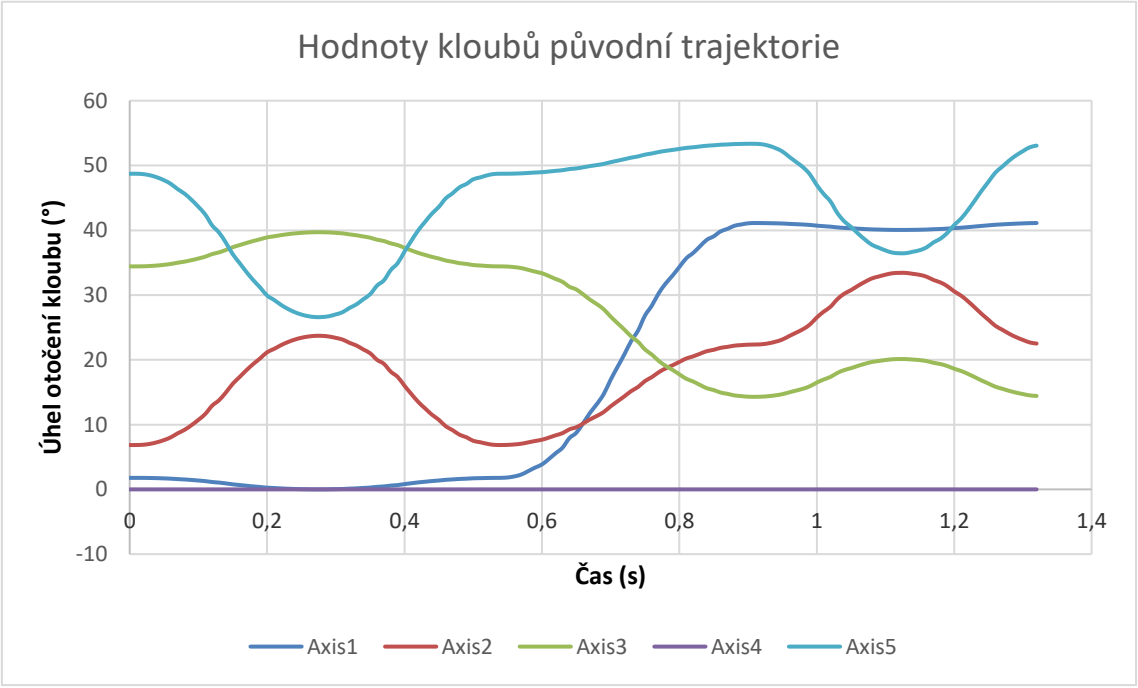
Pro simulaci byla tedy vybrána trajektorie optimální vyhodnocené, která je shodná s trajektorií při změně obou průchozích bodů. Vygeneruji tedy znovu celou trajektorii se stejnými váhami pro jednotlivé body, ale celková trajektorie mi nyní bude obsahovat pouze 11 bodů místo 100. Důvod pro změnu množství těchto bodů se nachází v simulaci, kde každý bod je počítán jako průjezdný

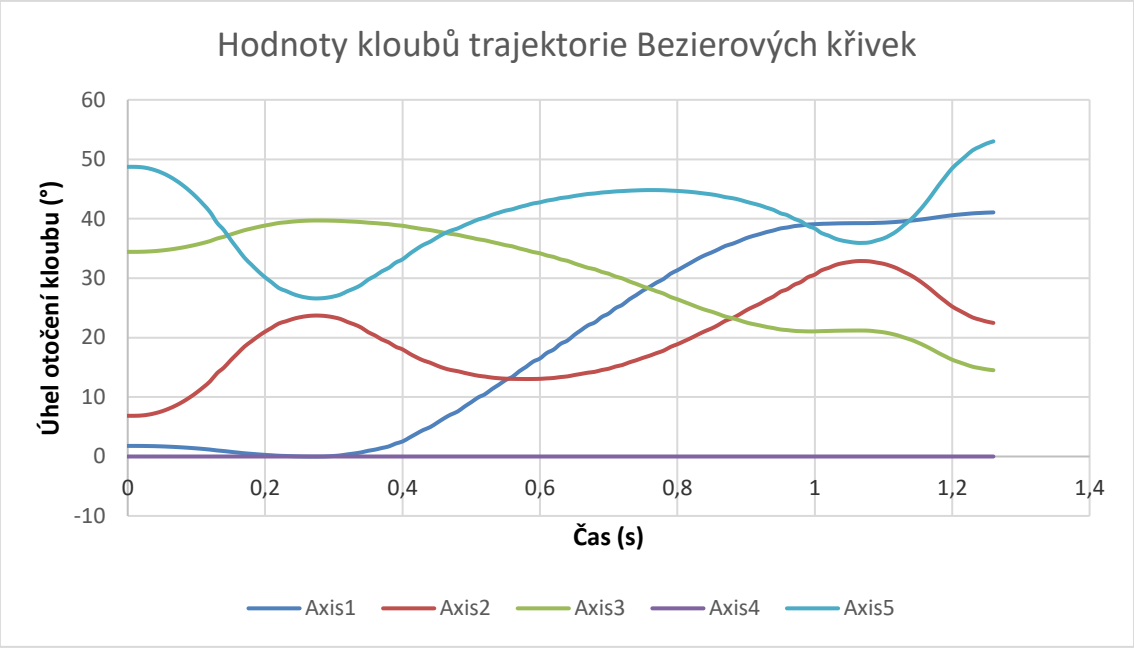
a tímto způsobem by se tedy v každém bodě robot pokusil zastavit a tímto prodloužil dobu průjezdu celou trajektorií. Částečně se tomuto problému vyhnu nastavením průjezdové vzdálenosti na 20 mm pro jednotlivé průjezdové body, tímto způsobem se robot nebude snažit tímto bodem přesně projet, ale pouze se k němu bude přibližovat bez nutnosti se bodu dotknout.



Obr. 42 Zobrazení průjezdné optimální trajektorie v simulaci

Na následujícím obrázku jsou dva grafy, jež jsou data z průjezdu trajektorií na Obr. 42. Při porovnání si můžeme všimnout že kloub 5 osy je vyhlazenější na trajektorii bézierovy křivky oproti původní trajektorii. Podobné vlastnosti můžeme pozorovat i všech ostatních kloubů. Co se týká délky obou trajektorií tak původní trajektorie má délku 651 mm a nová pouze 398 mm.

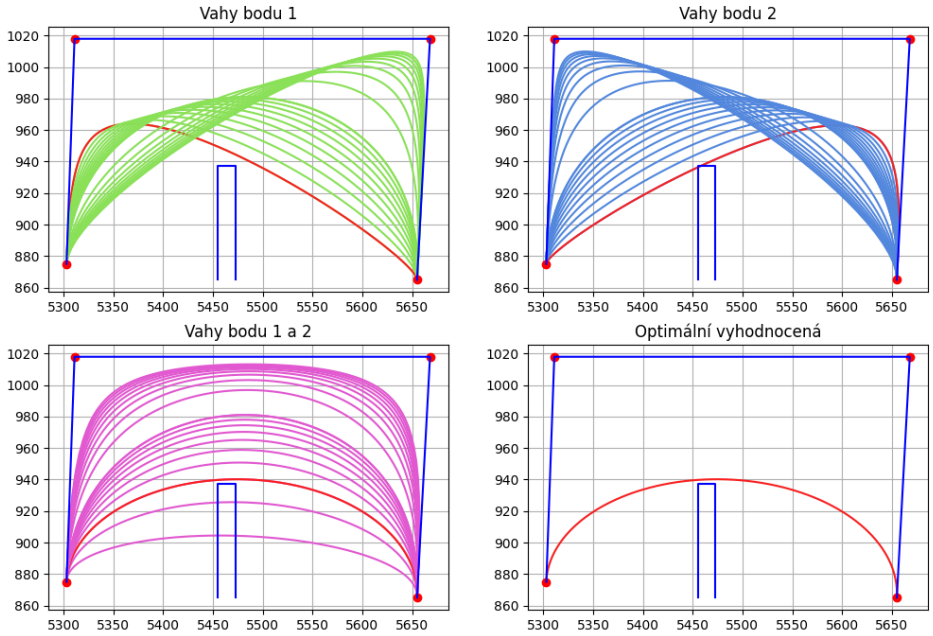




Obr. 43 grafické srovnání kloubových pohybů původní a optimální trajektorie

Příklad 2: Trajektorie z boxu nad odkladiště 2

Tato trajektorie byla sestavena tak aby minula okraj krabice s ohledem na co nejkratší trasu. Jsou zde znázorněny jak grafické výběry jednotlivých trajektorií, tak samostatný výpočet nejkratší vhodné trajektorie.

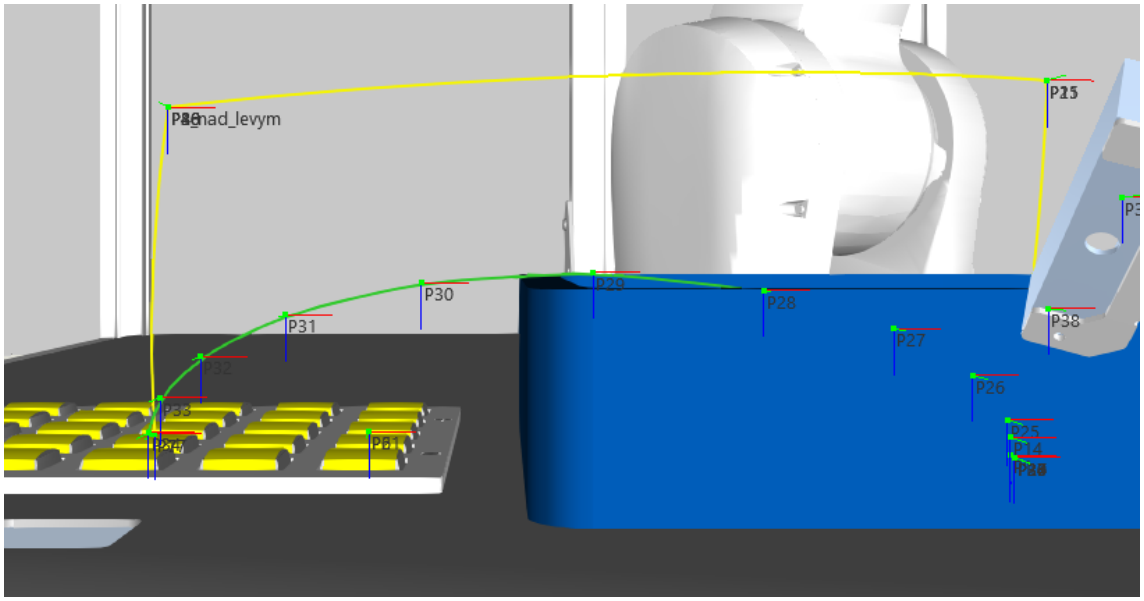


Obr. 44 Trajektorie z odkladiště 2 do boxu

Zde se jako v předchozím případě naše trajektorie shoduje s grafickým určením optimální trajektorie, a to s grafem pro změnu vah bodu 1 a 2. Celková délka nové trajektorie je 405 mm a délka původní trajektorie je 654 mm. Následné data jako je délka optimální zvolené trasy z grafického určení i samostatného výpočtu je níže v tabulce.

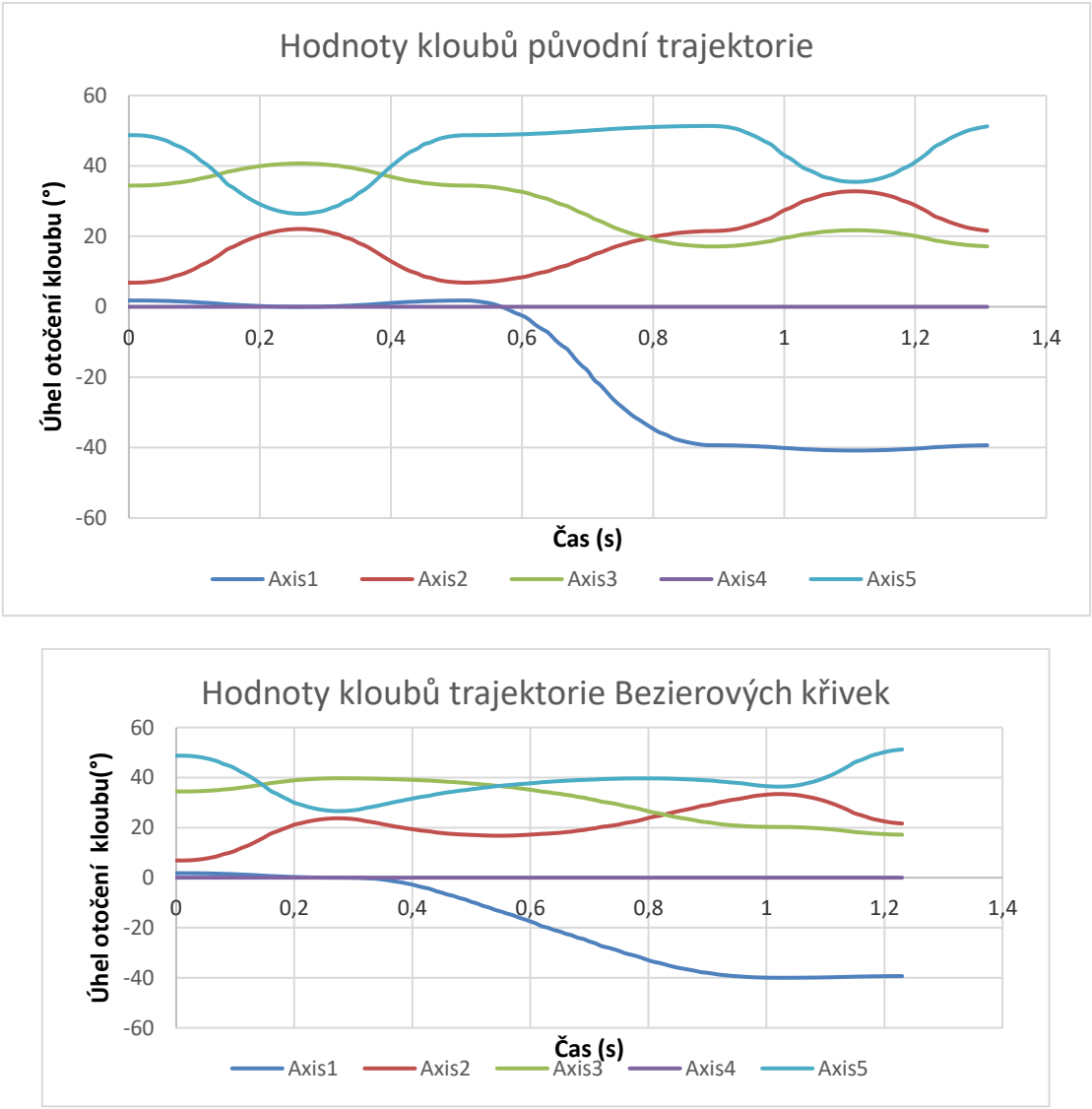
Tabulka 8 Váhy a délky trajektorií prvotního testu

Graf	Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)
Změna váhy bodu 1	[1, 0.1, 1, 1]	446.91
Změna váhy bodu 2	[1, 1, 0.1, 1]	435.01
Změna váhy bodu 1 a 2	[1, 0.3, 0.3, 1]	405.4
Optimální vyhodnocená	[1, 0.3, 0.3, 1]	405.4
Nejdelší trajektorie	[1,10,10,1]	551.24



Obr. 45 Testování navržené optimální trajektorie na levé odkladiště

Na obrázku výše můžeme vidět simulaci pro zaznamenanou původní trajektorii a mnou navrženou optimální trajektorii. Jak je na tomto obrázku i viditelné, nová trajektorie (vyznačena zeleně) se skládá z více průchozích bodů oproti původní. To je z důvodu že simulace si samostatně vypočítává trasy mezi jednotlivými body, a tak jí bylo zadáno více bodů na nové optimální trajektorii s úpravami jednotné konfigurace robotického ramene, zároveň byly všechny průchozí body až na počáteční a koncový nastaveny tak, aby se v nich robot nemusel zastavovat a působily jako průjezdové s nastavenou vzdáleností 20 mm, tímto jsem dosáhl ne velmi výrazného zpomalování v každém bodě.

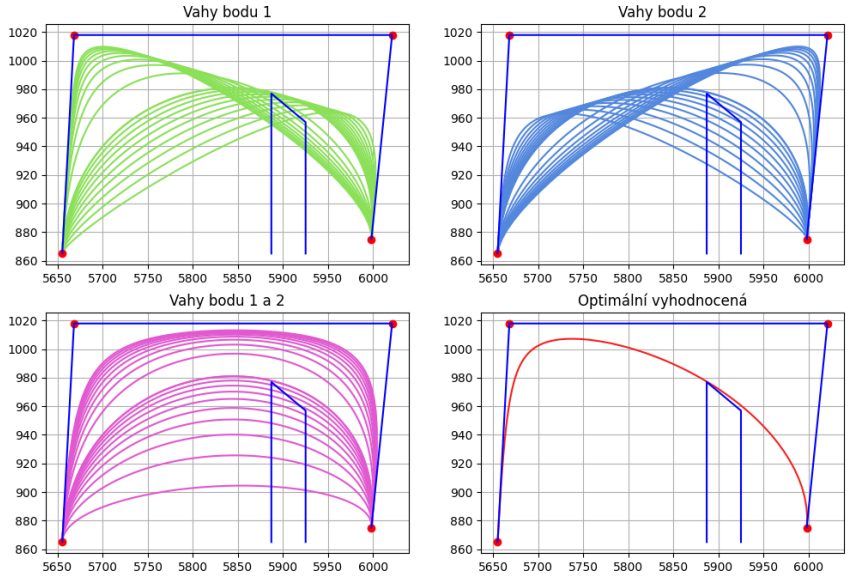


Obr. 46 Grafy kloubů původní a optimální trajektorie

Po vyexportování dat jednotlivých kloubů byl pozorován i čas kdy byla simulace dokončena a je zde vidět rozdíl kdy simulace s původní trajektorií je nejen delší vzdálenostně ale i časově, simulace optimální trajektorie trvá 1.23 sekundy a simulace původní trajektorie trvá 1.31 sekundy, z tohoto je viditelné že optimální trajektorie je nejen kratší ale také i časově méně náročná.

Příklad 3: Trajektorie s neobvyklým objektem mezi boxem a odkladištěm 1

Na tomto příkladu se bude testovat samostatný výpočet optimální trajektorie kde objekt se nenachází poblíž středu grafu a zároveň není nižší než polovina výšky celého polygonu původních bodů. Byla zvolena trajektorie mezi krabicí a odkladištěm 1 do kterého bude tato překážka umístěna, tato překážka má zároveň jednu stěnu vyšší než druhou.



Obr. 47 Grafické vykreslení trajektorií pro neobvyklý tvar objektu

Výsledný samostatný výpočet je vhodný ale po pohledu na grafické zobrazení upravování váhového bodu 1, upravování váhového bodu 2 a upravování váhového bodu 1 a 2 lze vyvodit že trajektorie z těchto grafů by také byly vhodné a dokonce by, co se týká délky trajektorie, mohly být vhodnější. Toto je vhodné otestovat jak výpočtem trajektorie, tak otestováním v simulaci.

Tabulka 9 Váhy a délky optimálních trajektorií

Graf	Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)
Změna váhy bodu 1	[1, 1, 1, 1]	453.38
Změna váhy bodu 2	[1, 1, 1, 1]	453.38
Změna váhy bodu 1 a 2	[1, 1, 1, 1]	453.38
Optimální vyhodnocená	[1, 6.7, 1.7, 1]	504.03
Nejdelší trajektorie	[1,10,10,1]	551.24

Dle vypozorovaného tvaru by se výsledné váhy mohly i lišit a stále být částečně upraveny pro přesnější návrh. Z grafického hlediska byly otestovány možnosti pro růst či pokles váhy o 0.2 na obou průchozích bodech s těmito výsledky.

Tabulka 10 Data z upravované trajektorie

Tabulka pro váhu 0.8 na prvním průchozím bodě		
Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)	Vyhověl/Nevyhověl
[1, 0.8, 0.8, 1]	442.67	Nevyhověl

[1, 0.8, 0.9, 1]	446.2	Nevyhověl
[1, 0.8, 1, 1]	449.54	Nevyhověl
[1, 0.8, 1.1, 1]	452.71	Vyhověl
[1, 0.8, 1.2, 1]	455.69	Vyhověl
Tabulka pro váhu 0.9 na prvním průchozím bodě		
Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)	Vyhověl/ Nevyhověl
[1, 0.9, 0.8, 1]	445.11	Nevyhověl
[1, 0.9, 0.9, 1]	448.4	Nevyhověl
[1, 0. 9, 1, 1]	451.54	Nevyhověl
[1, 0. 9, 1.1, 1]	454.53	Vyhověl
[1, 0. 9, 1.2, 1]	457.36	Vyhověl
Tabulka pro váhu 1 na prvním průchozím bodě		
Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)	Vyhověl/ Nevyhověl
[1, 1, 0.8, 1]	447.42	Nevyhověl
[1, 1, 0.9, 1]	450.52	Nevyhověl
[1, 1, 1, 1]	453.48	Vyhověl
[1, 1, 1.1, 1]	456.31	Vyhověl
[1, 1, 1.2, 1]	459	Vyhověl
Tabulka pro váhu 1.1 na prvním průchozím bodě		
Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)	Vyhověl/ Nevyhověl
[1, 1.1, 0.8, 1]	449.63	Nevyhověl
[1, 1.1, 0.9, 1]	452.54	Nevyhověl
[1, 1.1, 1, 1]	455.34	Vyhověl
[1, 1.1, 1.1, 1]	458.02	Vyhověl
[1, 1.1, 1.2, 1]	460.59	Vyhověl
Tabulka pro váhu 1.2 na prvním průchozím bodě		
Váha – [bod 1,bod 2,bod 3,bod 4]	Délka trajektorie (mm)	Vyhověl/ Nevyhověl
[1, 1.2, 0.8, 1]	451.71	Nevyhověl

[1, 1.2, 0.9, 1]	454.46	Nevyhověl
[1, 1.2, 1, 1]	457.12	Vyhověl
[1, 1.2, 1.1, 1]	459.68	Vyhověl
[1, 1.2, 1.2, 1]	462.13	Vyhověl

Z porovnávaných výsledků v tabulce výše se porovnaly jednotlivé nejkratší vyhovující trajektorie a následná nejlepší vyhodnocená trajektorie má definované váhy pro jednotlivé body s hodnotami [1, 0.8, 1.1, 1] a délkou trajektorie 452.71 cm.

Optimalizační funkce

Tato funkce se nachází v přiložených souborech, samostatně vytváří návrh ve 4 grafu na obrázcích v této kapitole (např. Obr. 47). Ze začátku si tato funkce převezme váhy s nulovou hodnotou v průchozích bodech a následně sleduje dotek vygenerované trajektorie s překážkou, pokud se trajektorie s překážkou střetla na levé straně pak je váha upravena o +0.1 pro průchozí bod nejbližší na ose X po levé straně od této překážky, obdobná funkce je prováděna i pro střet na pravé straně překážky kdy se posílila váha nejbližšímu bodu na pravé straně osy X od překážky. Jakmile se vygenerovaná trajektorie již překážky nedotkne, tak je graficky vymodelována jako optimální trajektorie. Takto funguje první funkce, druhá funguje s principem otestování všech možností, kdy testuje všechny váhy pro jednotlivé průchozí body, a to příkladně způsobem, že pro první průchozí bod je nastavena váha 0.1 a pro druhý průchozí bod jsou následně testovány všechny hodnoty mezi 0.1–10 (mezi 0.1–1 je posun o 0.1 a mezi hodnotami 1–10 je posun o 1).

Zadávání první funkce je jen o vložení původních bodů trajektorie a polohy překážky, návratem z této funkce je list 2D polí s hodnotou osy X a Y.

6.2 Vytváření programu pro reálné pracoviště

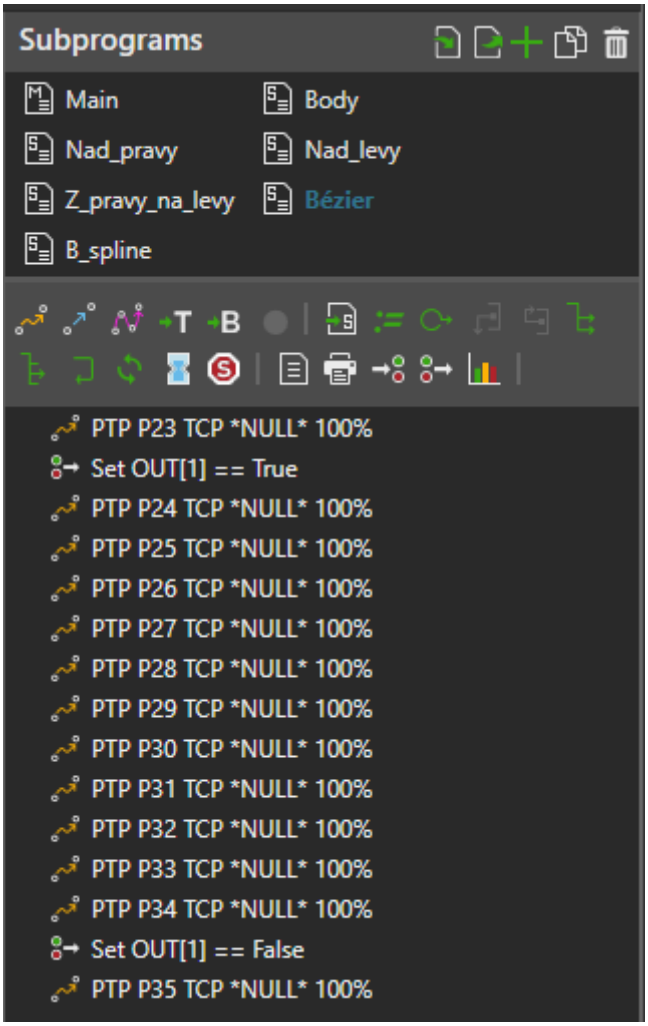
Po exportování jednotlivých bodů optimální trajektorie byla vybrána trasa, na kterou bude aplikována optimální trajektorie a následně vytvořený systematický a reprodukovatelný program.

Byla vybrána trasa mezi vyzvednutím v krabici a odkladem na překladiště 1 za předem připraveného nástroje. Na následujícím obrázku lze vidět využívaný kód který se v simulaci nachází a jeho úkolem je vytvořit dva robotické programy s trajektorií danou daty v excelu.

```
def Aproximace(Routine,data):
    start = Routine.addStatement(VC_STATEMENT_PTPMOTION)
    start.Tool = "TCP"
    position = start.Positions[0]
    position.Configuration = "cfx0"
    position.PositionInWorld = comp.Pos_start

    if Routine.Name == "Bézier":
        Barva = 1
    else:
        Barva = 2

    statement1 = Routine.addStatement(VC_STATEMENT_SETBIN)
    statement1.OutputPort, statement1.OutputValue = Barva, True

    """ Na matrix hodnoty se odkazuju pomocí P.[X,Y,Z] - pozice a
    pomocí PWR.[X,Y,Z] na jejich rotaci """

    xc = [comp.Pos_start.P.X, comp.Pos_upbox.P.X, comp.Pos_up1.P.X, comp.Pos_end1.P.X]
    yc = [comp.Pos_start.P.Z, comp.Pos_upbox.P.Z, comp.Pos_up1.P.Z, comp.Pos_end1.P.Z]



    for i in data:
        Matrix = comp.Matrix
        statement = Routine.addStatement(VC_STATEMENT_PTPMOTION)
        statement.Tool = "TCP"
        if (i in Bezier[1:len(Bezier)-1]) or (i in Bspline[1:len(Bspline)-1]):
            statement.AccuracyValue = 20
        position = statement.Positions[0]
        position.Configuration = "cfx0"
        Matrix.translateRel(i[0], 0, -i[1])
        position.PositionInWorld = Matrix

    statement2 = Routine.addStatement(VC_STATEMENT_SETBIN)
    statement2.OutputPort, statement2.OutputValue = Barva, False

    end = Routine.addStatement(VC_STATEMENT_PTPMOTION)
    end.Tool = "TCP"
    position = end.Positions[0]
    position.Configuration = "cfx0"
    position.PositionInWorld = comp.Pos_up1
```

Obr. 48 Kód využívaný pro generování programu ve VC

Po spuštění tohoto kódu se vygenerují 2 nové podprogramy které obsahují PTP pohyb a zapnutí/vypnutí signálu (tyto signály jsou k zapnutí vykreslování trajektorie kterou vybraný nástroj projede). Zároveň každý z těchto PTP pohybů má předem nadefinovanou konfiguraci, se kterou se má v každém bodě nacházet, a to z důvodu, že pokud by nebyla konfigurace předem dána tak si rameno může hledat i jinou kloubovou konfiguraci a tím nedodržet mnou danou trajektorii. Všechny průjezdové body jsou také nastaveny na předem danou rychlost a průjezdovou vzdálenost. Rychlost je ze začátku testování v simulaci nastavena na 100% rychlosti pohybu ramene a průjezdovou vzdálenost jsem nastavil na 20 mm od bodu, aby nebylo nutné v každém bodě zpomalovat, ale pouze kolem něj projet.



Obr. 49 Vygenerovaný program ve VC

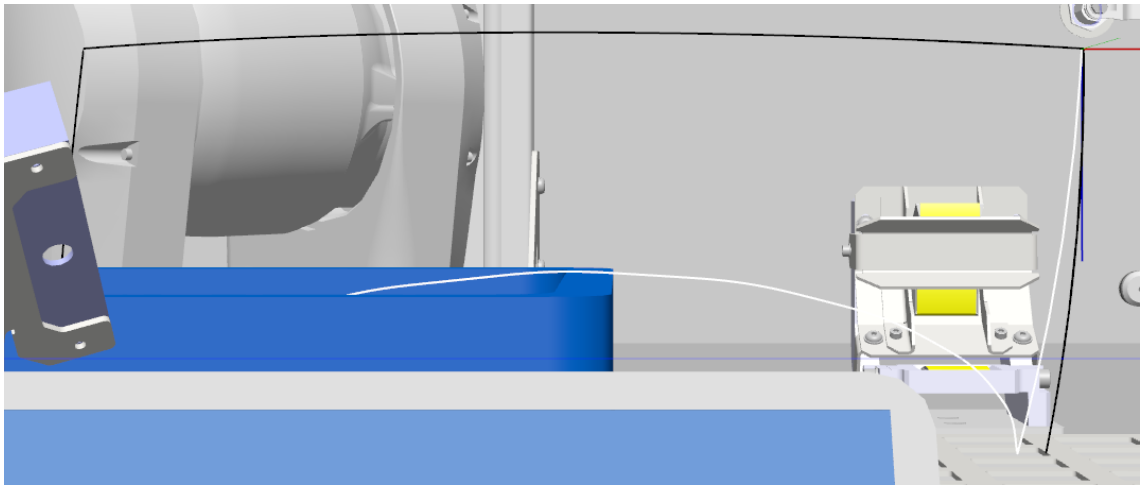
Následný vygenerovaný program jsem ověřil a pomocí předem připraveného a volně dostupného Add-onu jsem tento program převedl do jazyka se kterým je tento robot běžně ovládán.

Při práci v následujícím softwaru Robotstudio bylo zapotřebí vygenerovaný program upravit, původní vysílané signály zde nejsou, a tak jsem tento princip odstranil, jelikož nebyl potřebný. Zároveň jsem upravil rychlosti průjezdů jednotlivých bodů a nastavil sledování časové osy pro jednotlivé podprogramy, aby bylo možné tyto časy porovnat, doba průjezdu jednotlivými programy se následně zobrazí na teach pendantu.

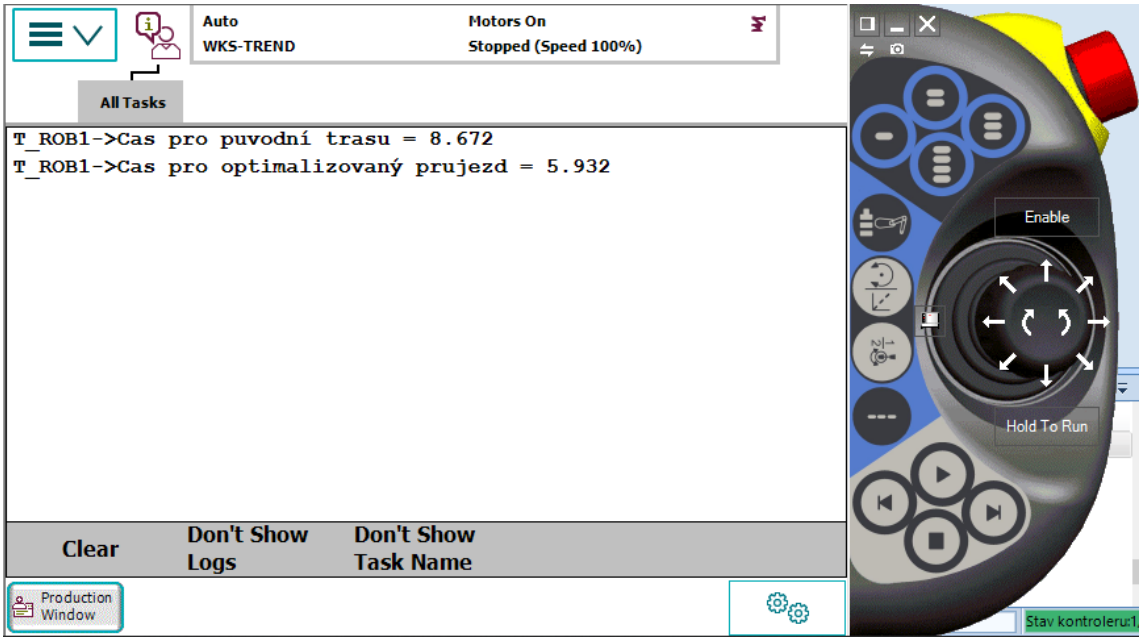
```
PROC main()
    VAR num time1;
    VAR num time2;
    TPErase;
    ClkStart clock1;
    Nad_pravy;
    ClkStop clock1;
    time1:=ClkRead(clock1);
    ClkReset clock1;
    MoveJ P2_4,vmax,fine,TCP;
    ClkStart clock1;
    Bezier;
    ClkStop clock1;
    time2:=ClkRead(clock1);
    TPWrite "Cas pro puvodní trasu = " \Num:=time1;
    TPWrite "Cas pro optimalizovaný prujezd = " \num:=time2;
ENDPROC
```

Obr. 50 Část upraveného programu v Robotstudiu

Na obrázku výše je část kódu v jazyce RAPID, který Robotstudio využívá a je v něm nastaveno vypsání délky průjezdu jednotlivými programy, vypsané časy na virtuální teachpendant je na Obr. 52. Po následném nahrání tohoto programu se očekává vytisknutí stejného textu na reálném robotickém pracovišti.



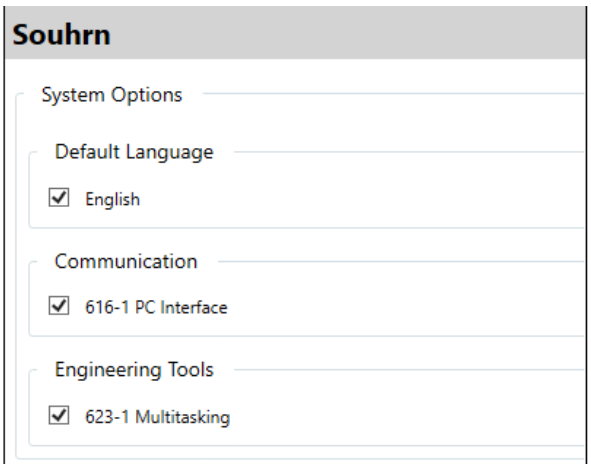
Obr. 51 Nasnímané trasy programu v Robotstudiu



Obr. 52 Simulovaný teachpendant v Robotstudiu

6.3 Vyhodnocení shody na digitálním dvojčeti

Pro propojení Robotstudia s visual components bylo využito UDP socketů kde Robotstudio vysíla údaje o každém z jednotlivých kloubů robota a Visual components je přijímá a předává do simulovaného robota. V robotstudiu byly vytvořeny 2 úlohy jež běží na pozadí a slouží k přijímání a posílání daných socketů, pro možnost mít více než jednu úlohu bylo zapotřebí nastavit do doplňků virtuálnímu kontroléru multitasking. Po přidání dostatečného množství úloh je zapotřebí kontrolér ještě restartovat a následně připravit jednotlivé kódy.



Obr. 53 Nastavení kontroleru robota

Jelikož byly oba softwary na jednom počítači tak se pracovalo na síťové adrese 127.0.0.1. V následujících obrázcích jsou snímky kódů pro odesílání a přijímání zpráv.

```
MODULE Module1

  PERS string robot_ip:="127.0.0.1";
  PERS string client_ip := "127.0.0.1";
  PERS num recv_port:=30002;
  PERS string last_data;
  VAR socketdev recv_socket;

  PROC main()
    VAR string c_ip;
    VAR num c_port;
    VAR string data;

    SocketCreate recv_socket\UDP;
    SocketBind recv_socket,robot_ip,recv_port;

    WHILE TRUE DO

      SocketReceiveFrom recv_socket\Str:=data,c_ip,c_port\Time:=1;
      IF data<>"" THEN
        last_data := data;
        client_ip := c_ip;  !Changes client IP in UDP_SEND task too.
      ENDIF

    ENDWHILE

  ERROR
    SkipWarn;
    data:="";
    TRYNEXT;
  ENDPROC

ENDMODULE
```

```
MODULE Module1

  PERS string client_ip := "127.0.0.1";
  PERS num send_port := 30001;
  VAR socketdev send_socket;
  PERS string di100;

  PROC main()

    client_ip := "127.0.0.1";
    SocketCreate send_socket \UDP;

    WHILE TRUE DO
      SendJointValues;
      WaitTime 0.01;
    ENDWHILE
  ENDPROC

  PROC SendJointValues()
    VAR jointtarget jt;
    VAR string data;

    jt := CJointT();
    data := "JOINTS:";
    data := data + NumToStr(jt.robax.rax_1, 3);
    data := data + "," + NumToStr(jt.robax.rax_2, 3);
    data := data + "," + NumToStr(jt.robax.rax_3, 3);
    data := data + "," + NumToStr(jt.robax.rax_4, 3);
    data := data + "," + NumToStr(jt.robax.rax_5, 3);
    data := data + "," + NumToStr(jt.robax.rax_6, 3);
    SocketSendTo send_socket, client_ip, send_port \Str := data;
  ENDPROC

  PROC SendVar(string id, num value)
    VAR jointtarget jt;
    VAR string data;

    data := id + ":" + NumToStr(value, 0);
    SocketSendTo send_socket, client_ip, send_port \Str := data;
  ENDPROC

ENDMODULE
```

Obr. 54 Ukázky kódů pro posílání a příjem dat

Pro práci se sockety je Robotstudio připraveno a můžu tedy využít už vytvořené funkce.

Následně už stačilo jen vytvořit funkci ve VC která tyto data bude sbírat a převádět hodnoty kloubů do zvoleného robotického ramene. Ve VC se využívá pythonu 2.9 který obsahuje knihovnu pro práci se sockety. Stačilo tedy danou knihovnu naimportovat a zadat správnou IP adresu a port na kterém budou spolu výše zmíněné dva softwary komunikovat. Nastavený port, který byl zodpovědný za příjem dat byl zvolen na hodnotu 30001 a odesílací port bylo zvolen na hodnotu 30002. Správný postup při spuštění je prvotně spustit simulaci ve VC a následně můžeme ať už pohybovat sami s robotem v Robotstudiu anebo mít připravený spustitelný program který se bude kopírovat do simulace ve VC. Ukázku tohoto kódu můžeme vidět na následujícím obrázku Obr. 55.

```python
from vcScript import *
from socket import *

local_ip = '127.0.0.1'
robot_ip = '127.0.0.1'
recv_port = 30001
send_port = 30002
SIZE = 1024

app = getApplication()
comp = getComponent()

def OnStart():
    #Executes before sim clock starts running when play is pressed
    global robot, exe, ctr, inmap, doStates
    global local_ip, robot_ip, recv_port, send_port, my_socket

    #Get associated robot
    robot = app.findComponent(comp.getProperty('RobotName').Value)
    if robot:
        exe = robot.findBehavioursByType(VC_ROBOTEXECUTOR)[0]
        inmap = exe.DigitalInputSignals
        inmap.OnSignalTrigger = writeDI   #Event that writes inputs from sim to robot
        ctr = robot.findBehavioursByType(VC_ROBOTCONTROLLER)[0]

    else:
        exe = None
        ctr = None
        inmap = None
    doStates = {}

    #Connection vars
    local_ip = comp.getProperty('LocalIP').Value
    robot_ip = comp.getProperty('RobotIP').Value
    recv_port = comp.getProperty('RecvPort').Value
    send_port = comp.getProperty('SendPort').Value

    #Init receiving socket
    print ('Receiving packets from local IP {0}, port {1}\n'.format(local_ip, recv_port))
    print ('Sending packets to IP {0}, port {1}\n'.format(robot_ip, send_port))
    my_socket = socket(AF_INET, SOCK_DGRAM)
    my_socket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
    my_socket.bind((local_ip, recv_port))
    my_socket.setblocking(0)


```

Obr. 55 Komunikační kód ve VC

Obr. 56 Reálné pracoviště

7 Zhodnocení dosažených výsledků závěrečné práce

V prvních dvou kapitolách je definován teoretický rozbor práce, který se zabývá rozborem problematiky metod simulace, zároveň zde je popsán způsob a využití simulace společně s ukázanými typy simulací které jsou děleny do softwaru výrobců a třetích stran. Druhá kapitola je zaměřena na rozbor problematiky optimalizace trajektorií s definováním interpolačních a aproximačních metod u kterých bylo definováno několik typů těchto metod a na které se v této práci soustředím a vybírám.

V třetí kapitole, která již patří do praktické části této práce, se pojednává o návrhu metodiky řízení optimální trajektorie, ve které je uvedeno několik diagramů, jakým způsobem se postupuje s praktickou částí této práce a přípravou a vložením daných optimalizačních metod společně se zvoleným hardwarem a softwarem využívaným v této práci.

Čtvrtá kapitola se věnuje simulačnímu prostředí a digitálnímu dvojčeti. Jedná se o nastavení vlastností tohoto dvojčete, umístění v simulačním prostoru, práci s jednotlivými prvky programu a celkovému pohledu na digitální dvojče, jeho funkčnost a ovládání ve vybraném simulačním prostředí.

V páté kapitole se zabývám optimalizací trajektorie dle zvolených kritérií a metod digitálního dvojčete. Do kritérií je zvolena jak celková dráha výsledné trajektorie, tak čas, během kterého se robotické rameno dostane mezi počátečním a koncovým bodem. Výsledkem v tomto srovnání je zjištěno že výsledné optimalizované trajektorie byly vždy kratší jak v délce, tak v časové oblasti. U časové oblasti závisí na průjezdové vzdálenosti a množství průjezdových bodů, pokud není průjezdová vzdálenost nijak nastavena tak se robotické rameno pokouší dostat do přesně definovaného bodu, a tedy zpomalí a v každém takovémto bodě zastaví, což je pro průjezdové body nežádoucí efekt, tento problém je vyřešen nastavením průjezdové vzdálenosti kdy se nesnaží o přesný průjezd ale o průjezd pouze v definované vzdálenosti okolo tohoto bodu.

Kapitola šestá je poslední kapitolou, v této kapitole se už zabývám implementací jednotlivých výsledků do digitálního dvojčete a reálného pracoviště. Digitální dvojče je vytvořeno mezi dvěma systémy, a to Robotstudiem a Visual Components. Tyto dva softwary následně mezi sebou komunikovali pomocí socketů kde Robotstudio posílalo zprávy s hodnotami jednotlivých kloubů. VC si tyto zprávy přebírá a následně rozebere tak aby byl schopný všechny hodnoty správně okamžitě přiřadit pro každý jednotlivý kloub. Zároveň je výsledný program nahrán do reálného robotického pracoviště, na kterém nebyl na oko poznatelný rozdíl oproti simulaci a digitálnímu dvojčeti, program byl také puštěn v manuálním režimu s teachpendantem aby bylo možné program okamžitě zastavit.

V této práci byla snaha o optimalizování původní trajektorie se zadanými body v simulaci třetí strany i s ohledem na možné objekty které se nachází uvnitř pracovní oblasti dané robotické buňky. V simulačních programech od výrobců robotů se často ze zkušenosti již nachází funkce pro optimalizování trajektorií s ohledem na jednotlivé klouby a také objekty v pracovním prostoru které jsou definovány jako nebezpečné zóny. Simulace třetí strany tyto funkce nemá, a proto se

na tuto funkčnost práce částečně soustředila, díky testování a úpravám jsem dosáhl požadované funkčnosti v oblasti 2D prostoru a také hledání trajektorie, při které nedochází ke srážce s objektem. Výsledná trajektorie pak je bodově bez problému importována do simulačního programu, ve kterém je i následně otestována a při chybě či střetu vyhlášena chyba trajektorie. Výsledný program byl následně využit v reálném pracovišti pro ověření využitelnosti.

Použitá literatura a citace

[1] Simulace. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-11-09]. Dostupné z: https://cs.wikipedia.org/wiki/Simulace

[2] Corporate elearning: 6 Reasons Why You Should Use Simulations In Corporation Training. *ELearning Industry* [online]. 2017 [cit. 2021-11-11]. Dostupné z: https://elearningindustry.com/simulations-in-corporation-training-6-reasons-use

[3] Robot programming and simulation software [online]. Octopuz [cit. 2021-1-20]. Dostupné z: https://octopuz.com/

[4] Robot Simulation and Programing [online]. RobotDK [cit. 2020-1-20 Dostupné z: https://robodk.com/

[5] FAMOS robotic [online]. carat robotic innovation [cit. 2020-1-20]. Dostupné z: http://famos-robotic.de/index.php?id=homepos\&L=1

[6] Robotmaster CAD/CAM for robots [online]. [cit. 2020-1-10]. Dostupné z: https://www.robotmaster.com/en

[7] Robot Programming Software - fast,easy automatic: AUTOMAPPPS [online]. Convergent Information Technologies GmbH – Schulstrasse 2 [cit. 2020-1-20]. Dostupné z: http://convergent-it.com/

[8] Verbotics Weld – Verbotics [online]. North Wollongong: Verbotics, 2020 [cit. 2020-1-20]. Dostupné z: https://verbotics.com/

[9] Německo: Siemens [cit. 2020-1-20]. Dostupné z: https://www.plm.automation.siemens.com/global/en/products/tecnomatix/

[10] Tecnomatix [online]. Německo: Siemens [cit. 2020-1-20]. Dostupné z: https://www.plm.automation.siemens.com/global/en/products/tecnomatix/

[11] Robotics and automation Simulation [online]. Německo: Siemens [cit. 2020-1-20]. Dostupné z: https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/robotics-automation-simulation.html

[12] Visual Componnets [online]. Visual Componnets [cit. 2020-1-20] Dostupné z: https://www.visualcomponents.com/products/visual-components/

[13] Teaching Robots with Visual Componnets [online]. Visual Components, 2017 [cit. 2020-1-20]. Dostupné z: https://www.visualcomponents.com/insights/articles/teaching-robots-visual-components/

[14] CIROS Studio for 3D Factory Simulation - [online]. Dortmund: VEROSIM Solution [cit. 2020-1-20]. Dostupné z: https://www.verosim-solutions.com/en/industry/ciros-studio/

[15] Offline Programming with FASTSUITE [online]. FASTSUITE [cit. 2020-1-20]. Dostupné z: https://www.fastsuite.com/en_EN/products/offline-programming.html

[16] MITSI, S., K.-D. BOUZAKIS, G. MANSOUR, D. SAGRIS a G. MALIARIS. Off-line programming of an industrial robot for manufacturing. The International Journal of Advanced Manufacturing Technology. 2005, 26(3), 262-267. ISSN 0268-3768. Dostupné z: doi:10.1007/s00170-003-1728-5

[17] Robotické systémy: Kinematika. Motion [online]. [cit. 2021-12-24]. Dostupné z: http://motion.cs.illinois.edu/RoboticSystems/Kinematics.html

[18] Everything You Need To Know About Robotic Arms. RS-online [online]. [cit. 2021-12-31]. Dostupné z: https://uk.rs-online.com/web/generalDisplay.html?id=ideas-and-advice/robotic-arms-guide

[19] What is a cartesian robot. Linear motion tips [online]. [cit. 2021-12-31]. Dostupné z: https://www.linearmotiontips.com/what-is-a-cartesian-robot/

[20] Cylindrical Robot : Diagram , Construction , Applications. Learnmech [online]. [cit. 2021-12-31]. Dostupné z: https://learnmech.com/cylindrical-robot-diagram-construction-applications/

[21] What is SCARA Robot? ATO [online]. [cit. 2022-01-01]. Dostupné z: https://www.ato.com/what-is-scara-robot

[22] The Effects of Different Robot Trajectories on Situational Awareness in Human-Robot Collaboration. Reserach gate [online]. [cit. 2022-01-11]. Dostupné z: https://www.researchgate.net/publication/339028843_The_Effects_of_Different_Robot_Trajectories_on_Situational_Awareness_in_Human-Robot_Collaboration

[23] Computing the Forward Kinematics of 6DOF Robotic Arm. ResearchGate: Kinematics [online]. [cit. 2022-01-12]. Dostupné z: https://www.researchgate.net/publication/330729384_Computing_the_Forward_Kinematics_of_6DOF_Robotic_Arm

[24] Interpolation example. *Wikimedia* [online]. [cit. 2022-03-09]. Dostupné z: https://commons.wikimedia.org/wiki/File:Interpolation_example_polynomial.svg

[25] Interpolation example linear. *Wikimedia* [online]. [cit. 2022-03-09]. Dostupné z: https://commons.wikimedia.org/wiki/File:Interpolation_example_linear.svg

[26] Kvadratická bézierova křivka. *Wikimedia* [online]. [cit. 2022-03-09]. Dostupné z: https://commons.wikimedia.org/wiki/File:B%C3%A9zier_2_big.svg

[27] Kubická bézierova křivka. *Wikimedia* [online]. [cit. 2022-03-09]. Dostupné z: https://commons.wikimedia.org/wiki/File:B%C3%A9zier_3_big.svg

Seznam příloh

Soubor s modelem Visual Components

Soubor s modelem Robot Studio

Soubor s videi digitálního dvojčete

Soubor s použitými Python kódy