FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Navádění průmyslových robotů s dodatečnými stupni volnosti

Jonáš Somora

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Navádění průmyslových robotů s dodatečnými stupni volnosti

Jonáš Somora

Vedoucí práce
Ing. Ondřej Vaníček

PLZEŇ 2023

Citace v seznamu literatury:
SOMORA, Jonáš. *Navádění průmyslových robotů s dodatečnými stupni volnosti*. Plzeň, 2023. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Ondřej Vaníček.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jonáš SOMORA**
Osobní číslo:	**A19B0321P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Automatické řízení a robotika**
Téma práce:	**Navádění průmyslových robotů s dodatečnými stupni volnosti**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s problematikou modelování sériových manipulátorů.
2. Prostudujte dostupné programovací prostředky pro oblast robotiky a 3D geometrie.
3. Vytvořte simulační model pracoviště s průmyslovým robotem na lineárním pojezdu.
4. Analyzujte problematiku singulárních poloh průmyslového robotu a možnosti jejich vyhýbání s využitím dodatečných stupňů volnosti.
5. Navrhněte algoritmus řízení pohybu zvoleného robotu s dodatečnými stupni volnosti.
6. Implementujte navržený algoritmus a ověřte jej na zadané trajektorii koncového efektoru.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

1. R. P. Paul, 'Robot Manipulators: Mathematics, Programming and Control", MIT Press, 1981.
2. L. Huo, L. Baron, 'The joint limits and singularity avoidance in robotic welding', Industrial Robot: An International Journal, 2008.
3. P. Corke, J. Haviland, 'Not your grandmother's toolbox – the Robotics Toolbox reinvented for Python', International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021.

Vedoucí bakalářské práce: **Ing. Ondřej Vaníček**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2022**
Termín odevzdání bakalářské práce: **22. května 2023**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 17. října 2022

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Plzni dne 22. května 2023

. . . . . .
Jonáš Somora

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato bakalářská práce se zabývá řízením robota umístěném na lineárním pojezdu, pomocí kterého se robot může vyhnout singulárním polohám. Cílem práce je generovat optimální polohy pojezdu tak, aby pohyb robota byl plynulý a bez singulárních poloh. Simulace robota je založena na základě skutečného problému automatického laserového čištění svařovaných dílů. Vzhledem k problému je nutné, aby byl robot vždy přesně kolmo na čištěnou plochu.

Samotné řešení problému se skládá z simulace různých poloh pojezdu pro každý bod procesní trajektorie, výběru optimálních poloh a jejich následné realizaci v simulovaném prostředí. Porovnání jednotlivých poloh je vyhodnocováno na základě determinantu Jacobiho matice.

Abstract

This bachelor's thesis is focused on controlling a robot on linear rail, which the robot can use to avoid singular positions. Aim of this thesis is to generate optimal positions of the rail in a way, in which movement of the robot can be smooth and without singular positions. Simulation of the robot is based on a real issue with automated laser cleaning for welded parts. This goal requires, that the robot is always directly vertical to the cleaned area.

The solution consists of simulating all posible positions of the rail for each point in the process trajectory, choosing the optimal position and then realising the movement in simulated enviroment. Comparation of individual positions is determined with determinant of the Jacobian matrix.

Klíčová slova

Manipulátor • Singulární polohy • Dodatečné stupně volnosti • Python

Poděkování

Rád bych vyjádřil poděkování Ing. Ondřeji Vaníčkovi za jeho vedení a podporu během mé bakalářské práce. Jeho odborné znalosti, profesionální přístup a motivace sehrály klíčovou roli pro úspěšné dokončení této práce. Byl nejen mým vedoucím, ale také inspirací a průvodcem, který mě povzbuzoval k dosažení stanovených cílů.

Obsah

Úvod 1

Automatizace a robotizace průmyslu je trendem poslední doby. Od automatické pásové dopravy přes spolupráce člověk-stroj až po plně automatické výrobní linky. Roboti nahrazují čím dál více i komplexnější a nemonotónní pracovní činnosti. Mezi takovéto činnosti patří i svařování a očišťování svařovaných dílů. Jelikož takovéto obrobky jsou často rozměrné, je nutné pohybovat manipulátorem v prostoru. Tento problém je většinou řešen pomocí lineárního pojezdu, pomocí nějž robot získá další stupeň volnosti, a operátor získá možnost lépe robota navádět prostorem. Zároveň tím vzniká i komplexnější úloha řízení. Krom běžného řízení manipulátoru s šesti stupni volnosti nyní musíme řídit i lineární pojezd. Díky tomu je také možné vyhýbání singulárním polohám.

Singulární polohy robota jsou takové polohy, ve kterých robot přichází o jeden či více stupňů volnosti. Takováto situace většinou nastává, pokud dvě či více pohybových os splynou do jedné. V takovém případě mají shodný směr a robot dočasně přichází o jednu dimenzi pohybu. Problém také nastává v okolí těchto poloh, jelikož je nutné velmi rychle otáčet některými klouby, vznikají vysoké kloubové rychlosti i zrychlení. Takovýto pohyb může v krajních případech poškodit samotný stroj. Je proto žádoucí se těmto polohám vyhýbat a vhodně plánovat trajektorii tak, aby stroj mohl fungovat bez nebezpečně vysokých rychlostí.

Abychom mohli robota navigovat v bezpečných zónách daleko od singulárních poloh, je nutné takovéto zóny detekovat a popsat. Singulární polohy se vyznačují změnou hodnosti jacobiánu. V takovém případě by matice měla determinant roven nule. V okolí singulární polohy je obecně determinant blízký nule, detekce zón blízko singulární polohy lze tedy realizovat pomocí determinantu jacobiánu, je-li menší než předem zvolená hraniční hodnota, považujeme tuto konfiguraci robota za nevhodnou a snažíme se ji vyhnout.

Předmětem praktické části této práce je robotické pracoviště pro laserové čištění, na kterém ZČU provádí vývoj systému navádění robotů na základě 3D skenování. Prezentované modely obrobků byly získány právě z tohoto pracoviště. Rozhraní i datový formát vstupních dat byly navrženy tak, aby bylo možné v budoucnu použít navržený algoritmus v rámci vyvíjeného systému v aplikacích povrchové úpravy.

2 Problematika modelování sériových manipulátorů

Vytváření modelu sériového manipulátoru je komplexní úloha, která byla v průběhu let optimalizována až do podob úmluv či standardizovaných postupů, jak model vytvořit. Tyto standardy zajišťují vyšší čitelnost modelů a v některých případech algoritmizovaný postup, jak výsledný model tvořit. Tato kapitola se zaměřuje na definování souřadných systémů vázaných na konstrukci robota a jejich vzájemné propojení až do výsledné transformační matice, vyjadřující polohu koncového efektoru v závislosti na aktuální poloze jednotlivých kloubů.

2.1 Reprezentace polohy v prostoru

Abychom mohli robota navádět do požadované polohy, musíme být schopni ji vyjádřit. Obecně se poloha skládá ze dvou částí, souřadnice samotného bodu a prostorová orientace, ve které na daný bod ukazujeme. Tyto souřadnice jsou čísla, která sama o sobě nemají jakoukoliv informační hodnotu, je nutné je vždy vztahovat vůči nějakému souřadnému systému. Základním souřadný systém bývá umístěn v nepohyblivé základně robota. Polohu v prostoru tedy popisují dva vektory - translační a rotační. Vektor t vyjadřuje polohu, vektor v vyjadřuje orientaci.

$$t = [x, y, z] \tag{2.1}$$

$$v = [\alpha, \beta, \gamma] \tag{2.2}$$

2.1.1 Reprezentace rotace

Výpočty a manipulace se dvěma vektory je není optimální, pro reprezentaci úhlů se využívá matice rotace, která má oproti reprezentaci jednotlivými úhly výhody z hlediska vlastnosti matic. Tuto matici lze chápat jako matici vektorů jednotlivých os nového s.s.[1] vyjádřeném v s.s. původní matice. $R_2^1 = [x_2^1 y_2^1 z_2^1]$ Z této definice

[1] Nadále bude pojem souřadný systém zkracován do podoby s.s.

vyplývají nutné vlastnosti rotační matice. Ta má vždy rozměry 3×3 a vždy musí být ortonormální, jednotlivé vektory mají jednotkovou normu a jsou vzájemně kolmé. Matice rotace obsahuje tři nezávislé parametry vyjadřující úhlovou polohu vzhledem k jednotlivým osám. Vzhledem k ose rotace se rotační matice vždy liší. Jejich součin je poté výsledná matice rotace. V závislosti na pořadí násobení však získáme dvě rozdílné matice, násobíme-li v pořadí $R = R_x \cdot R_y \cdot R_z$, získáme matici postupné rotace kolem aktuálních os. Opačné pořadí násobení vyjadřuje matici fixní rotace kolem původních os. [2]

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\alpha) & -s(\alpha) \\ 0 & s(\alpha) & c(\alpha) \end{bmatrix}, R_y(\beta) = \begin{bmatrix} c(\beta) & 0 & s(\beta) \\ 0 & 1 & 0 \\ -s(\beta) & 0 & s(\beta) \end{bmatrix}, R_z(\gamma) = \begin{bmatrix} c(\gamma) & -s(\gamma) & 0 \\ s(\gamma) & c(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

Úhly α, β, γ obvykle nazýváme Eulerovy úhly. Vzhledem k způsobu zavedení rotační matice je zřejmé, že lze snadno převést Eulerovy úhly do výsledné matice, a to prostým dosazením. Oproti této dopředné transformaci je její opak, zpětná transformace, značně komplikovanější úlohou. Zpětnou transformaci lze chápat jako soustavu devíti rovnic o třech neznámých.

$$R = \begin{bmatrix} c(\beta)c(\gamma) & -c(\beta)s(\gamma) & s(\beta) \\ s(\alpha)s(\beta)c(\gamma) + c(\alpha)s(\gamma) & -s(\alpha)s(\beta)s(\gamma) + c(\alpha)c(\gamma) & -s(\alpha)c(\beta) \\ -c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) & c(\alpha)s(\beta)s(\gamma) + s(\alpha)c(\gamma) & c(\alpha)(\beta) \end{bmatrix} \quad (2.4)$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2.5)$$

Řešení těchto rovnic závisí na úhlu β, v případech, kdy $\beta = \pm\frac{\pi}{2}$ nastává singularita, při které nelze určit zbylé dva úhly, ale pouze jejich rozdíl či součet. Ve zbylých případech, opět v závislosti na $cos(\beta)$ určíme úhly dle následujících vzorců. $cos(\beta) \geq 0$:

$$\alpha = atan2(-r_{23}, r_{33}) \quad (2.6)$$

$$\beta = atan2(r_{13}, \sqrt{r_{23}^2 + r_{33}^2}) \quad (2.7)$$

$$\gamma = atan2(-r_{12}, r_{11}) \quad (2.8)$$

$cos(\beta) < 0$:

$$\alpha = atan2(r_{23}, -r_{33}) \quad (2.9)$$

$$\beta = atan2(r_{13}, \sqrt{r_{23}^2 + r_{33}^2}) \quad (2.10)$$

$$\gamma = atan2(r_{12}, -r_{11}) \quad (2.11)$$

[2] Pro ušetření místa je funkce sinus zkrácena pouze na s, funkce kosinus na c

Eulerovy úhly, respektive matice rotace není jediná možná reprezentace orientace. Další možná reprezentace je například pomocí obecné osy rotace. Tato neminimální reprezentace obsahuje čtyři parametry, jednotkový vektor $r = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}$ a úhel rotace kolem tohoto vektoru ϑ. Tato reprezentace ovšem opět obsahuje singularitu při zpětné transformaci $R \longrightarrow (r, \vartheta)$, tentokrát pro úhly $\vartheta = 0, \pi$.

Jednotkový kvaternion je neminimální reprezentace, která neobsahuje singulární případy. Tato reprezentace je odvozena z vektoru obecné rotace, úhel ϑ předefinovává do jednotkového vektoru Π.

$$\Pi = \begin{bmatrix} \eta & \epsilon^T \end{bmatrix}^T \tag{2.12}$$

$$\eta = cos(\frac{\vartheta}{2}) \tag{2.13}$$

$$\epsilon = \begin{bmatrix} \epsilon_x & \epsilon_y & \epsilon_z \end{bmatrix}^T = sin(\frac{\vartheta}{2}) \cdot r \tag{2.14}$$

2.1.2 Reprezentace translace

Reprezentace polohy se vždy váže k určitému souřadnému systému. Nevíme-li, v jakém s.s. je poloha vyjádřena, poté není tato informace jakkoliv užitečná. Vždy je proto nutné vyjádřit referenční s.s. Standardní reprezentací je horní index a tato práce tento styl zápisu také používá. Poloha bodu A v prvním s.s. bude tedy vyjádřena A^1, v druhém s.s. A^2. Známe-li polohu bodu v určitém s.s., může být žádoucí tento bod vyjádřit i v jiném. Přechod mezi s.s. popisují dva pohyby. Prvním z nich je translační pohyb, jak se nový s.s. vzdaluje od původního. Označme počátek s.s. vektorem O. Rozdíl nového počátku O_2 a původního O_1 je tedy směrový vektor vyjadřující translaci mezi jednotlivými s.s. Počátek je ve svém s.s. vždy nulový vektor a lze jej tedy z rovnice vynechat.

$$r^1_{1,2} = O^1_2 - O^1_1 = O^1_2 \tag{2.15}$$

Nový souřadný systém tedy máme vyjádřen rotační maticí R^1_2 a vektorem $r^1_{1,2}$. Pomocí těchto dvou informací můžeme libovolný bod vyjádřený v novém s.s. přepočítat do souřadnic původního. Známe-li rotační matici R^2_1, můžeme také provést opačný výpočet. Matici R^2_1 lze také získat inverzí matice R^1_2 či její transpozicí. Vzhledem k vlastnostem rotačních matic jsou totiž tyto operace identické. Tyto vzorce se dají nadále zjednodušit do podoby homogenní transformační matice, tato matice obsahuje veškeré informace o poloze, tedy rotaci i translaci.

$$A^1 = r^1_{1,2} + R^1_2 \cdot A^2 \tag{2.16}$$

$$A^2 = -R^2_1 \cdot r^1_{1,2} + R^2_1 \cdot A^1 \tag{2.17}$$

2.1.3 Homogenní transformační matice

Transformační matice je matice velikosti 4×4, chceme-li pomocí ni vyjádřit souřadnice bodu, musíme tento bod prvně převést do homogenních souřadnic přidáním jedničky na konec vektoru $A = \left[A \; 1\right]^T$. Poté lze souřadnice v jiném s.s. získat prostým vynásobením $A^1 = T_2^1 \cdot A^2$. Opět lze provádět i inverzní výpočet, pomoci inverzní matice $T_1^2 = (T_2^1)^{-1}$ lze získat polohu bodu v novém souřadném systému $A^2 = T_1^2 \cdot A^1$.

$$T_2^1 = \left[\begin{array}{ccc:c} & R_2^1 & & r_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.18}$$

Z vlastnosti násobení homogenních transformačních matic dále plyne možnost snadného sloučení několika postupných transformací do jedné, tato vlastnost je pro robotiku velice důležitá, jelikož jednou z hlavních úloh oboru je právě nalezení transformace mezi souřadným systémem nepohyblivé základny robota a koncovým efektorem, které spojuje kinematický řetězec několika ramen a pohyblivých kloubů. To realizujeme postupnou transformací mezi jednotlivými souřadnými systémy kloubů manipulátoru.

$$T_n^0 = T_1^0 \cdot T_2^1 \cdot \ldots \cdot T_n^{n-1} = \prod_{i=1}^{n} T_i^{i-1} \tag{2.19}$$

Běžnou praxí je také kompenzovat polohu základny a koncového efektoru. Kompenzací polohy základny je myšleno umístění základny robota v prostoru. Koncový efektor také nemá nulovou velikost, a je tedy žádoucí tuto informaci zanést do popisu kinematiky. Obě tyto kompenzace jsou prováděny konstantní transformační maticí.

Určení jednotlivých matic není triviální úlohou, pro její zjednodušení bylo standardizováno několik úmluv pro popis kinematiky manipulátoru, například Denavit-Hartenbergova úmluva, která byla zvolena pro popis manipulátoru v této práci.

2.2 Popis kinematiky manipulátoru

Řízení manipulátoru lze dělit dle známých souřadnic. Obecně jednodušší úlohy, ve kterých známe kloubové souřadnice a hledáme zobecněné souřadnice, tedy souřadnice koncového efektoru v základním souřadném systému. Takovéto úlohy jsou nazývány přímé. Opačné úlohy, tedy nalezení kloubových souřadnic se znalostí požadované polohy koncového efektoru, jsou nazývány inverzní. Dále dělíme úlohy na geometrické, ve kterých hledáme pouze polohové vektory a okamžité kinematické úlohy. Okamžité kinematické úlohy hledají vztahy mezi rychlostí kloubových

souřadnic, případně jejich zrychlení, a rychlostí, respektive zrychlení, koncového efektoru.

2.2.1 Přímá a inverzní geometrická úloha

Geometrické úlohy hledají vztah mezi kloubovými souřadnicemi a zobecněnými souřadnicemi v prostoru. Přímou geometrickou úlohu lze chápat jako zobrazení z prostoru kloubových souřadnic do prostoru zobecněných souřadnic 2.20.

$$X = F(q) \tag{2.20}$$

$$q = \begin{bmatrix} \vartheta_1 & \vartheta_2 & \dots & \vartheta_n \end{bmatrix} \tag{2.21}$$

Vektor q je vektor kloubových souřadnic, v případě rotačního kloubu $\vartheta_i = \theta_i$, tedy aktuální úhel rotačního aktuátoru, běžně vyjádřen v radiánech. Pro translační klouby $\vartheta_i = d_i$, tedy poloha lineárního pístu. Výstupem tohoto zobrazení je poloha koncového efektoru, zpravidla vyjádřena minimální formou, například vektorem $\begin{bmatrix} x & y & z & w & p & r \end{bmatrix}$, kde w, p, r označují Eulerovy úhly. Tato práce simuluje manipulátor s šesti rotačními klouby, vektor q má tedy tvar $q = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}$ a výsledkem tohoto zobrazení bude vždy jedno analytické řešení.

Při návrhu trajektorie manipulátoru je ovšem výsledná poloha známá, zatímco vektor q je neznámý. Tato situace vyžaduje řešení inverzní geometrické úlohy zapsané rovnicí 2.22. Obecně je nalezení inverzní funkce F^{-1} pro sériové manipulátory komplexní úloha, která není vždy řešitelná analyticky.

$$q = F^{-1}(X) \tag{2.22}$$

Řešení inverzní geometrické úlohy pro obecné manipulátory s šesti rotačními klouby bylo popsáno například ve studii [MC94]. Existují však jednodušší postupy pro nalezení inverzní geometrické úlohy pro některé konkrétní architektury manipulátorů. Například pro manipulátory se sférickým zápěstím můžeme úlohu rozdělit na triviální úlohy, které lze řešit analyticky. Tento postup dekompozice na dílčí úlohy byl popsán například v práci doktora Martina Švejdy [Šve11]. V případech, kdy nelze vypočítat analytické řešení, jsou využívány numerické metody, které řešení aproximují.

Manipulátory, které mají více stupňů volnosti, než je nezbytně nutné, jsou nazývány redundantní manipulátory. Pro takové manipulátory zjevně existuje nekonečně mnoho řešení, jelikož získáme více neznámých než rovnic. Některé neznámé tak můžeme považovat za volné parametry a lze je využít pro optimalizaci trajektorie, jednak pro vyhnutí singulárním polohám či minimalizaci rychlosti jednotlivých kloubů. Tyto parametry jsou voleny návrhářem, který si sám vybere které klouby chce předdefinovat. V případě této práce je volen jako volný parametr právě lineární

pojez. Po fixaci volných parametrů je úloha řešena obdobně jako v případě neredundantních manipulátorů. Pro matematickou formulaci úlohy rozdělíme vektor q na q_{par} a q_{var}. Výsledná rovnice bude mít následující tvar.

$$q_{var} = F^{-1}(X, q_{par}) \tag{2.23}$$

Funkci F, vyjadřující závislost zobecněných a kloubových souřadnic, lze chápat jako popis postupného přechodu ze základního souřadného systému na souřadný systém koncového efektoru. $F(q) = T_n^0(q)$ Pro obecný popis transformace v prostoru je zapotřebí šest parametrů, vhodným vymezením způsobu definice sousedních souřadných systémů lze snížit počet těchto parametrů na čtyři. Standardizace popisu kinematiky manipulátorů, primárně s účelem zjednodušení, vedla k postupnému vytvoření několika úmluv, které umožňují algoritmicky definovat celkovou kinematiku. Pro potřeby této práce budeme pracovat pouze s Denavit-Hartenbergovu úmluvou, existují však i další, například Khalil-Kleinfingerova úmluva [KK86]. Ač se K-K úmluva může zdát přirozenější svým umísťováním počátku souřadného systému do kloubu, který způsobuje pohyb daného ramene, je tato metoda méně používaná než právě D-H metoda.

2.2.2 Denavit-Hartenbergova úmluva

Denavit-Hartenbergova úmluva [DH55] je aktuálně nejfrekventovaněji používaná metoda pro popis sériových manipulátorů. Úmluva předpokládá spojení dvou ramen jediným kloubem, který má pouze jeden stupeň volnosti, je proto nevhodná pro popis rozvětvených či paralelních robotů. Samotný algoritmus popisu manipulátoru probíhá ve čtyřech krocích, které postupně definují nový souřadný systém F_i na základě souřadného systému F_{i-1}.

1. Určíme osu z_i jako osu rotace kloubu J_{i+1} v případě rotačního kloubu, v případě translačního kloubu jako osu translace.

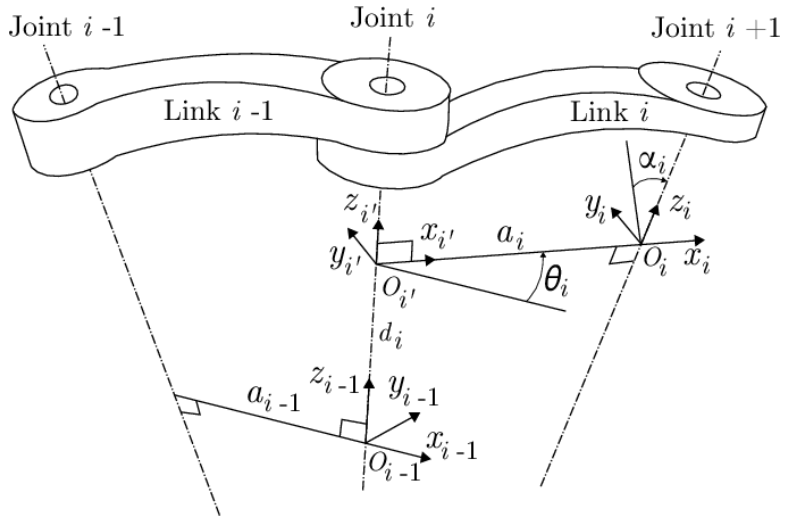
2. Umístíme počátek nového souřadného systému O_i do průsečíku nové osy z_i a nejkratší společné normály os z_{i-1}, z_i.

3. Určíme novou osu x_i ve směru normály definované osami z_{i-1} a z_i, ve směru od kloubu J_i

4. Umístíme osu y tak, aby dotvořila pravotočivý souřadný systém.

Postup je ilustrován na obrázku 2.1. Pomocí nově vytvořeného s.s. poté určíme čtyři parametry D-H úmluvy, které vytváří homogenní transformační matici.

- a_i …kolmá vzdálenost os z_{i-1} a z_i



Obrázek 2.1: Ilustrace postupu pro D-H úmluvu

- α_i …odchylka os z_{i-1} a z_i

- d_i …kolmá vzdálenost os x_{i-1} a x_i

- θ_i …odchylka os x_{i-1} a x_i

Parametr a_i vyjadřuje vzdálenost os z_{i-1} a z_i ve směru osy x_i. Parametr α obsahuje informaci o odchylce os z_{i-1} a z_i kolem osy x_i. Třetí parametr d je vzdálenost os x_{i-1} a x_i ve směru osy z_{i-1}, pro translační klouby je tento parametr proměnný a vyjadřuje pohyb tohoto kloubu. Poslední parametr θ vyjadřuje odchylku os x_{i-1} a x_i kolem osy z_i, pro rotační klouby definuje pohyb daného kloubu, zbylé parametry jsou konstantní a vyjadřují polohu kloubu.

Tento algoritmus lze vyjádřit jako součin čtyř transformačních matic. Vzhledem k vlastnostem homogenních matic obsahujících pouze rotaci, či pouze translaci, lze tyto dvě matice snadno sloučit do jedné s zachováním možnosti vyčíst jednotlivé transformace, viz rovnice 2.25. Výsledná transformační matice T_{i+1}^{i} vyjadřuje souřadný systém následujícího kloubu v závislosti na aktuálním kloubu.

Jakmile získáme transformační matice pro jednotlivé klouby, můžeme je sloučit do výsledné matice pomocí rovnice 2.19. Výsledná homogenní transformační matice T_n^0 vyjadřuje polohu koncového efektoru v souřadném systému základního.

$$T_i^{i-1} = Trans(z, d_i) \cdot Rot(z, \theta_i) \cdot Trans(x, a_i) \cdot Rot(x, \alpha_i) \tag{2.24}$$

$$= \begin{bmatrix} c(\theta_i) & -s(\theta_i) & 0 & 0 \\ s(\theta_i) & c(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & c(\alpha_i) & -s(\alpha_i) & 0 \\ 0 & s(\alpha_i) & c(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

$$= \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\theta_i)c(\alpha_i) & -c(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.26}$$

Existují ovšem případy, pro které D-H úmluva nedefinuje souřadné systémy jednoznačně. Pro základní s.s. lze jednoznačně určit pouze z-ovou osu, ostatní části lze zvolit libovolně. V případě, kdy jsou dvě osy kloubů rovnoběžné, nelze jednoznačně určit počátek nového s.s., jelikož neexistuje jeden průsečík těchto přímek. Nebude existovat žádný, nebo jich je nekonečně mnoho. Při vytváření posledního s.s. není jednoznačně určena osa z, jelikož tento s.s. není propojen s žádným kloubem.

Druhá běžně používaná úmluva, Khalil-Kleinfingerova, vychází právě z D-H úmluvy. Tu modifikuje tak, aby bylo možné ji využít pro paralelní, případně větvené manipulátory. K-K úmluva, stejně jako D-H úmluva, vyžaduje pouze čtyři parametry pro popis sériových manipulátorů. Pro popis rozvětvených manipulátorů již vyžaduje parametrů šest. Další odlišností od D-H úmluvy je umisťování jednotlivých souřadných systémů. Zatímco D-H úmluva počátky situuje na konec ramene, v případě K-K jsou vloženy přímo na rotační osy kloubu. Toto umístění přináší několik výhod, jelikož tímto způsobem lze popisovat právě větvené a paralelní roboty. Navíc je tento způsob přehlednější a lépe čitelný pro komplexnější manipulátory.

2.2.3 Přímá a inverzní okamžitá kinematická úloha

Obdobně jako v případě geometrické úlohy, okamžitá kinematická úloha řeší souvislost mezi kloubovými a zobecněnými souřadnicemi, tato úloha se ovšem zaměřuje na rychlosti a zrychlení. Opět volíme vektor q jako vektor kloubových souřadnic, tentokrát však hledáme časové derivace tohoto vektoru $\dot{q}$ a $\ddot{q}$, taktéž potřebujeme vyjádřit rychlost a zrychlení zobecněných souřadnic, pomocí vektoru X a jeho derivací $\dot{X}$, respektive $\ddot{X}$. Další veličina použitá v následujících vzorcích, $J(q)$ je jakobián.

Jakobián je zobrazení, které získáme parciální derivací funkce $F(q)$, kterou jsme získali v přímé geometrické úloze. Výsledkem přímé kinematické úlohy je vektor rychlostí, respektive zrychlení, zobecněných souřadnic a při výpočtu se počítá přímo s jakobiánem. Pro inverzní úlohu je nutné získat inverzi jakobiánu a výstupem této funkce jsou vektory rychlosti a zrychlení kloubových souřadnic.

$$J(q) = \frac{\partial F(q)}{\partial (q)} \tag{2.27}$$

$$\dot{X} = J(q) \cdot \dot{q} \tag{2.28}$$

$$\ddot{X} = \dot{J}(q, \dot{q}) \cdot \dot{q} + J(q) \cdot \ddot{q} \tag{2.29}$$

$$\dot{q} = J^{-1}(q) \cdot \dot{X} \tag{2.30}$$

$$\ddot{q} = J^{-1}(q) \cdot (\ddot{X} - \dot{J}(q, \dot{q}) \cdot \dot{q}) \tag{2.31}$$

2.2.4 Vlastnosti kinematického jakobiánu

Jakmile jakobián ztratí plnou hodnost, přestane být regulární a matice se stane maticí singulární, což indikuje že manipulátor se dostal do singulární polohy. Singulární poloha je taková konfigurace manipulátoru, ve které manipulátor ztrácí jeden, či více, stupňů volnosti. Taková situace může nastat při splynutí dvou či více os pohybu kloubů do jedné, nebo v případech, kdy se manipulátor ocitne na okraji svého pracovního prostoru.

Problémové nejsou jen samotné singulární polohy, ale i jejich okolí, ve kterém nastávají situace, kdy je nutné pro malou změnu polohy koncového efektoru vykonat neúměrně velkou změnu kloubové souřadnice.

Pomocí SVD rozkladu jakobiánu, rovnice 2.32, získáme dvě unitární matice, složeny z vlastní vektorů a diagonální matici Σ, která obsahuje vlastní čísla matice J. Původní definici kloubových rychlostí dále můžeme vyjádřit pomocí nově vytvořených matic. Ze vztahu 2.35 vyplývá zmiňovaná vlastnost v okolí singulární polohy.

$$J = U \cdot \Sigma \cdot V^T \tag{2.32}$$

$$\Sigma = diag\begin{bmatrix}\sigma_1 & \sigma_2 & \dots & \sigma_n\end{bmatrix} \tag{2.33}$$

$$\dot{Q} = (U\Sigma V^T)^{-1} \cdot \dot{X} = V\Sigma^{-1}U^T \cdot \dot{X} \tag{2.34}$$

$$= V \cdot diag\begin{bmatrix}\sigma_1^{-1} & \sigma_2^{-1} & \dots & \sigma_n^{-1}\end{bmatrix} \cdot U^T \cdot \dot{X} \tag{2.35}$$

Matice, která nemá plnou hodnost se nazývá singulární. Jednou z vlastností singulárních matic je nulový determinant. Této vlastnosti lze využít pro nalezení singulárních poloh manipulátoru, respektive jejich okolí. Matice, které jsou téměř singulární mají determinant téměř nulový.

A-star prohledávací algoritmus

3

Nalezení spojité cesty mezi dvěma body v 2D prostoru je častá úloha při plánování trajektorií. V případě této úlohy reprezentují jednotlivé dimenze různá nastavení lineární dráhy a body procesní trajektorie. Procesní trajektorii manipulátor prochází rychlostí jeden bod za jednotku času, a lze tak tuto dimenzi chápat jako tok času.

Existuje mnoho různých prohledávacích algoritmů, porovnání běžněji používaných algoritmů bylo provedeno například v této studii [CZ22]. Autoři studie popisují několik odlišných algoritmů prohledávání různých komplexností. Jedním z nejběžnějších algoritmů je algoritmus A*, poprvé popsán ve studii [HNR68].

Algoritmus A* s vhodně zvolenou heuristikou nalezne nejkratší cestu mezi dvěma body za relativně krátký čas. Heuristická funkce vyžaduje základní znalosti o daném problému, nesprávně zvolená heuristika může způsobit nalezení neoptimální cesty, nebo zhroucení algoritmu. Základní verze algoritmu A* volí jako heuristickou funkci h vzdálenost do cílového bodu $T = (x_t, y_t)$. Standardně je pro výpočet vzdálenosti volena Euklidovská vzdálenost 3.1, je ovšem pro výpočet možné použít i jiné vzorce, například diagonální vzdálenost 3.2 nebo vzdálenost Manhattanská 3.3, v závislosti na možnostech pohybu v prostoru. Zároveň s heuristickou funkcí je nutné pro každý uzel grafu znát aktuální uraženou vzdálenost g, respektive cenu dosavadní trajektorie. Výsledná rozhodovací funkce $f = g + h$ je součet ceny dosavadní trajektorie a odhadu ceny zbývající cesty.

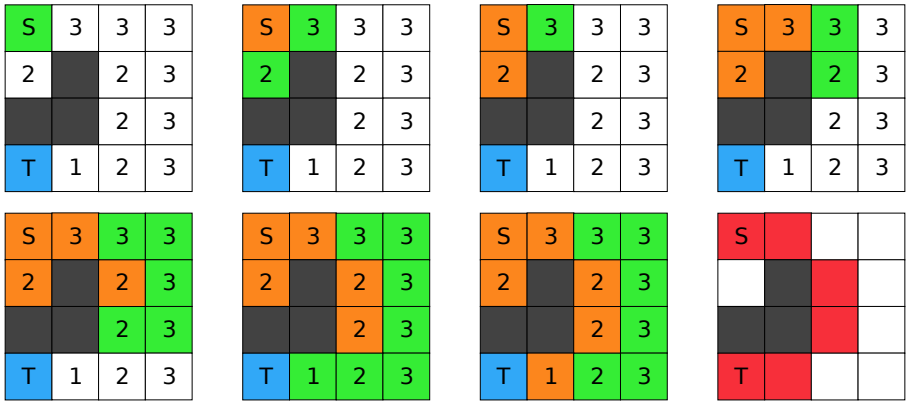
$$h_E = \sqrt{d_x^2 + d_y^2} \tag{3.1}$$

$$h_d = d_x + d_y + (\sqrt{2} - 2) \cdot min(d_x, d_y) \tag{3.2}$$

$$h_M = d_x + d_y \tag{3.3}$$

$$d_x = x - x_t \tag{3.4}$$

$$d_y = y - y_t \tag{3.5}$$



Obrázek 3.1: Modelový příklad algoritmu A*

3.1 Popis průběhu algoritmu

Samotný algoritmus začíná vytvořením dvou množin, první množina je pro otevřené a zatím nenavštívené uzly, druhá množina pro uzly již navštívené. Zvolený počáteční uzel je uložen do otevřené množiny a může začít hlavní smyčka algoritmu, která probíhá, dokud je množina otevřených uzlů neprázdná.

1. Nalezení uzlu s nejmenší hodnotou funkce f a její přesun do množiny zavřených uzlů.
2. Pokud je přesunutý uzel cílový, ukončení algoritmu.
3. Expanze uzlu.
4. Vyhodnocení potomků.

Expanze uzlu spočívá v přidání všech sousedních uzlů, které již nebyly navštíveny, do otevřené množiny. V obecném případě prohledávání grafů je možný pohyb v osmi směrech od uzlu. Pro každý potomek je vyhodnocena rozhodovací funkce f. Existuje-li shodný bod v množině otevřených uzlů, který má vyšší hodnotu funkce f, je nahrazen nově vygenerovaným.

Příklad funkce algoritmu pro vyhledání cesty v 2D prostoru popsán obrázky 3.1. Body označené zelenou barvou jsou aktuálně otevřené, oranžově jsou zabarveny již navštívené body. Výsledná nejkratší cesta je znázorněna červenou barvou.

Programovací prostředky pro řízení a 3D geometrii

4

Realizace řízení manipulátoru může být provedeno pomocí mnoha různých prostředků, od velice primitivních, až po sofistikované nástroje, které uživateli usnadňují práci.

4.1 Zvolené prostředky v prostředí Python

Tato práce se zaměřuje na řízení pomocí Python modulů, jelikož práce se do budoucna může stát součástí většího projektu, který již tyto moduly využívá, bylo by využití odlišných prostředků komplikací pro případné budoucí propojení této práce. Hlavním modulem pro veškeré výpočty s robotem je modul robotics-toolbox. Tento modul nabízí předdefinované manipulátory a metody, které s nimi pracují. Jelikož tato práce simuluje manipulátor neobsažený v základní nabídce této knihovny, bylo nutné jej ručně doplnit. Model simulovaného manipulátoru byl vytvořen ve formě URDF souboru, tento značkovací jazyk se svou strukturou podobá např. XML souborům. V souboru jsou obsaženy informace o jednotlivých osách, typu kloubů a rozměry ramen. Soubor může obsahovat i odkazy na STL soubory s 3D modely jednotlivých ramen pro účely simulací a detekce kolizí. Tento modul dále obsahuje metody pro výpočet přímé i inverzní geometrické úlohy, možnost přidat manipulátoru koncový efektor. Modul robotics-toolbox zahrnuje veškeré metody, které jsou nutné pro práci s manipulátory, je relativně uživatelsky přívětivý. Nemalou výhodou je podrobná dokumentace a kvalitní návody přímo od autorů Jesse Haviland a Peter Corke, díky kterým je snadné se s modulem seznámit a začít naplno využívat jeho potenciál. Je vhodné poznamenat, že tento modul pracuje s metry. Vstupní data jsou s ohledem na simulovanou platformu robotů FANUC generována v milimetrech a je tedy nutné vstupní data přepočítat do správného tvaru.

Veškerou manipulaci s homogenními transformačními maticemi obstarává modul spatialmath. Tento modul umožňuje generování rotačních matic z Eulerových

úhlů zadaných v radiánech či stupních. Tento modul je nadstavbou populárního modulu Numpy, což umožňuje vytváření homogenních transformačních matic i z matic vytvořených právě modulem Numpy. Této symbiózy je využito při načítání plánované trajektorie z csv souboru. Načtená data jsou prvně vytvarována do očekávaného tvaru a vzniklá Numpy matice je poté předefinována na homogenní transformační matici.

Pro vytvoření uživatelského prostředí bylo využito modulu open3d, který obsahuje předdefinovaná prostředí na vizualizaci 3D objektů a možnosti tyto prostředí rozšířit o další funkční bloky. Pomocí nich operátor může načíst 3D model objektu, se kterým chce dále pracovat a předdefinovanou trajektorii. Trajektorie se poté zobrazí na načteném modelu a operátor ji může vyhodnotit, případně nahrát alternativní. Samotná dialogová okna pro načtení souboru jsou vytvářena pomocí modulu tkinter.

Vizualizace výsledné simulace probíhá v prostředí modulu Swift, tento modul nabízí vytváření simulovaného prostředí v reálném čase. Do prostředí je možno přidat další objekty, pro účely této práce byl přidán kvádr simulující lineární pojezd a kužel jakožto koncový efektor manipulátoru.

4.2 Alternativní řídící prostředky

V technické praxi jsou běžně využívány i jiné řídící prostředky, jejich vlastnosti ovšem nebyly pro tento projekt vhodné. Jedním z těchto simulátorů je RoboDK tento software nabízí rozsáhlé možnosti simulace a intuitivní uživatelské rozhraní, neumožňuje ovšem dostatečnou volnost v plánování trajektorie. Další potenciální nevýhodou oproti zvoleným prostředkům je nutnost získání licence, oproti zvoleným prostředkům, které spadají pod opensource licence a jsou tedy bezplatné.

Dalším velice oblíbeným nástrojem je ROS, Robot Operating System. Tento nástroj opět nabízí velké množství možností, jak simulovat manipulátory a navrhovat řízení. Práce v tomto prostředí je ovšem subjektivně neintuitivní a velkou nevýhodou je špatná funkčnost pro Windows uživatele. V neposlední řadě toto prostředí také nenabízí takové možnosti inteligentního řízení, jako zvolené nástroje v prostředí Pythonu.

V neposlední řadě se nabízí využít knihovnu robotics-toolbox v prostředí Matlabu [Cor96]. Matlab je prostředí velice vhodné pro matematické výpočty a úlohy, umožňuje vytvářet simulace prostředí Simulink a nabízí obdobnou uživatelskou volnost jako prostředí Python, využití tohoto prostředí ovšem nebylo zvoleno, vzhledem k již zmíněné možnosti práci propojit s větším projektem. Navíc toto prostředí opět vyžaduje vlastnění licence.

5 Navržený algoritmus řízení manipulátoru s přidaným stupněm volnosti

Proces nalezení trajektorie lineárního pojezdu s cílem vyhnutí se singulárním polohám je realizován v několika krocích, které jsou zakresleny v procesním diagramu 5.1.

1. Načítání vstupních dat je realizováno pomocí uživatelského rozhraní, vstupní data obsahují procesní trajektorii, homogenní transformační matici modelu a 3D model obrobku.

2. Vygenerování procesní trajektorie jako seznam homogenních transformačních matic v základním souřadném systému.

3. Simulace bodů trajektorie pro jednotlivé polohy lineární dráhy, pro každou pozici manipulátoru je vyhodnocen determinant jakobiánu a uložen pro další zpracování.

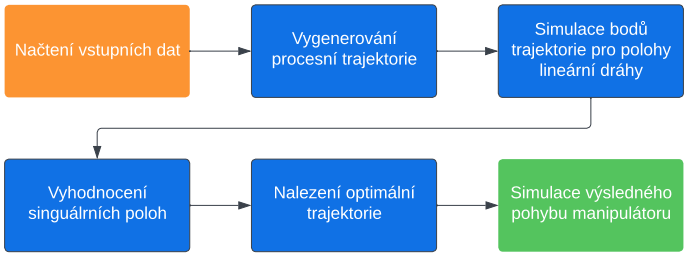
4. Vyhodnocení singulárních poloh a jejich okolí na základě determinantu jakobiánu. Na základě těchto dat je vytvořena mapa potenciálních trajektorií.

5. Nalezení optimální trajektorie pomocí prohledávacího algoritmu A*.

6. Grafická simulace výsledného pohybu manipulátoru, vyhodnocení rychlostí kloubových souřadnic.

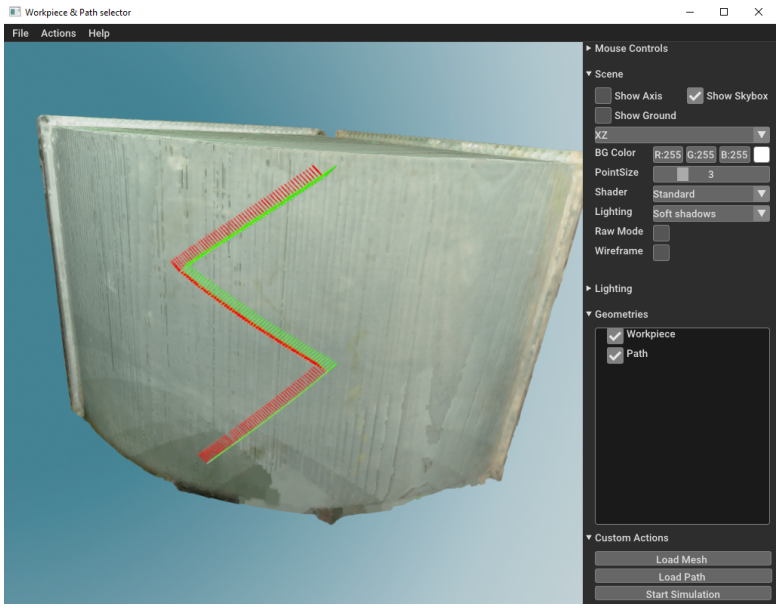
5.1 Načtení a zpracování vstupních dat

Pro reálné využití programu je žádoucí vytvořit uživatelské rozhraní, které umožňuje načtení vstupních souborů bez nutnosti práce s terminálem či upravování zdro-



Obrázek 5.1: Diagram návrhu trajektorie

jového kódu. Pomocí knihovny open3d bylo vytvořeno primitivní prostředí s přidanými tlačítky pro načítání jednotlivých souborů a spouštěním simulace. První tlačítko pro nahrání modelu ve formátech STL či PLY, druhé pro nahrání CSV souboru s trajektorií. Posledním tlačítkem se spustí simulace a zavře uživatelské rozhraní. Spuštěním simulace program zároveň vygeneruje STL soubor s modelem opracovaného dílu, pokud již tento soubor neexistuje. STL model je potřebný pro vizualizaci v prostředí Swift, které s PLY soubory neumí pracovat. Při testování bylo rozhraní optimalizováno pro možnost změny nahraného souboru, pro případy kdy operátor vyhodnotí nahranou trajektorii jako nevhodnou.



Obrázek 5.2: Uživatelské rozhraní s vloženým modelem a naplánovanou trajektorií

Vstupní data jsou rozdělena do několika souborů, hlavním z nich je CSV soubor obsahující informace o požadované poloze koncového efektoru, včetně rotace.

id	laser on	x	y	z	w	p	r	e1	a1	a2
0	1	-197,79	-196,81	248,14	88,97	-42,24	143,01	0	0	0
0	1	-195,11	-198,84	251,84	90,26	-42,24	142,89	0	0	0
0	1	-192,42	-200,87	255,54	89,62	-42,23	143,41	0	0	0
0	1	-189,74	-202,91	259,24	89,64	-42,24	143,28	0	0	0

Tabulka 5.1: Ukázka několika řádek vstupního CSV souboru

Tento soubor tvoří jedenáct sloupců. První sloupec obsahuje informaci o ID dané cesty, druhý sloupec říká zda má, či nemá, být zapnuta procesní technologie. Následující tři sloupce obsahují informaci o poloze koncového efektoru ve formátu x, y, z, vyjadřující translační část. Následováno sloupci w, p, r, které obsahují Eulerovské úhly pro schéma rotace ZYX. Poloha lineárního pojezdu je obsažena v devátém sloupci. Další sloupce obsahují informace potřebné pro reálné stanoviště manipulátoru, ovšem pro zjednodušený model simulovaný v této práci jsou zanedbatelné, pro úplnost si je přesto zmíníme. Dva sloupce obsahují polohu rotačních kloubů polohovadla, na kterém je umístěn opracovávaný díl. Ukázka tohoto souboru se nachází v tabulce 5.1.

Z tohoto souboru jsou potřebné pouze sloupce s informací o poloze a orientaci koncového efektoru. Ty jsou převedeny na homogenní transformační matici a vzhledem k různým jednotkám vstupních dat a knihovny robotics-toolbox převedeny z milimetrů do metrů.

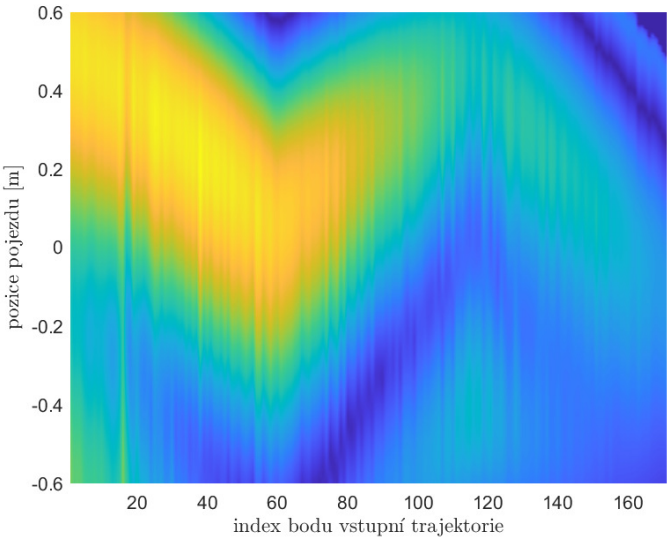
Dalšími vstupními soubory jsou pomocné 3D modely určené pouze pro simulační účely. Prvním z nich je kužel simulující koncový efektor a kolejnice, která nahrazuje lineární pojezd robota. Poslední soubor, který program vyžaduje pro správný běh, je YML soubor obsahující homogenní transformační matici pro převod bodů trajektorie z souřadného systému obrobku do základního s.s. Soubor s touto maticí musí být umístěn ve stejné složce, jako samotný model obrobku.

5.2 Řízení lineárního pojezdu

Samotné řízení, respektive hledání optimální trajektorie je posuzováno na základě absolutní hodnoty determinantu Jakobiánu. Tato hodnota je indikátorem, zda je manipulátor v okolí singulární polohy. Zvolíme proto vhodnou minimální hodnotu, a všechny determinanty menší než tato hodnota označíme za nevhodné polohy, kterým se chceme vyhnout.

5.3 Simulace možných trajektorií

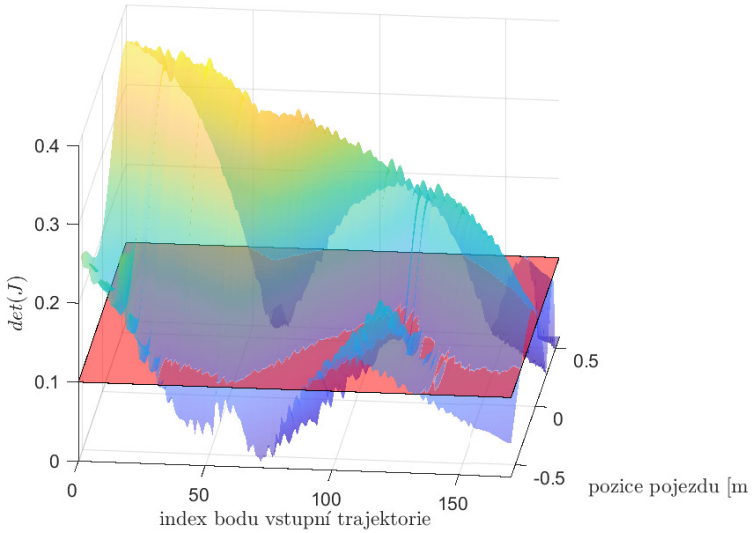
Abychom mohli získat zmíněný determinant, musíme prvně nasimulovat veškeré polohy manipulátoru, do kterých se může dostat. Robota tedy postupně posouváme po lineárním pojezdu a pro každou polohu se pokusíme projít kompletní trajektorii. V případě, že se není možné nasimulovat jednu z poloh pro dané nastavení pojezdu, je tato poloha předčasně ukončena a veškeré chybějící hodnoty nahrazeny nulou. Tento proces může být velice zdlouhavý a pro případy opakovaného simulování identické trajektorie je nežádoucí jej opakovat. Po vygenerování kompletní mapy determinantů jsou veškerá data uložena do CSV souboru a při případném dalším spuštění se program prvně pokusí tato data načíst a v případě neúspěchu je začne generovat. Příklad výsledné mapy determinantů lze vidět na obrázku 5.3.



Obrázek 5.3: Hodnoty determinantů jakobiánů pro všechny generované pozice

5.4 Vyhodnocení oblastí blízkých singulárním polohám

Na základě vytvořené mapy determinantů byla následně vytvořena mapa singulárních poloh, respektive jejich okolí. Hraniční hodnota determinantu byla zvolena tak, aby byl výsledný pohyb manipulátoru dostatečně klidný, ale zároveň tak, aby nevznikaly zbytečně velké zakázané zóny. Vykreslení jednotlivých hodnot determinantu s hraniční rovinou lze vidět na obrázku 5.4. Hodnoty pod rovinou jsou vyhodnoceny jako okolí singulárních poloh a algoritmus vyhledávání trajektorie nesmí vytvořit cestu skrze tyto zóny.



Obrázek 5.4: Hodnoty determinantů jakobiánů pro všechny generované pozice

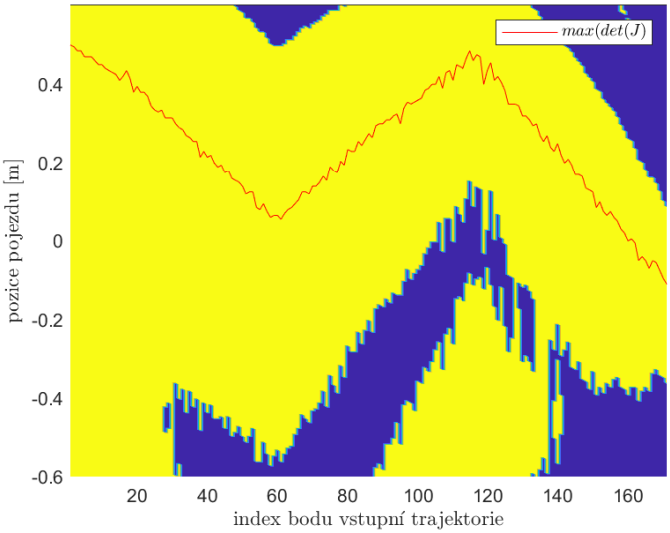
5.5 Generování optimální trajektorie

Optimální trajektorie ve smyslu této úlohy je taková trajektorie, ve které se robot nedostává do okolí singulární polohy a zároveň je pohyb lineárního pojezdu minimální a plynulý. Každá manipulace s robotem vytváří potenciální kmity koncového efektoru a tedy nepřesné výsledky.

Prvotní pokus o nalezení takové trajektorie splňoval pouze první požadavek. V tomto případě byla výsledná poloha lineárního pojezdu taková poloha, ve které byl největší. Ač takováto trajektorie plně splnila první požadavek, nebyla plynulá a mezi jednotlivými body vznikaly velké skoky v poloze pojezdu, jak lze vidět na obrázku 5.5. Od této metody se proto rychle upustilo.

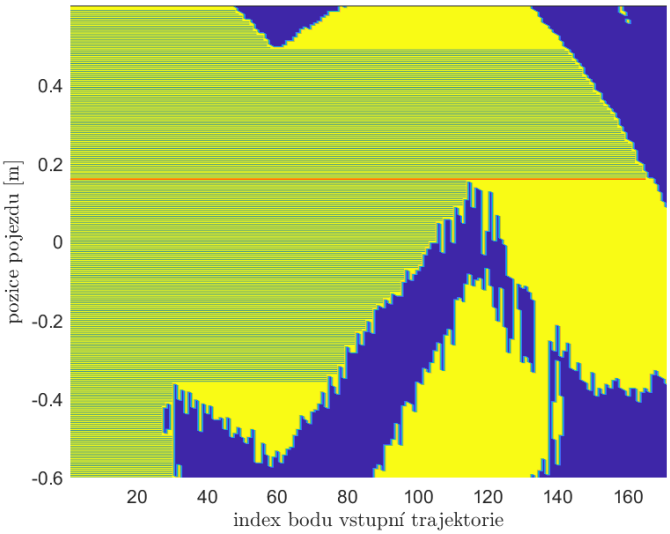
Druhá metoda, která již byla úspěšná pro oba požadavky, je založena na algoritmech prohledávání cest. Jelikož pro každý bod cílové trajektorie generujeme n poloh lineárního pojezdu, a každou z těchto poloh ohodnocujeme na základě hodnoty determinantu jakobiánu pro dané nastavení, vzniká tak mapa s překážkami, skrze kterou hledáme cestu. Hledání optimální trajektorie je rozděleno do tří částí. V první části nalezneme počáteční bod trajektorie, poté je nutné určit cílový bod a posledním krokem je nalézt vhodnou cestu mezi nimi.

Algoritmus pro vyhledání počátečního bodu byl volen na základě požadavku pro minimalizaci pohybu lineárního pojezdu. Algoritmus postupně prochází veškeré možné polohy pojezdu a pro každé nastavení měří maximální možnou dobu strávenou v této poloze, tedy jak dlouho je možné robota ponechat na stejném místě. Tyto údaje se ukládají do pole a index maximálního prvku tohoto pole je právě po-



Obrázek 5.5: Prvotní nalezená trajektorie na základě maximálních hodnot determinantu

čáteční bod. Tento algoritmus byl vizualizován do obrázku 5.6



Obrázek 5.6: Vizualizace algoritmu pro nalezení počátečního bodu

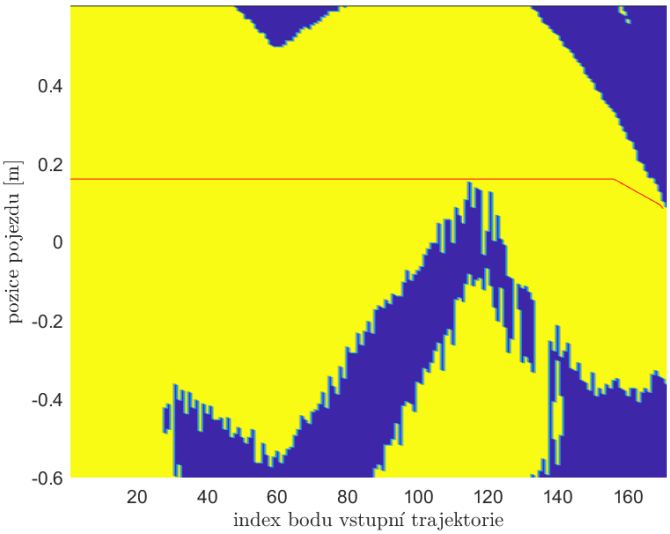
Požadavek na cílovou polohu by mohl být stejné nastavení lineárního pojezdu jako v prvním kroku, v případě že by se jednalo o cyklický děj a manipulátor by opakovaně konal stejný proces. Tato situace ovšem nenastává a není proto vhodné cílovou polohu definovat. Vzhledem k faktu, že v každém kroku algoritmu probíhá

posun v x-ové ose a cíl algoritmu je projít veškeré požadované body s minimálním pohybem lineárního pojezdu, je cílem celá přímka pro poslední časový okamžik.

Vytvoření spojité trajektorie mezi těmito body bylo realizováno pomocí A* prohledávaní, představeného v kapitole 3. Algoritmus, respektive heuristickou funkci, bylo nutné upravit pro tuto úlohu. Vzhledem k neexistenci cílového bodu, ale pouze snahou dokončit trajektorii, byla použita vzdálenost pouze souřadnic osy x $h = |x - x_t|$. Navíc je nutné hlídat, zda se právě vytvářený uzel nepohybuje v blízkosti singulárních poloh. Je-li tomu tak, uzel není vygenerován.

Expanze uzlu navíc v této úloze musí být také modifikována. Jak již bylo zmíněno, osu x lze v této úloze chápat jako osu času, expanze uzlu tedy vytváří potomky pouze v kladném směru osy x, a to pouze o jeden krok. Pohyb v y-ové ose symbolizuje pohyb lineárního pojezdu, je tedy nutné rozhodnout jaký maximální pohyb lze za jeden časový okamžik realizovat. Lineární pojezd simulujeme s krokem $5mm$ a maximální pohyb byl zvolen na dvojnásobek této hodnoty. Při expanzi uzlu tedy vznikne pět potomků

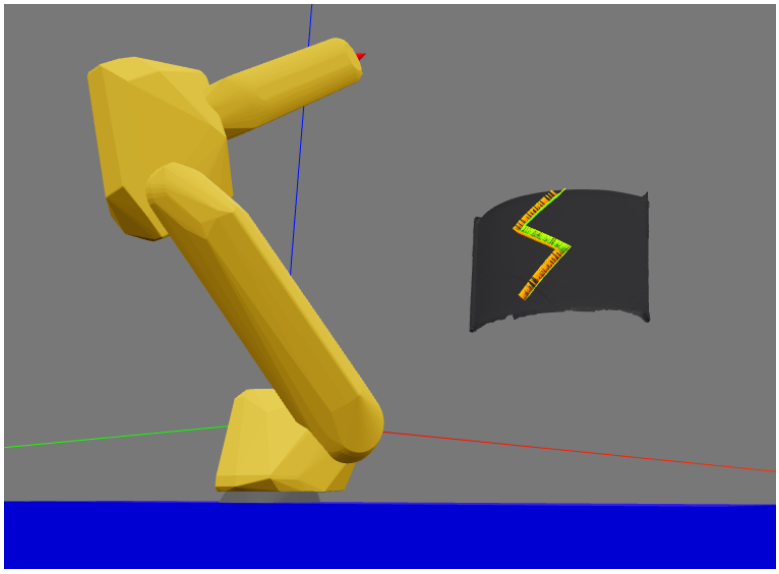
Pomocí nalezené trajektorie je vygenerována B-spline křivka, která trajektorii aproximuje. Tato aproximace má spojitý průběh, díky kterému je výsledný pohyb lineárního pojezdu plynulejší a nevznikají skoky. Výsledná trajektorie plánované trasy z obrázku 5.2 lze vidět v grafu 5.7



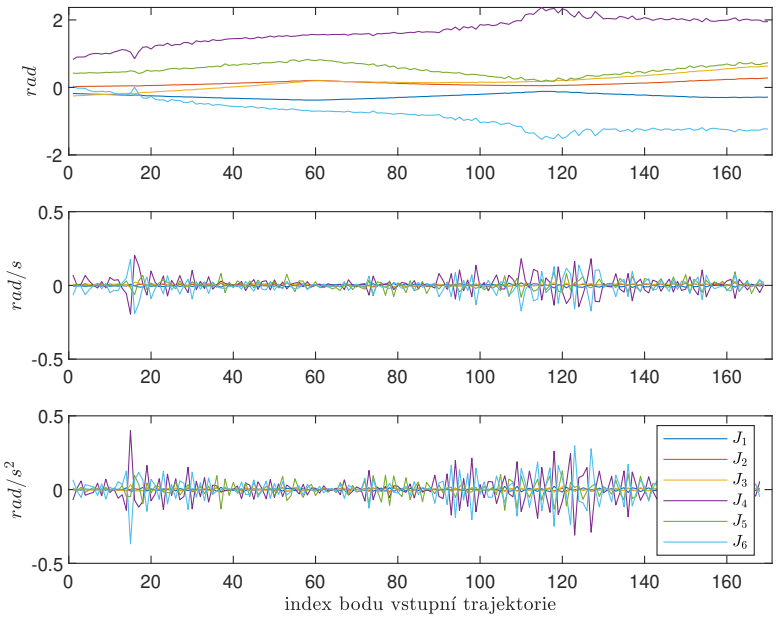
Obrázek 5.7: Vygenerovaná trajektorie pomocí A* algoritmu

5.6 Průchod optimální trajektorie

Výsledná trajektorie je poté simulována v prostředí Swift. Do simulovaného světa zobrazíme manipulátor, lineární pojezd a model obrobku. Pro lepší vizualizaci byl přidán koncový efektor v podobě červeného kužele. Prostředí Swift lze vidět na obrázku 5.8 Po provedení simulace je možné vyhodnotit pohyb manipulátoru z hlediska poloh , rychlostí a zrychlení jednotlivých kloubů, které byly vykresleny do grafu 5.9. Z grafů lze vidět, že veškeré kloubové rychlosti jsou v rozumných mezích. Při návrhu není brána v potaz procesní rychlost koncového efektoru, Nelze se tedy bavit o hodnotách jako takových. Průběh rychlostí i zrychlení je však rozumně hladký a nevznikají velké skoky mezi jednotlivými body procesní trajektorie. To je dostatečným indikátorem, který dokazuje že manipulátor se nepohybuje v okolí singulárních poloh.



Obrázek 5.8: Ukázka simulačního prostředí Swift



Obrázek 5.9: Polohy jednotlivých kloubů a jejich derivace

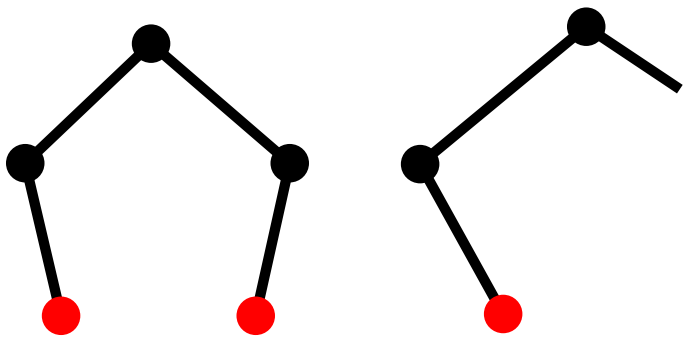
Simulační ověření navrženého algoritmu — 6

6.1 Popis robota a jeho simulačního modelu

Standardně se v průmyslu využívají manipulátory, které lze dělit do dvou kategorií na základě jejich konfigurace.

Sériové manipulátory jsou manipulátory s otevřeným kinematickým řetězcem, to znamená že každé rameno, které není základnou robota ani koncový efektor, je spojeno s právě dvěma dalšími rameny. Jednotlivé aktuátory jsou přímo na ramenu manipulátora a zvyšují tím celkovou hmotnost pohyblivé části robota. Výhodou takovéto konstrukce je velký pracovní prostor a flexibilita.

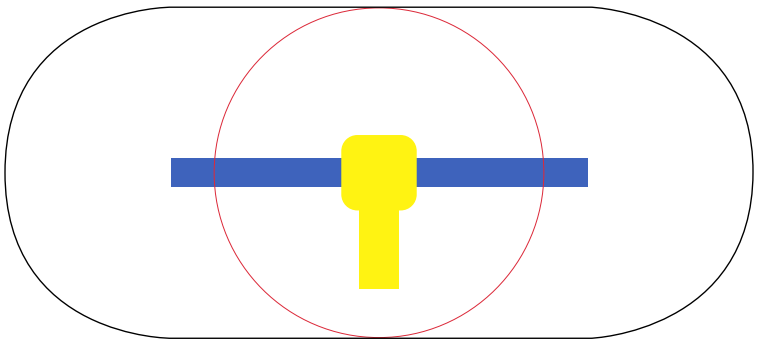
Do druhé kategorie paralelních manipulátorů patří ti roboti, které mají uzavřený kinematický řetězec. Koncový efektor je se základnou spojen několika různými rameny a existuje více základních ramen. Hlavní výhodou paralelních manipulátorů je možnost umístění aktuátorů k základně robota, díky čemuž je celková konstrukce lehčí a dynamičtější. Vzhledem k komplexnější konstrukci mají paralelní manipulátory obecně menší pracovní prostor.



Obrázek 6.1: Ukázka paralelního a sériového manipulátoru

6.1.1 Simulované robotické pracoviště

Manipulátor simulovaný v této práci patří do kategorie sériových manipulátorů. Fanuc M20iA/20M, robot v reálném pracovišti a zároveň simulovaný robot, je šestiosý manipulátor s maximální zátěží na zápěstí dvacet kilogramů. Takto vysoká nosnost je nutný parametr vzhledem k hmotnosti 3D skeneru, laseru a pomocných zařízení, která jsou umístěna jako koncový efektor. Účelem pracoviště je možnost 3D skenování objektů a jejich následné povrchové opracování. V pracovišti je plánována práce s rozměrnějšími obrobky. Robot je proto umístěn na lineárním pojezdu, který se může pohybovat, z pohledu základního souřadného systému, v ose y. Tím se násobně zvětší pracovní prostor manipulátoru a přidaný stupeň volnosti zároveň umožňuje vyhýbání singulárním polohám, ilustrováno na obrázku 6.2. Součástí pracoviště je také dvouosé robotické polohovadlo, pomocí kterého robot získá další stupně volnosti. V rámci této práce bylo polohovadlo zanedbáno. Kompletní pracoviště je zobrazeno na obrázku 6.4.

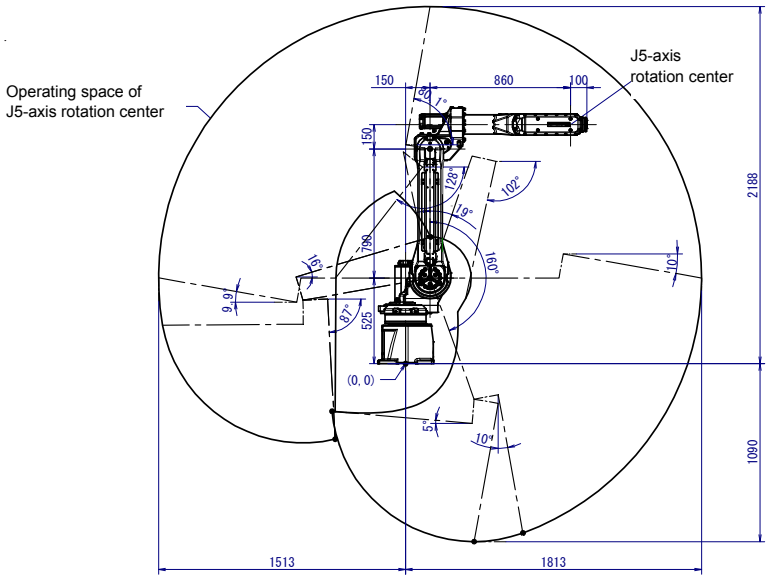


Obrázek 6.2: Rozdíl pracovních prostorů manipulátoru, pohled shora

Pro účely simulace vyjádříme manipulátor pomocí D-H úmluvy popsané v kapitole 2.2.2. V tabulce 6.1 jsou uvedeny parametry jednotlivých kloubů, získané z oficiálního webu výrobce [FAN23]. Základní souřadný systém je umístěn $450mm$ nad přírubou základny, tím je způsobena změna parametru $d_1 = 525 - 450 = 75mm$. Veškeré klouby jsou rotační a proto je parametr θ kloubová souřadnice. Pomocí dříve vypsaných instrukcí získáme šest homogenních transformačních matic, které poté sloučíme do jedné.

6.2 Simulace dalších modelů a procesních trajektorií

Vytvořený algoritmus byl testován na několika modelech obrobku s různými procesními trajektoriemi. Průběh simulace prvního modelu, včetně vyhodnocení výsledků,



Obrázek 6.3: Vizualizace geometrií robota Fanuc M20iA/20M

Obrázek 6.4: Fotografie simulovaného pracoviště robota

byl popsán v rámci kapitoly návrhu 5 pro lepší pochopení algoritmu jako takového.

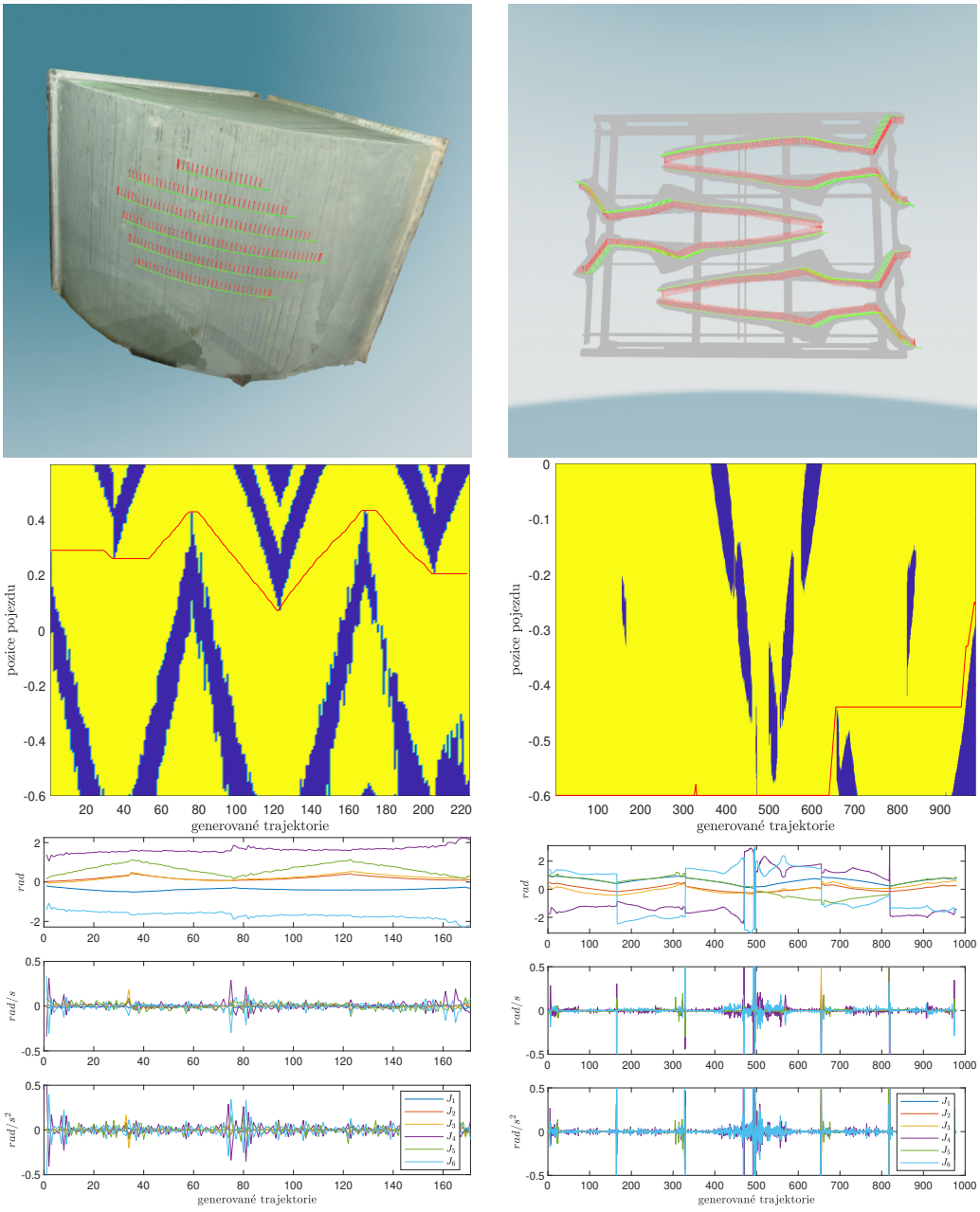
Na několika následujících obrázcích a grafech jsou zobrazeny další modely, procesní trajektorie a hlavně výsledné polohy jednotlivých kloubových souřadnic. V některých simulovaných cestách lze vidět větší skoky mezi jednotlivými body, které jsou způsobeny nespojitostmi procesní trajektorie. Tyto skoky se samozřejmě projevují na kloubových rychlostech a vznikají velká zrychlení. V reálné aplikaci by byly dotvořeny mezibody eliminující tyto skoky. Pro simulační účely byl ovšem tento problém zanedbán. Druhá anomálie, která se vyskytuje pouze na pravém obrobku

$Joint_i$	$d_i[mm]$	$\theta_i[deg]$	a_i	$\alpha_i[deg]$
1	75	θ_1	150	90
2	0	θ_2	790	0
3	0	θ_3	150	90
4	860	θ_4	0	-90
5	0	θ_5	0	90
6	100	θ_6	0	0

Tabulka 6.1: Tabulka D-H parametrů manipulátoru Fanuc M20iA/20M

v okolí pětistého bodu procesní trajektorie, je způsobena „přetečením“ úhlu šestého kloubů, tedy skokem z hodnoty $-\pi$ do $+\pi$. Reálná rychlost je ovšem mnohem menší.

I na těchto modelech, respektive procesních trajektoriích, lze vidět splnění požadavku na minimální pohyb lineárního pojezdu a zároveň požadavku na uhýbání singulárním polohám, dostatečná vzdálenost od singularit se opět projevuje minimálním zrychlením kloubových aktuátorů.



Obrázek 6.5: Ukázka dalších modelů, procesních trajektorií a výsledných kloubových souřadnic

Závěr 7

Cílem této práce bylo navrhnout řízení manipulátoru s přidaným stupněm volnosti pomocí lineárního pojezdu tak, aby se manipulátor nedostal do blízkosti singulárních poloh a zároveň byl pohyb lineárního pojezdu minimální. V práci je popsán postup popisu souřadných systémů jednotlivých ramen, jejich spojení do homogenní transformační matice a využití této matice pro vytvoření přímé i inverzní okamžité kinematické úlohy.

Pomocí simulování manipulátoru v různých polohách lineárního pojezdu byla vytvořena mapa determinantů kinematického jakobiánu, který je indikátorem singulárních poloh. Vytvořená mapa byla klíčem pro prohledávací algoritmus, který poté vygeneroval vhodné polohy lineární dráhy pro jednotlivé body procesní trajektorie. Výsledný pohyb manipulátoru byl simulován v grafickém prostředí a vyhodnocen na základě rychlosti pohybu jednotlivých kloubů, respektive jejich zrychlení. Simulace proběhly na několika modelech obrobků s různými procesními trajektoriemi a výsledný pohyb manipulátoru je vždy plynulý z hlediska rychlostí kloubových souřadnic.

Navázat na tuto práci bude možné pomocí rozšíření simulace o další dva stupně volnosti v podobě dvouosého polohovadla, které je součástí reálného pracoviště. Následně je možné rozšířit algoritmus plánování o vstupní parametr procesní rychlosti, pomocí které by bylo možné blíže simulovat reálný proces. Kompletní úlohu by poté bylo možné implementovat do reálného systému navádění robotu, ze kterého tato práce vychází.

Bibliografie

[Cor96] CORKE, P.I. A robotics toolbox for MATLAB. *IEEE Robotics & Automation Magazine*. 1996.

[CH21] CORKE, Peter; HAVILAND, Jesse. Not your grandmother's toolbox – the Robotics Toolbox reinvented for Python. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[DH55] DENAVIT, Jacques; HARTENBERG, Richard S. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*. 1955, roč. 22, č. 2, s. 215–221.

[FAN23] FANUC, Corporation. FANUC robot M20iA/20 Product Information. *Yamanashi, Japan*. 2023.

[HNR68] HART, Peter E.; NILSSON, Nils J.; RAPHAEL, Bertram. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. 1968, roč. 4, č. 2, s. 100–107.

[HB08] HUO, Liguo; BARON, Luc. The joint-limits and singularity avoidance in robotic welding. *Industrial Robot: An International Journal*. 2008.

[CZ22] CHAI, Zishu; ZHANG, Zongyuan. Mobile Robot Path Planning in 2D space: A survey. In: *2022 International Symposium on Control Engineering and Robotics (ISCER)*. IEEE, 2022.

[KK86] KHALIL, Wisama; KLEINFINGER, J. A new geometric notation for open and closed-loop robots. In: *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. IEEE, 1986, sv. 3.

[MC94] MANOCHA, Dinesh; CANNY, John F. Efficient inverse kinematics for general 6R manipulators. *IEEE transactions on robotics and automation*. 1994, roč. 10, č. 5.

[Pau81] PAUL, Richard P. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[Šve11] ŠVEJDA, Martin. *Kinematika robotických architektur*. Západočeská univerzita v Plzni, 2011.

[Šve16] ŠVEJDA, Martin. *Optimalizace robotických architektur*. Západočeská univerzita v Plzni, 2016.

[Zha+21] ZHANG, Jing; WU, Jun; SHEN, Xiao; LI, Yunsong. Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star. *International Journal of Advanced Robotic Systems*. 2021, roč. 18, č. 5.

Seznam obrázků