

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-104376-104853

PLÁNOVANIE TRAJEKTÓRIE PRE ROBOT S
KINEMATICKOU ŠTRUKTÚROU SCARA
DIPLOMOVÁ PRÁCA

2024

Bc. Michal Bíro

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-104376-104853

PLÁNOVANIE TRAJEKTÓRIE PRE ROBOT S
KINEMATICKOU ŠTRUKTÚROU SCARA
DIPLOMOVÁ PRÁCA

Študijný program:	Robotika a kybernetika
Názov študijného odboru:	kybernetika
Školiace pracovisko:	Ústav robotiky a kybernetiky
Vedúci záverečnej práce:	Ing. Michal Dobiš, PhD..

Bratislava 2024

Bc. Michal Bíro

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Robotika a kybernetika
Autor:	Bc. Michal Bíro
Diplomová práca:	Plánovanie trajektórie pre robot s kinematickou štruktúrou SCARA
Vedúci záverečnej práce:	Ing. Michal Dobiš, PhD..
Miesto a rok predloženia práce:	Bratislava 2024

Kľúčové slová: SCARA, plánovanie trajektórie, RRT

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Robotics and cybernetics
Author:	Bc. Michal Bíro
Master's thesis:	Trajectory planning for a SCARA kinematic structure robot
Supervisor:	Ing. Michal Dobiš, PhD..
Place and year of submission:	Bratislava 2024

Abstract

Keywords: SCARA, trajectory planning, RRT

Pod'akovanie

Chcel by som sa pod'akovať môjmu vedúcemu práce ...

Obsah

Úvod	1
1 Kinematika robota	2
1.1 SCARA	2
1.2 Kinematická štruktúra nášho robotického ramena	3
1.2.1 2 stupne voľnosti	5
1.2.2 3 stupne voľnosti	6
2 Algoritmy plánovania trajektórie	8
2.1 RRT	8
2.2 RRT*	9
2.3 RRT - connect (prepojený)	10
2.4 Potenciálové pole	11
2.5 STOMP algoritmus	11
3 Výber algoritmu	13
3.1 RRT	13
3.2 Riešenie RRT - python	13
3.2.1 3 stupne voľnosti	16
3.2.2 2 stupne voľnosti	17
4 Implementácia algoritmu	18
4.1 Vizualizácia	18
4.2 Objekt v kolízii	19
4.3 Overenie dosiahnuteľnosti pozície	20
4.4 Kontrola dosiahnuteľnosti cieľa	21
4.5 3 stupne voľnosti	22
4.5.1 Kartézsky súradnicový systém	22
4.5.2 Kľbový súradnicový systém	22
4.6 2 stupne voľnosti	22
5 Experiment	23
5.1 Opis experimentu	23
5.2 2 stupne voľnosti	23
5.2.1 RRT	23

5.2.2	RRT*	23
5.3	3 stupne voľnosti	23
5.3.1	Kľbový priestor	23
5.3.2	Kartézsky priestor	23
5.3.3	RRT*	23
5.4	Typy objektov	23
Záver		24

Zoznam obrázkov a tabuliek

Obrázok 1.1	SCARA robot - smery pohybov[<empty citation>]	2
Obrázok 1.2	Brightpick Autopicker	4
Obrázok 1.3	Model robota - Rviz	5
Obrázok 1.4	Pracovný priestor robotického ramena	6
Obrázok 2.1	Vizualizácia princípu RRT algoritmu	9
Obrázok 2.2	Vplyv počtu iterácií na výslednú trajektóriu [<empty citation>] . .	10
Obrázok 2.3	Prepojenie 2 stromov [<empty citation>]	10
Obrázok 2.4	Potenciálové pole [<empty citation>]	11
Obrázok 2.5	Generované trajektórie [<empty citation>]	12
Obrázok 3.1	RRT - dvojrozmerný priestor	14
Obrázok 3.2	RRT* - dvojrozmerný priestor	14
Obrázok 3.3	RRT connect - dvojrozmerný priestor	15
Obrázok 3.4	RRT - trojrozmerný priestor	15
Obrázok 4.1	Vizualizácia pracovného priestoru	18
Obrázok 4.2	Objekt v kolízii	19
Obrázok 4.3	Spojenie 2 bodov v konfiguračnom priestore	21
Obrázok 4.4	Pohyb objektu v pracovnom priestore	22
Tabuľka 1.1	Parametre ramena	5
Tabuľka 3.1	Rozsah konfiguračného priestoru - kartézsky súradnicový systém . .	16
Tabuľka 3.2	Rozsah konfiguračného priestoru - kľbový súradnicový systém . . .	16
Tabuľka 3.3	Rozsah konfiguračného priestoru - kľbový priestor	17

Zoznam algoritmov

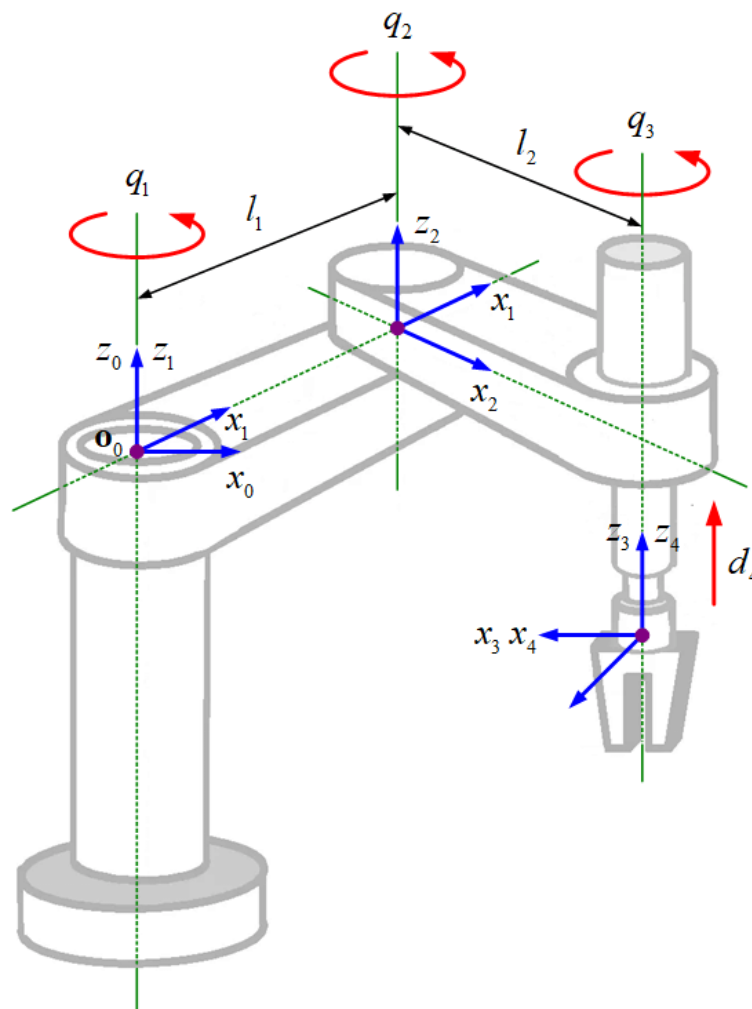
Zoznam výpisov

Úvod

1 Kinematika robota

1.1 SCARA

Scara robot je typ priemyselného robotického systému, ktorý bol vyvinutý na vykonávanie presných a opakujúcich sa úloh. Navrhol ho v roku 1979 vedec Hiroshi Makino z Yamanashi Univerzity[**<empty citation>**]. Názov SCARA je akronym (Selective Compliance Assembly Robot Arm), ktorý značí, že vertikálna poddajnosť (compliance) je väčšia ako poddajnosť v horizontálnom smere. Jeho hlavnou úlohou je najčastejšie presúvanie objektov z jedného miesta na druhé. Scara roboty majú obvykle 4 stupne voľnosti a svojou konštrukciou pripomínajú ľudskú ruku. (obr. 1.1). Ide o 3 rotačné a 1 posúvny kĺb, ktoré zabezpečujú pohyb robota vo všetkých



Obr. 1.1: SCARA robot - smery pohybov[**<empty citation>**]

osiach x , y a z . Umiestnením otáčavých kĺbov v jednej rovine je pohyb robota v porovnaní s robotickými ramenami so 6 stupňami voľnosti, značne znevýhodnený. Taktiež nosnosť týchto

robotov je značne obmedzená, z dôvodu konštrukčných parametrov ako je nosnosť ložiska. Kinematická štruktúra SCARA robotov však prináša aj výhody, pre ktoré sú tieto roboty veľmi obľúbené a často používané.

SCARA robot disponuje tromi rotačnými kĺbmi a jedným posuvným kĺbom, ktoré mu umožňujú pohyb v priestore vo všetkých troch osiach x , y a z . Tento dizajn umožňuje robotovi vykonávať presné a opakujúce sa úlohy aj keď umiestnenie rotačných kĺbov v jednej rovine obmedzuje jeho pohybovú schopnosť v porovnaní s robotmi s robotickými ramenami so 6 stupňami voľnosti. Napriek tomu, vďaka svojej kinematickej štruktúre, SCARA roboty prinášajú niekoľko výhod, ktoré ich robia obľúbenými vo viacerých odvetviach priemyslu.

Jednou z hlavných výhod je ich schopnosť dosiahnuť vysokú presnosť a rýchlosť pri vykonávaní úloh. Navyše, ich nízka hmotnosť, jednoduché ovládanie a kompaktné rozmery prispievajú k ich širokému uplatneniu. Okrem toho, SCARA roboty sú často cenovo dostupnejšie v porovnaní s inými typmi robotov, čo ich robí atraktívnou voľbou pre mnohé spoločnosti.

Vďaka týmto vlastnostiam sa SCARA roboty využívajú v rôznych odvetviach priemyslu, vrátane montáže, balenia, manipulácie s chemikáliami, potravinárskej výroby, laboratórií, farmaceutického priemyslu a automobilového priemyslu. Ich schopnosť vykonávať opakujúce sa úlohy s vysokou presnosťou a efektívnosťou ich robí dôležitým nástrojom v moderných výrobných prostrediach.

V našej práci sa nebudeme venovať len klasickej štruktúre SCARA robota so 4 stupňami voľnosti ale budeme skúmať a overovať riešiteľnosť daného problému aj pomocou iných kinematických štruktúr. Z dôvodu zníženia ceny robota budeme testovať, či je pre náš problém potrebný robot so 4 stupňami voľnosti (3 rotačné kĺby, 1 posuvný kĺb) alebo vieme štruktúru ešte viacej zjednodušiť. Naším cieľom je identifikovať najefektívnejšiu kinematickú štruktúru pre danú aplikáciu a zvážiť výhody a nevýhody rôznych prístupov.

1.2 Kinematická štruktúra nášho robotického ramena

Robotické rameno s kinematickou štruktúrou SCARA, ktorému sa v tejto práci venujeme je súčasťou komplexného robotického riešenia. Ide o robota Brightpick Autopicker (obr. 1.2) skladajúceho sa z mobilného podvozku, na ktorom je umiestnená zdvižná plošina s krabicami slúžiacimi na uskladnenie tovaru a skompletizovanie objednávky. Na stĺpe, po ktorom sa táto plošina pohybuje v smere osi z , je umiestnené naše robotické rameno. Jeho úlohou je presun tovaru z krabice s tovarom do krabice s objednávkou. Poslednou časťou tohoto robota je skener, ktorý slúži na určenie pozície prenášaného objektu v krabici.

Pre vývoj algoritmu plánovania trajektórie sa budeme zaoberať iba vybranou časťou tohto komplexného riešenia a to konkrétne robotickým ramenom. Samotné robotické rameno sa

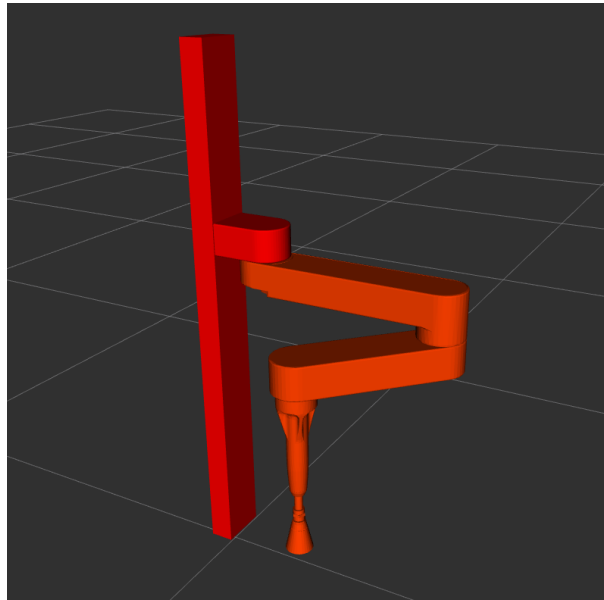


Obr. 1.2: Brightpick Autopicker

skladá z 2 ramien a 2 rotačných kĺbov (obr. 1.2). V porovnaní s klasickou kinematickou štruktúrou SCARA (obr. 2.2) nemá tretí rotačný kĺb, ktorý by zabezpečil rotáciu efektoru - teda rotáciu samotného prenášaného objektu. Taktiež neobsahuje posuvný kĺb, ktorý by zabezpečil pohyb v smere osi z . Tento pohyb totiž zabezpečuje zdvižná plošina, na ktorej sú umiestnené krabice s tovarom. Pre potreby našej úlohy si teda kinematickú štruktúru môžeme zjednodušiť na 2 stupne voľnosti - 2 rotačné kĺby. Plánovanie trajektórie budeme riešiť len od bodu uchytenia objektu efektorom robota (počiatočný bod trajektórie) po dosiahnutie bodu uloženie objektu do krabice (koncový bod trajektórie). Toto obmedzenie nám umožní lepšie optimalizovať pohyb a

rameno	dĺžka [mm]	rozsah [rad]
link1	360	$0 - \pi/2$
link2	260	$0 - 2\pi$

Tabuľka 1.1: Parametre ramena



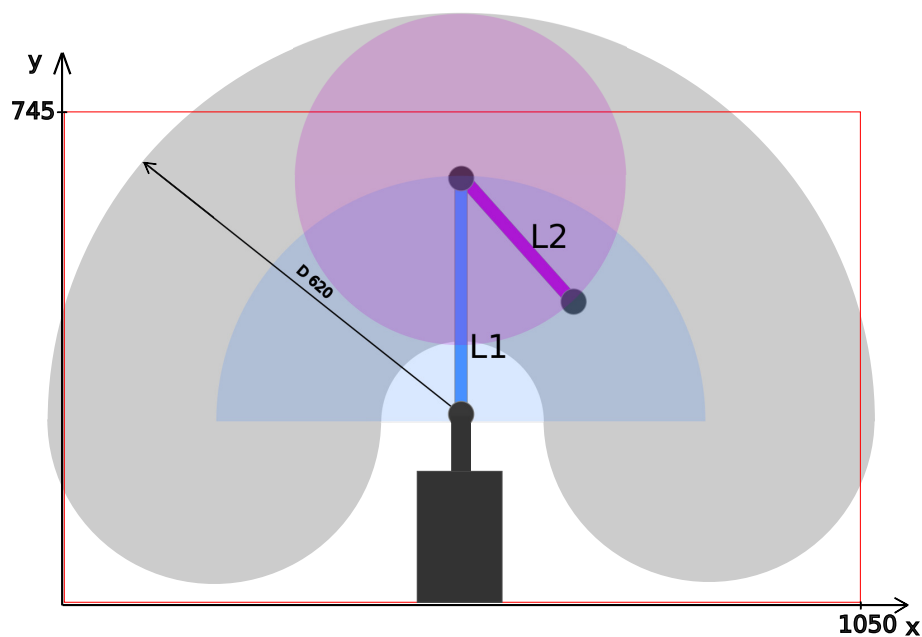
Obr. 1.3: Model robota - Rviz

zvážiť výhody a nevýhody kinematickej štruktúry.

Parametre robotického ramena sú uvedené v tabuľke 1.1. Parametre robota sa prejavujú v jeho pracovnom priestore (obr. 2.3), kde môžeme vidieť rozsah pohybu a dosiahnuteľných polôh nášho robotického ramena v 2D priestore. Modrou farbou je označené prvé rameno, rozsah jeho pohybu je vyznačený rovnakou farbou. Druhé rameno robota je označené fialovou farbou, čo platí aj pre rozsah jeho pohybu. Všetky dosiahnuteľné polohy robotickým ramenom sú označené sivou farbou. Pracovný priestor robota aj s uchopeným prenášaným objektom je na obrázku označený červenou farbou. Tento priestor je určený rozmermi celého robota, konkrétne zdvižnou plošinou, ktorej sú umiestnené krabice s tovarom a objednávkou. Pri prenášaní objektu z jednej krabice do druhej nesmie objekt ani robotické rameno vyjsť z toho priestoru aby nedošlo ku kolízii, keďže mimo tohoto rozsahu nevieme zaručiť voľný priestor.

1.2.1 2 stupne voľnosti

Prvou testovanou štruktúrou bude s 2 rotačnými kĺbmi, čo zodpovedá 2 stupňom voľnosti. V tejto kinematickej štruktúre sa budú otáčať 2 ramená robota a nástroj na uchytenie objektu, ktorý sa nachádza na konci kinematického reťazca bude fixný. Pri tejto konfigurácii je kritické



Obr. 1.4: Pracovný priestor robotického ramena

testovať schopnosť robota preniesť objekty rôznych tvarov a rozmerov zo štartovacej konfigurácie do cieľovej. Osobitnú pozornosť je potrebné venovať väčším podlhovastým objektom, kde môže nastať problém. Robot by mohol mať obmedzenú schopnosť otáčania objektov v rámci svojich dvoch rotácií ramien, čo by mohlo spôsobiť, že nedokáže dosiahnuť požadovanú konfiguráciu. Ďalším potenciálnym problémom môže byť kolízia so stĺpom, na ktorom je robotické rameno upevnené.

Testovanie na rôznych objektoch umožní identifikovať tieto obmedzenia a prípadne navrhnúť riešenia, ako ich prekonať. Zároveň bude možné získať dôležité informácie o výkonnosti a schopnostiach tohto typu kinematickej štruktúry v reálnych pracovných podmienkach.

1.2.2 3 stupne voľnosti

Druhá testovaná kinematická štruktúra sa bude líšiť od prvej pridaním ďalšieho rotačného kĺbu, ktorý umožní otáčať nástrojom na uchytienie objektu. Táto úprava by mala zlepšiť schopnosť robota nájsť vhodnú trajektóriu pre presun objektu zo štartovacej do cieľovej pozície. Počas testovania tejto konfigurácie budeme sledovať hlavne parametre, ako je čas, za ktorý robot prejde danú trajektóriu, a jej efektívnosť.

Dôležité je zaznamenať dáta, ktoré nám umožnia zhodnotiť, či pridaný rotačný kĺb prináša dostatočný prínos voči prvej konfigurácii. Tieto údaje nám pomôžu posúdiť, či investícia do efektívnejšieho robota s väčšími možnosťami je opodstatnená z hľadiska finančnej efektívnosti v porovnaní s lacnejším robotom, ktorý má obmedzené možnosti.

Kombinácia dosiahnuteľnosti cieľovej polohy pri rozličných rozmeroch objektov, časo-

vých dát a analýza efektivity bude kľúčová pri rozhodovaní, ktorá kinematická štruktúra je najvhodnejšia pre danú aplikáciu z hľadiska nákladov a výkonnosti.

2 Algoritmy plánovania trajektórie

Algoritmus plánovania trajektórie je matematický postup alebo procedúra, ktorá určuje optimálnu cestu alebo pohyb medzi počiatočným a koncovým bodom v priestore. Cieľom algoritmu plánovania trajektórie je nájsť optimálnu trajektóriu, ktorá spĺňa určité kritériá, ako sú bezpečnosť, efektivita, minimálny čas, minimalizácia spotrebovanej energie, alebo minimalizácia vplyvu na okolie. Tieto kritériá môžu byť definované podľa konkrétnej aplikácie a požiadaviek.

V tejto kapitole sa budeme venovať analýze rôznych algoritmov využívaných v oblasti robotiky. Na základe tejto analýzy budeme schopní určiť, ktorý z týchto algoritmov najlepšie vyhovuje našim potrebám a zabezpečí optimálnu kombináciu medzi presnosťou a rýchlosťou výpočtu. Vybraný algoritmus bude slúžiť ako základ pre úspešnú implementáciu a funkčnosť nášho systému pre plánovanie trajektórií.

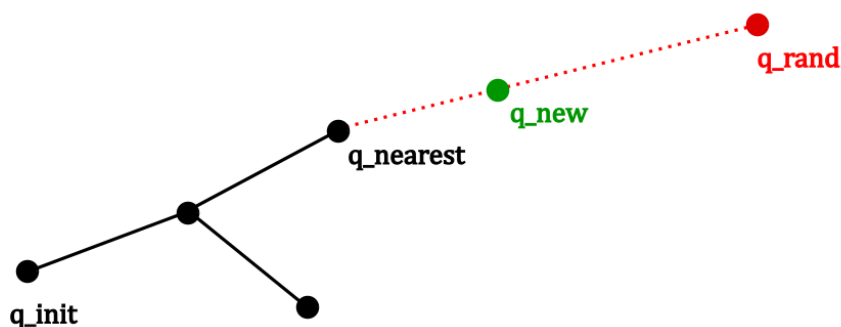
2.1 RRT

Algoritmus RRT – z anglického Rapidly-exploring Random Tree, by sa dal do slovenčiny preložiť ako rýchlo rastúci náhodný strom. Algoritmus je v oblasti robotiky veľmi rozšírený a obľúbený vďaka svojej schopnosti rýchlo prehľadávať vysoko dimenzionálny konfiguračný priestor, v ktorom zohľadňuje prekážky v priestore ako aj dynamiku telesa.

Ide o algoritmus založený na prehľadávaní konfiguračného priestoru C , kde sa v iteráciách vytvárajú náhodné uzly - q , ktorých spájaním vznikajú nové potenciálne cesty. Každý uzol q reprezentuje pozíciu a orientáciu telesa v 2D alebo 3D priestore []. Pri plánovaní trajektórie je generovaný kontinuálny rad uzlov (obr. 1.2), ktorý začína v počiatočnom stave q_{init} , a postupne sa rozrastá až dosiahne koncový stav q_{goal} . Generovanie stromu prebieha v iteráciách kedy sa vždy vygeneruje náhodná konfigurácia q_{rand} a nájde sa k nej najbližší uzol patriaci stromu. Od daného uzla je následne vo vzdialenosti v od uzla $q_{nearest}$ vytvorený nový uzol q_{new} . Proces rozrastania stromu sa končí v momente ak sa q_{new} nachádza v okolí cieľovej konfigurácie q_{goal} , ktoré je definované vzdialenosťou d .

Ak neuvažujeme voľný konfiguračný priestor C_{free} treba v procese generovania taktiež overovať kolíziu s danými prekážkami v priestore - C_{obs} . Novo generovaný uzol stromu nesmie nachádzať v priestore prekážky - C_{obs} , a ani cesta medzi dvoma susednými uzlami nesmie kolidovať s prekážkou. Pokiaľ uzol spĺňa tieto podmienky, je zaradený do štruktúry stromu, v opačnom prípade je vyradený a proces pokračuje ďalej.

Ide o neoptimálne riešenie, keďže body v konfiguračnom priestore sú generované náhodne



Obr. 2.1: Vizualizácia princípu RRT algoritmu

a výsledný tvar trajektórie vzniká ako najkratšia cesta medzi nami vygenerovanými bodmi. Na zlepšenie trajektórie existuje viacero možností, ktoré ponúkajú suboptimálne riešenia problému, jedným z nich je napríklad RRT*.

Algoritmus RRT môže mať nevýhodu v generovaní neoptimálnych výsledkov, pretože generuje body v konfiguračnom priestore náhodne a tvorí trajektóriu ako najkratšiu cestu medzi týmito bodmi. Na zlepšenie výslednej trajektórie existuje niekoľko možností, medzi ktorými je aj RRT*.

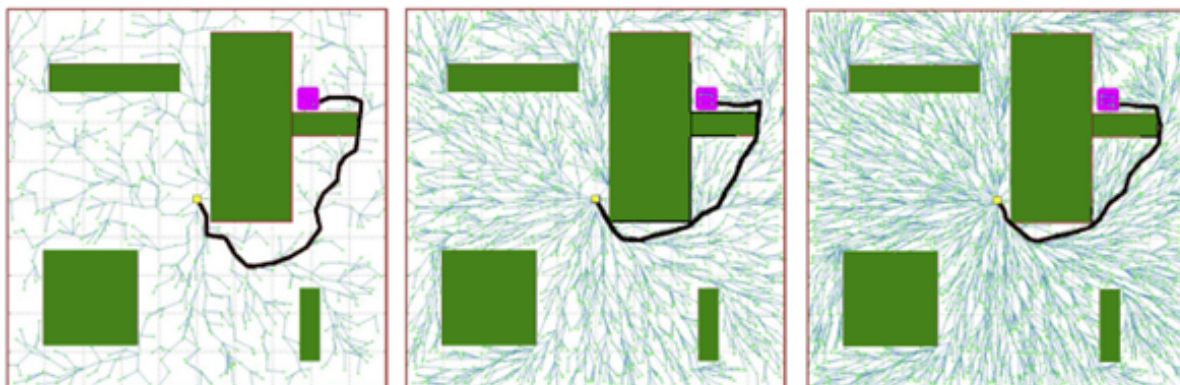
2.2 RRT*

Jedná sa o modifikáciu RRT algoritmu, ktorej cieľom je v porovnaní s pôvodným RRT nájsť kratšiu trajektóriu. Hlavnou úpravou oproti jednoduchému RRT algoritmu je výpočet vzdialenosti z počiatočného do aktuálneho uzlu a následné overenie, či neexistuje v okolí uzol, ktorého vzdialenosť by bola menšia ako vzdialenosť momentálneho prepojenia. Ak táto situácia nastane, algoritmus vyberie prepojenie uzlov, ktoré vytvoria kratšiu trasu. Algoritmy sa tiež líšia ukončovacou podmienkou, kde pre RRT* je určený jasný počet iterácií. Od počtu iterácií závisí výsledný tvar trajektórie, kde so zvyšujúcim sa počtom je algoritmus schopný nájsť kratšie a plynulejšie trasy (obr. 2.2).

Tieto úpravy a rozdiely umožňujú algoritmu RRT* dosiahnuť lepšie výsledky v porovnaní s pôvodným RRT, čo ho robí atraktívnou voľbou pre úlohy, kde je kritické dosiahnutie čo najlepšej trajektórie v danom priestore. Je dôležité zohľadniť aj možnosť zvýšenia času výpočtu pri použití algoritmu RRT*. Aj keď má RRT* potenciál nájsť kratšie a plynulejšie trasy ako klasický RRT, zvyšuje sa časová náročnosť algoritmu vzhľadom na výpočet vzdialeností a overovanie optimality spojení medzi uzlami.

Pri navrhovaní algoritmu pre plánovanie trajektórií je preto dôležité hľadať rovnováhu medzi kvalitou výsledných trajektórií a časovou náročnosťou výpočtu. Niekedy môže byť ak-

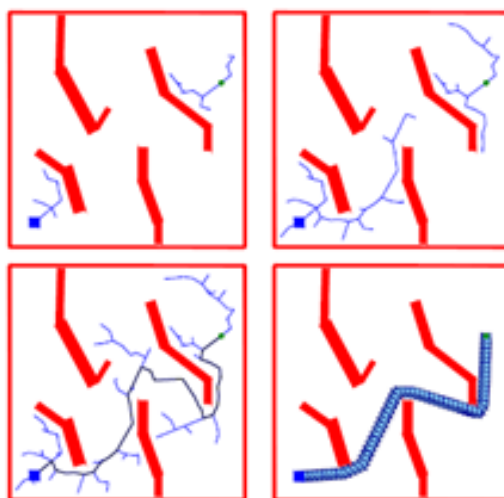
ceptovateľné mať mierne suboptimálnu trajektóriu, ak to znamená výrazné zrýchlenie výpočtu.



Obr. 2.2: Vplyv počtu iterácií na výslednú trajektóriu [[empty citation](#)]

2.3 RRT - connect (prepojený)

Algoritmus RRT Connect (spojenie) je ďalšou modifikáciou algoritmu RRT. V procese rozširovania stromovej štruktúry vytvára dva stromy z počiatočnej a koncovj konfigurácie, ktoré sa šíria v konfiguračnom priestore, až kým nedôjde k ich spojeniu a vytvoreniu cesty. (obr. 2.3).



Obr. 2.3: Prepojenie 2 stromov [[empty citation](#)]

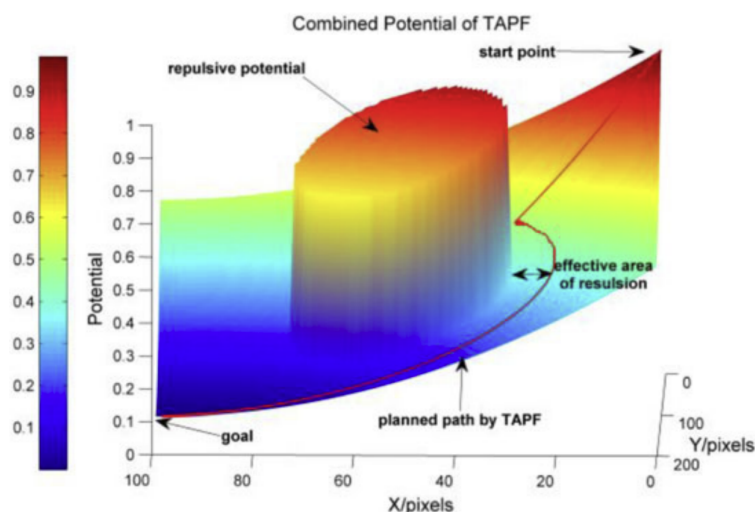
Tento algoritmus má výhodu v tom, že nájdenie trajektórie je často ľahšie dosiahnuteľné jedným z dvoch možných smerov pohybu medzi dvoma bodmi, čo závisí od prekážok v priestore C_{obs} . Použitím algoritmu RRT Connect sa zvyšuje pravdepodobnosť úspešného nájdenia trajektórie od počiatočného bodu ku koncovému.

Tento prístup umožňuje efektívnejšie prehľadávanie konfiguračného priestoru, pretože vytvára dva stromy, ktoré sa šíria smerom k cieľu. Tým sa zvyšuje šanca na rýchlejšie nájdenie vhodnej trajektórie, najmä v prípade, že existuje jasný smer pohybu medzi počiatočným a koncovým bodom.

2.4 Potenciálové pole

Plánovanie trajektórie na základe potenciálového poľa využíva koncept odpudivých a príťažlivých polí. Príťažlivé pole generuje veľmi nízke hodnoty so stredom v cieľovom bode, ktoré sa so zväčšujúcou sa vzdialenosťou od cieľa zvyšujú. Odpudivé pole naopak generuje veľmi vysoké hodnoty v okolí prekážok v priestore. Kombináciou týchto dvoch polí dostávame tzv. potenciálové pole (obr. 2.4) so silným sklonom ku cieľu, čo umožňuje generovanie trajektórie s tendenciou vyhýbať sa prekážkam.

Tento prístup k plánovaniu trajektórie je intuitívny a efektívny v situáciách, kde je cieľ jasne definovaný a potrebné je vyhnúť sa prekážkam v priestore. Použitie potenciálového poľa môže viesť k vytvoreniu plynulej a bezpečnej trajektórie, ktorá dosahuje cieľový bod s minimálnym rizikom kolízií.



Obr. 2.4: Potenciálové pole [empty citation]

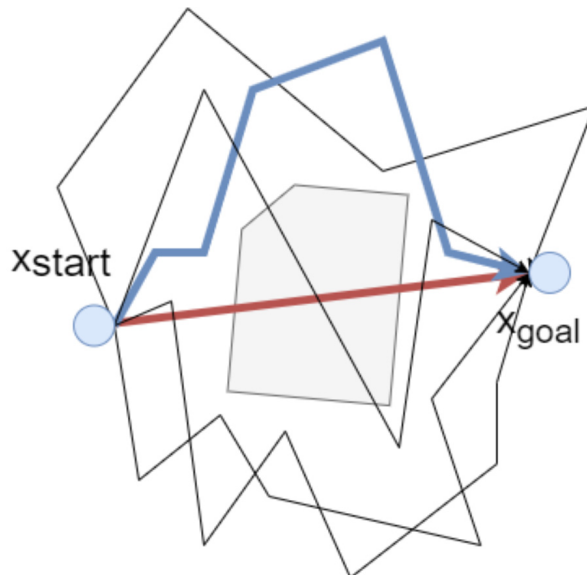
2.5 STOMP algoritmus

Algoritmus STOMP (Stochastic Trajectory Optimized Motion Planner) funguje na základe iteratívneho procesu, ktorý sa skladá z niekoľkých krokov. Počiatočná trajektória je generovaná ako lineárna interpolácia medzi počiatočným a koncovým bodom. STOMP pracuje s fixným počtom krokov na trajektóriu, nazývaných "timesteps", a tiež s definovaným počtom iterácií,

v ktorých algoritmus hľadá trajektóriu. Následne je generovaný šum, ktorý je aplikovaný na každý bod trajektórie, čo umožňuje prehľadávanie konfiguračného priestoru v okolí počiatočnej trajektórie (obr. 2.5).

Každá vygenerovaná trajektória je potom ohodnotená pomocou účelovej funkcie. V prípade, že priestor obsahuje veľké množstvo prekážok, môže nastať situácia, kedy algoritmus nenájde riešenie. Zvýšením hodnoty šumu sa zväčší priestor prehľadávania a zvýši sa pravdepodobnosť nájdenia bezkolíznej trasy, avšak za cenu zníženej optimalizácie trajektórie. Toto môže byť riešené zvýšením počtu iterácií, avšak to môže mať za následok zvýšenie výpočtového času algoritmu.

Zvolenie správnych parametrov a vyváženie medzi priestorom prehľadávania, optimalizáciou trajektórie a výpočtovým časom je kľúčové pre úspešné použitie algoritmu STOMP v plánovaní trajektórie v prostredí s prekážkami.



Obr. 2.5: Generované trajektórie [**<empty citation>**]

3 Výber algoritmu

Hlavnou časťou našej práce je implementácia algoritmu na hľadanie trajektórie pre pohyb objektu v priestore. Výber správneho algoritmu je preto kľúčový. Je nevyhnutné vybrať taký algoritmus, ktorý najlepšie zodpovedá našej úlohe.

Pri rozhodovaní o vhodnom algoritme je dôležité zohľadniť nielen schopnosť algoritmu nájsť optimálnu trajektóriu, ale aj jeho rýchlosť výpočtu. Vzhľadom na to, že náš robotický systém pracuje v reálnom čase, je kľúčové zvoliť algoritmus, ktorý dokáže rýchlo generovať trajektórie bez zbytočného oneskorenia.

V tejto kapitole sa budeme bližšie venovať algoritmu plánovania trajektórie, ktorý sme sa rozhodli zvoliť na riešenie nášho problému a vysvetlíme si dôvody jeho výberu. Taktiež si predstavíme existujúce riešenie, na ktorom budeme v našej práci stavať, rozširovať a upravovať ho podľa potrieb našej úlohy. Na záver si vysvetlíme, ako budeme pristupovať k plánovaniu trajektórie pri jednotlivých kinematických štruktúrach, ktoré budeme testovať.

3.1 RRT

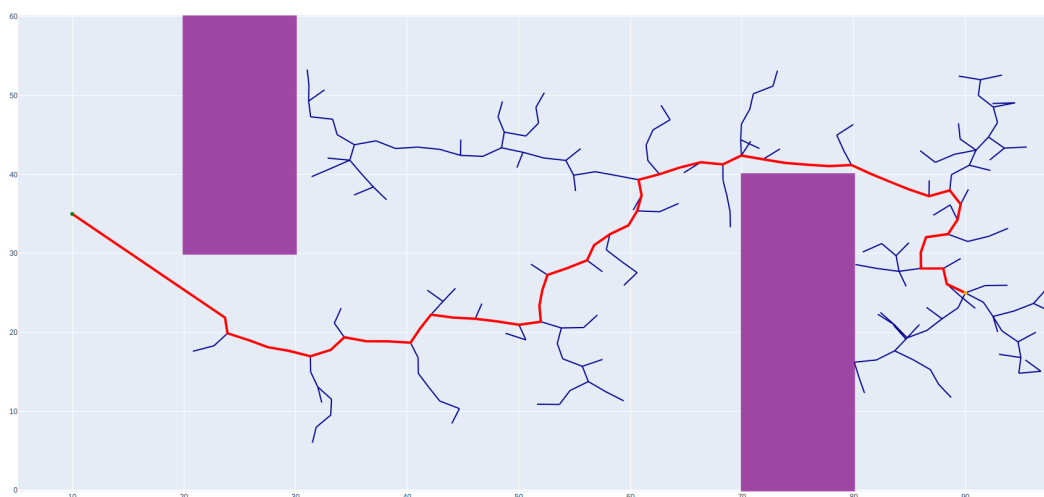
Analýzou jednotlivých typov algoritmov pre plánovanie trajektórie sme dospeli k záveru, že najvhodnejšou voľbou pre našu úlohu bude RRT algoritmus. Tento výber sme urobili najmä pre jeho schopnosť rýchlo prehľadávať konfiguračný priestor. RRT algoritmus je vhodný nielen pre svoju rýchlosť, ale aj pre jeho univerzálnosť, pretože ho vieme využiť ako v kartézskom súradnicovom systéme, tak aj v kľbovej súradnicovej sústave. Táto flexibilita bude dôležitá pri implementácii plánovania trajektórie pre rôzne kinematické štruktúry.

Ďalšou výhodou RRT algoritmu je jeho jednoduchá a rýchla modifikovateľnosť podľa našich potrieb. V našej práci budeme tiež porovnávať RRT algoritmus s jeho modifikáciou, RRT*. Budeme sa snažiť nájsť optimálne riešenie z hľadiska času výpočtu a kvality výslednej trajektórie.

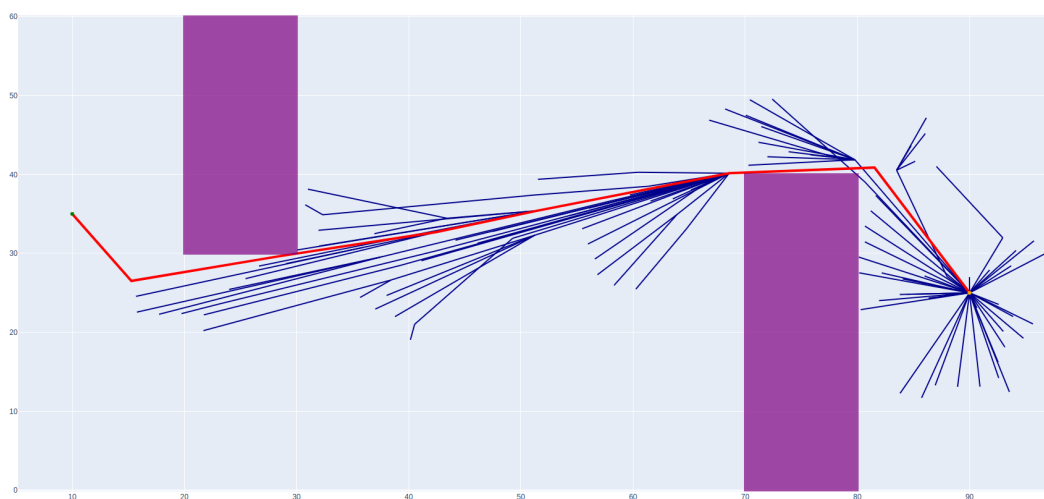
3.2 Riešenie RRT - python

RRT algoritmus je široko využívaný a populárny, a preto existuje mnoho dostupných implementácií v rôznych programovacích jazykoch. V našej práci budeme pracovať v jazyku Python, preto sme sa rozhodli analyzovať rôzne voľne dostupné implementácie tohto algoritmu v tomto jazyku.

Z dostupných riešení a implementácií RRT algoritmu v jazyku Python sme sa rozhodli použiť implementáciu od autorov knižnice rrt-algorithms []. Obsahuje samotný RRT algoritmus (obr. 3.1) ako aj jeho modifikácie - RRT* (obr. 3.2) a RRT connect (obr. 3.3).



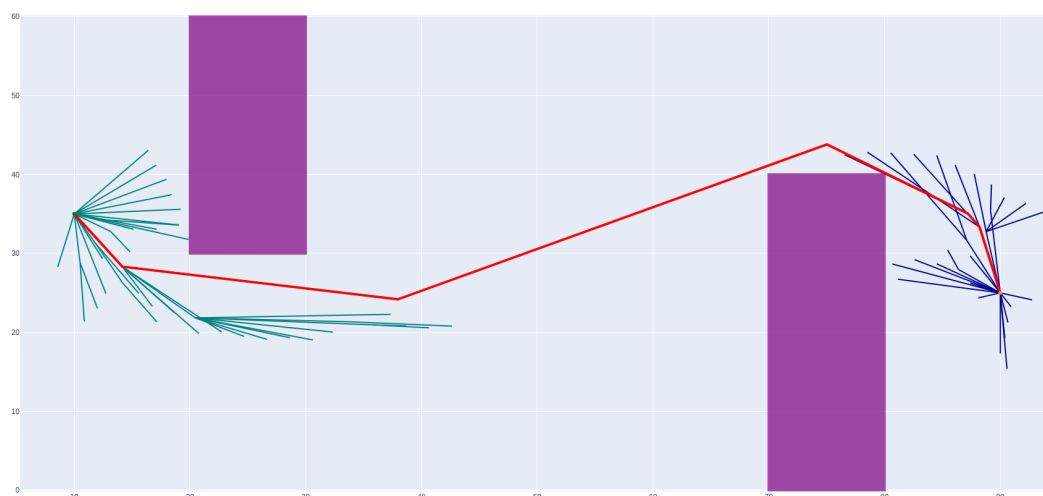
Obr. 3.1: RRT - dvojrozmerný priestor



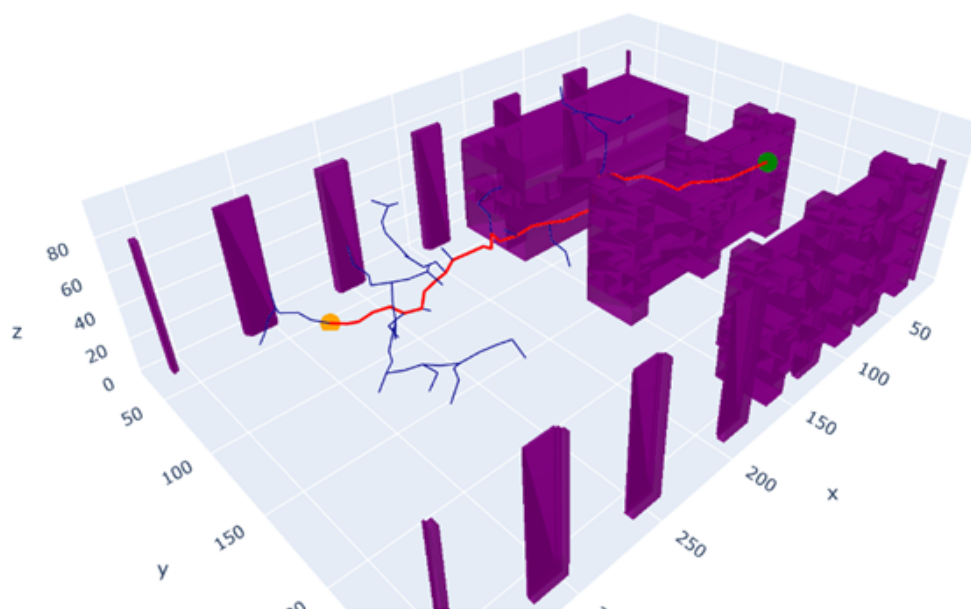
Obr. 3.2: RRT* - dvojrozmerný priestor

Riešenie, na ktorom budeme stavať, je vytvorené pre plánovanie v kartézskom súradnicovom systéme a to v dvoj aj v trojrozmernom priestore (obr. 3.4). Algoritmu je potrebné zadať rozsah prehl'adávaného priestoru, počiatkový a koncový bod a ak sa v priestore nachádzajú prekážky, ich pozície v prehl'adávacom priestore. Ako môžeme vidieť na obrázkoch 3.1, 3.2, 3.3, táto implementácia pracuje iba s hmotným bodom, neuvažuje teda rozmery prenášaného objektu.

Pre riešenie našej úlohy bude potrebné rozšíriť tento algoritmus tak, aby neuvažoval o prenášanom objekte ako o hmotnom bode, ale aby bral do úvahy aj jeho rozmery. Prístup, ktorý je možné zvoliť je kontrola kolízií v generovaných bodoch na základe ktorej bude tento bod uznaný za nevyhovujúci a bude generovaná nová bezkolízna konfigurácia. Objekt bude v pries-



Obr. 3.3: RRT connect - dvojrozmerný priestor



Obr. 3.4: RRT - trojrozmerný priestor

tore definovaný bodom, v ktorom je uchopený robotickým ramenom - v našom prípade to bude geometrický stred objektu, a jeho orientáciou v priestore. Stred bude definovaný súradnicami x, y pracovného priestoru v kartézskom súradnicovom systéme. Orientácia objektu bude reprezentovaná ako uhol medzi osou objektu a osou x v stupňoch.

Plánovanie bude prebiehať na rozdielnych kinematických štruktúrach, preto je dôležité zvoliť správny prístup pre každú z nich a prispôbiť algoritmus ich individuálnym potrebám.

3.2.1 3 stupne voľnosti

Plánovanie trajektórie pri kinematickej štruktúre s 3 stupňami voľnosti budeme riešiť primárne v kartézskom súradnicovom systéme. Pre implementáciu nášho algoritmu použijeme RRT v trojrozmernom konfiguračnom priestore, pričom osi x a y budú reprezentovať osi x a y v pracovnom priestore robota. Os z konfiguračného priestoru bude zodpovedať natočeniu objektu okolo jeho stredu, ktorý nám bude reprezentovať bod uchytenia objektu. Rozsah konfiguračného priestoru (tab. 3.1) bude zodpovedať rozsahu pracovnému priestoru robota v osiach x, y a rozsahu maximálnej rotácie objektu v priestore.

x [mm]	y [mm]	z [°]
0 - 1050	0 - 745	0 - 360

Tabuľka 3.1: Rozsah konfiguračného priestoru - kartézsky súradnicový systém

Pri kinematickej štruktúre s 3 stupňami voľnosti budeme tiež testovať plánovanie trajektórie v kĺbovom priestore. Tieto dáta budeme neskôr používať na porovnanie plánovania trajektórie v kĺbovom a kartézskom priestore. Budeme sledovať čas výpočtu algoritmu, úspešnosť nájdenia cesty a kvalitu trajektórie.

Plánovanie trajektórie v kĺbovom priestore nám umožní preskúmať, ako dobre sa algoritmus dokáže prispôbiť špecifikám kinematickej štruktúry robota a aký vplyv má voľba priestoru na výslednú trajektóriu. Porovnanie výsledkov s plánovaním trajektórie v kartézskom priestore nám poskytne ucelený obraz o výkonnosti a efektívnosti oboch prístupov.

Osi konfiguračného priestoru x, y a z budú odpovedať otočeniam jednotlivých kĺbov robotického ramena (tab. 3.2).

x [°]	y [°]	z [°]
0 - 180	0 - 360	0 - 360

Tabuľka 3.2: Rozsah konfiguračného priestoru - kĺbový súradnicový systém

3.2.2 2 stupne voľnosti

Plánovanie trajektórie pri štruktúre s 2 stupňami voľnosti, ktoré predstavujú 2 rotačné kĺby robotického ramena, nám neumožňuje plánovať v kartézskom priestore, pretože nie každá konfigurácia by bola dosiahnuteľná. Z tohto dôvodu je nevyhnutné použiť plánovanie v kĺbovom konfiguračnom priestore. Pôjde o dvojrozmerný konfiguračný priestor, kde osi x a y budú predstavovať rotácie jednotlivých kĺbov. Rozsah konfiguračného priestoru, ktorý odpovedá rozsahu pohybu jednotlivých kĺbov je uvedený v tabuľke 3.2.

$x [^\circ]$	$y [^\circ]$
0 - 180	0 - 360

Tabuľka 3.3: Rozsah konfiguračného priestoru - kĺbový priestor

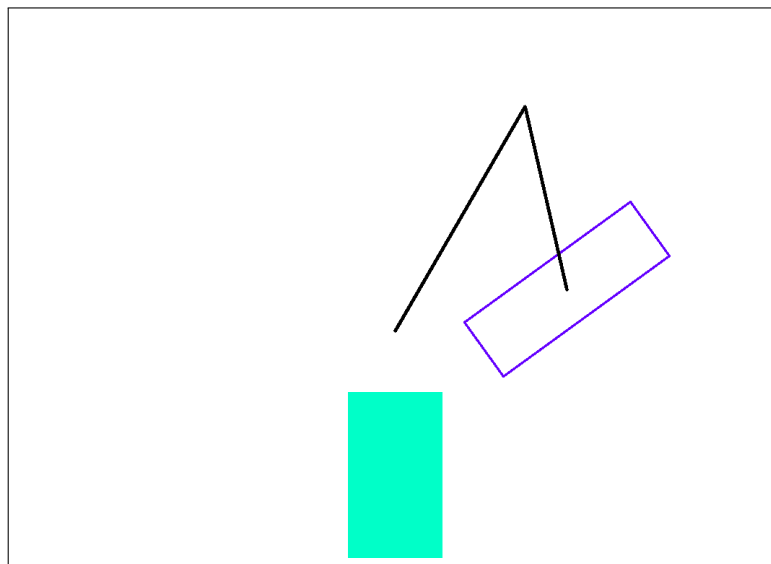
4 Implementácia algoritmu

Kým v predošlej kapitole (kap. 3) sme si v jednoduchosti vysvetlili ako budeme pristupovať ku plánovaniu trajektórie pri jednotlivých kinematických štruktúrach a opísali sme si konfiguračný priestor pre jednotlivé štruktúry, v tejto kapitole si bližšie vysvetlíme, čo bolo potrebné implementovať do pôvodného kódu aby vyššie spomenuté prístupy fungovali.

Ukážeme si fundamentálne funkcie, na ktorých stojí náš algoritmus a vysvetlíme si ich úlohy a prístup, ktorý sme pri ich implementácii zvolili.

4.1 Vizualizácia

Pre potreby vývoju nášho algoritmu bolo v prvom rade potrebné vytvoriť vizualizáciu prenášaného objektu v priestore. Pôvodné riešenie uvažovalo len hmotný bod, ktorého pohyb v priestore sa dá jednoducho zobrazit' v grafe. Pri našom probléme má však prenášaný objekt svoj, tvar, rozmery a orientáciu v priestore. Z toho dôvodu bola vytvorená vizualizácia, ktorá nám zobrazuje priemet pracovného priestoru robota v osiach x , y (obr. 4.1). Obsahuje prenášaný predmet, prekážku, ktorá predstavuje stĺp, na ktorom je robotické rameno uchytené v prípade plánovania trajektórie v kĺbovom priestore aj samotné robotické rameno. Pracovný priestor spolu s prekážkami a objektom sme vizualizovali v mierke $1\text{ mm} = 1\text{ pixel}$.



Obr. 4.1: Vizualizácia pracovného priestoru

Táto vizualizácia spĺňala v prvom rade kontrolnú funkciu, či vygenerovaná trajektória spĺňa dané požiadavky - sa objekt pri pohybe vyhýba prekážkam a nevychádza z pracovného priestoru.

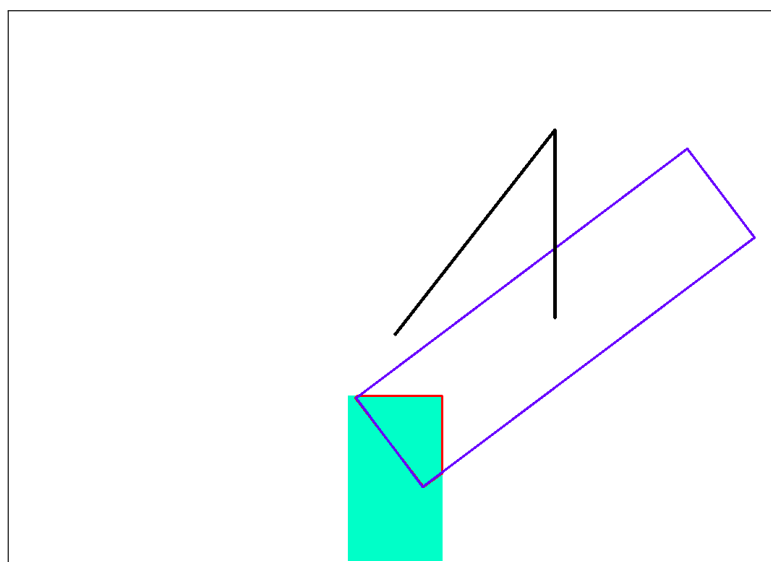
Prenášaný objekt sme si vizualizovali ako obdĺžnik. Ide o univerzálny geometrický tvar najlepšie odpovedajúci väčšine tovaru, ktorý by sa mohol v sklade nachádzať. Takmer každá krabica alebo iný objekt sa nám do roviny xy premietne práve ako obdĺžnik. Objekt sme teda definovali pozíciou jeho stredu, výškou, šírkou a uhlom otočenia.

Pre vypracovanie tejto časti sme využili knižnicu openCV.

4.2 Objekt v kolízii

Hmotný bod prenášaný priestorom má v porovnaní s objektom veľkú výhodu práve v tom, že vieme presne povedať kde v priestore sa nachádza. Tým pádom vieme jednoducho určiť, či je v kolízii s prekážkami alebo opustil pracovný priestor. Pri objekte vieme takýmto spôsobom kontrolovať len jeden jeho bod, ktorým je definovaný a to je v našom prípade jeho stred. Úlohu kontroly kolízie objektu s prekážkami to teda výrazne komplikuje aj vzhľadom na to, že náš objekt mení v priestore aj svoju orientáciu.

Preto bolo potrebné vytvoriť funkciu, ktorej úlohou bude zistiť či sa náš objekt v danej konfigurácii nenachádza v kolízii alebo neopustil pracovný priestor robota. Touto funkciou je *collision_check()*. Ide o základnú funkciu na ktorej stojí celý náš algoritmus a je identická pre všetky kinematické štruktúry. Pracuje v kartézskom súradnicovom systéme. Na základe vstupných údajov, ktorými sú pozícia stredu objektu - súradnice x a y , rozmery objektu - šírka a výška, a natočenie objektu nám vráti informáciu o tom či sa objekt nachádza v bezkolíznej pozícii.



Obr. 4.2: Objekt v kolízii

Pre kontrolu kolízie objektu s prekážkami v priestore sme použili knižnicu shapely. Pomo-

cou jej funkcií sme boli schopní na základe parametrov objektu získať pozíciu jeho rohov v priestore a tiež overiť prienik 2 objektov - prenášaného a prekážky, čo nám slúžilo na zistenie kolízie (obr. 4.2).

Do tejto funkcie sme zahrnuli kontrolu situácie, že prenášaný objekt opustí pracovný priestor robota. Keďže nevieme zaručiť, že priestor mimo pracovného priestoru robota bude voľný, treba ho považovať za obsadený. V prípade, že objekt opusti pracovný priestor je táto konfigurácia tiež považovaná za kolíziu.

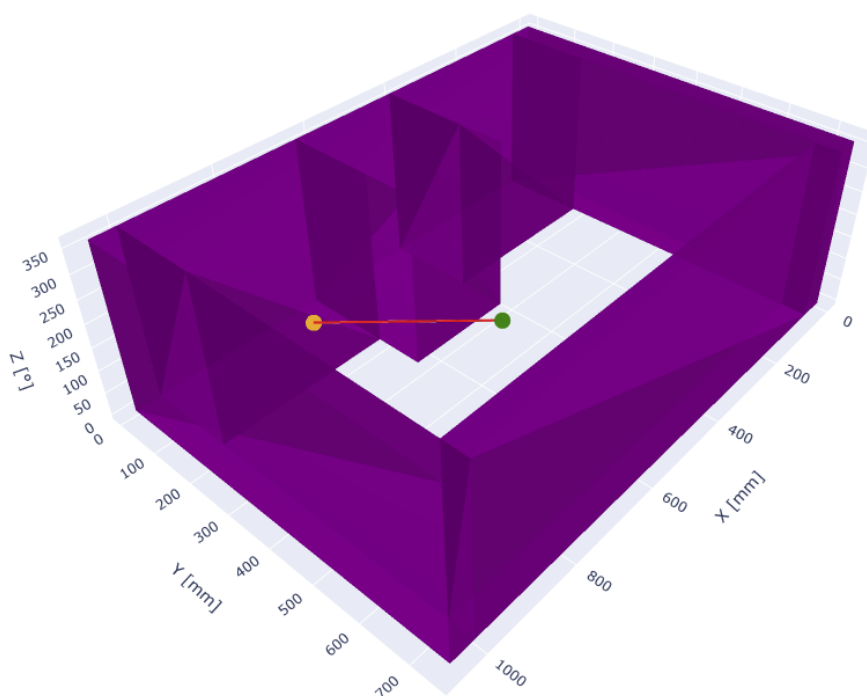
4.3 Overenie dosiahnuteľnosti pozície

Generovanie trajektórie prebieha v krokoch, kedy sú postupne generované nové konfigurácie alebo inak povedané pozície, ktorými objekt prechádza od počiatku k cieľu (kapitola 2.1). Prvým krokom je určiť, či sa vygenerovaný nachádza v bezkolíznej konfigurácii, k čomu slúži funkcia *collision_check()*. Druhým krokom je zistiť, či sa objekt z bodu, v ktorom sa práve nachádza dokáže dostať do nového bodu, ktorý bol vygenerovaný. Ak by ho objekt z danej pozície nemohol dosiahnuť napr. kvôli prekážke v ceste, tento bod nemôže byť pridaný do generovaného stromu. V prípade hmotného bodu ide o priame spojenie 2 bodov priamkou. Ak sa priamka pretína s prekážkou tak táto pozícia nie je dosiahnuteľná.

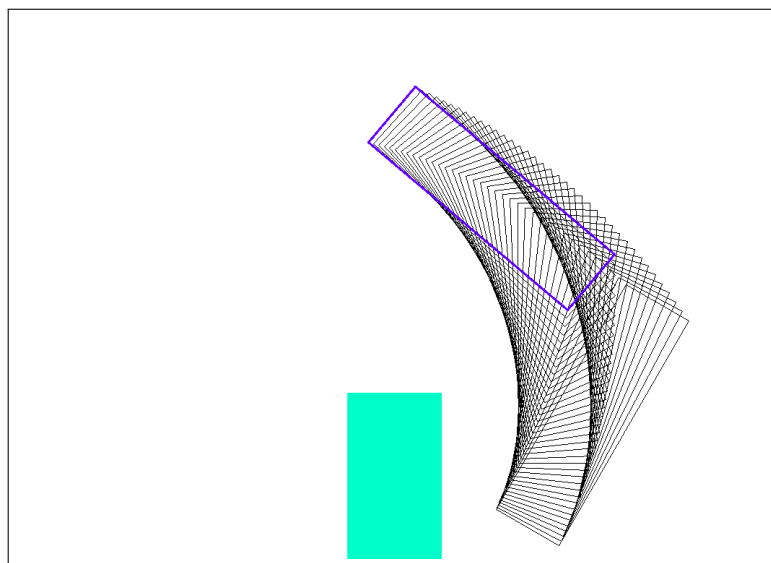
Pri objekte treba uvažovať jeho pohyb v priestore. Každá pozícia, ktorou objekt pri danom pohybe prejde musí byť bezkolízna. Na obrázkoch 4.3 a 4.4 môžeme vidieť porovnanie medzi pracovným a konfiguračným priestorom. Kým v konfiguračnom priestore je pohyb opísaný priamkou v trojrozmernom priestore, v pracovnom priestore ide o pohyb obdĺžnika, kedy sa menia jeho pozície v osiach x , y aj uhol natočenia.

Pre simuláciu pohybu v priestore sme použili lineárne vzorkovania. Poznáme počiatočnú a koncovú pozíciu, ktoré nám definujú pohyb - prejdenú vzdialenosť a smer. Vypočítame si veľkosť úsečky medzi danými 2 pozíciami v konfiguračnom priestore. Na základe nami zvolenej veľkosti vzorky si rozdelíme danú uhlopriečku na daný počet vzoriek. Postupne inkrementujeme jednotlivé prvky počiatočnej pozície $(x, y, uhol)$ o hodnotu, ktorá odpovedá veľkosti vzorky v danej osi. Tým dosiahneme prechodové pozície, pričom v každej z nich kontrolujeme, či sa objekt nenachádza v kolízii. Ak sú všetky prechodové pozície vyhodnotené ako bezkolízne vieme, že daný cieľový bod je dosiahnuteľný. Túto úlohu zabezpečuje funkcia *linear_sampling_collision_check()*.

OBRAZOK - lineárne vzorkovanie



Obr. 4.3: Spojenie 2 bodov v konfiguračnom priestore



Obr. 4.4: Pohyb objektu v pracovnom priestore

4.4 Kontrola dosiahnuteľnosti cieľa

Ukončovacia podmienka algoritmu býva vo väčšine prípadov určená vzdialenosťou generovaného bodu v strome od cieľovej pozície, keď sa generovaný bod dostane do blízkosti cieľa je vytvorené prepojenie a algoritmus je ukončený. Ďalšou možnosťou je použiť ako ukončovacia podmienku maximálny počet iterácií. Algoritmus prehl'adáva priestor až dokým nie je

vygenerovaný daný počet bodov a až následne hľadá prepojenie s cieľom. Táto ukončovacia podmienka sa dá použiť napríklad pri RRT*, kedy je počtu iterácií úmerná kvalita výslednej trajektórie. Pri algoritme RRT-connect sú generované 2 stromy a ukončenie algoritmu nastane pri ich spojení.

V našom algoritme využíva určitú kombináciu jednotlivých prístupov. Jedným so vstupných parametrov, ktorý si definujeme pri spustení algoritmu je pravdepodobnosť, s ktorou si bude algoritmus v priebehu generovania stromu kontrolovať dosiahnuteľnosť cieľa. V prípade, že je cieľ z daného bodu dosiahnuteľný je vygenerovaná trajektória a algoritmus je ukončený. Na kontrolu dosiahnuteľnosti cieľa z posledného generovaného bodu nám slúži funkcia *linear_sampling_collision_check()*.

Z dôvodu jednoduchosti pracovného priestoru (obsahuje len jednu prekážku) vo väčšine prípadov prenášaných objektov bude možné nájsť priame spojenie hneď z počiatočnej do cieľovej pozície. Za účelom zvýšenia rýchlosti výpočtu algoritmu v daných prípadoch si algoritmus hneď na začiatku skontroluje dosiahnuteľnosť cieľa, čím sa preskočí celý proces generovania stromu.

Algoritmu je tiež stanovený maximálny počet iterácií. V prípade dosiahnutia maximálneho počtu iterácií je overená dosiahnuteľnosť cieľa z daného bodu. Ak spojenie neexistuje algoritmus je ukončený a trajektória nebola nájdená.

4.5 3 stupne voľnosti

4.5.1 Kartézsky súradnicový systém

co sme pridali - preco vizualizacia, collisioncheck, path sampling, ... vyvojove diagramy orezanie konfiguračného priestoru

4.5.2 Kľbový súradnicový systém

co sme pridali - preco - FK, IK , vizualizacia, ... vyvojove diagramy orezanie konfiguračného priestoru

4.6 2 stupne voľnosti

opisat rozdiel medzi 2DOF a 3DOF princíp predrotácie vyvojove diagramy

5 Experiment

5.1 Opis experimentu

5.2 2 stupne voľnosti

5.2.1 RRT

5.2.2 RRT*

5.3 3 stupne voľnosti

5.3.1 Kľbový priestor

5.3.2 Kartézsky priestor

5.3.3 RRT*

5.4 Typy objektov

- experiment s 5 typmi objektov v krabici

Záver

Conclusion is going to be where?

Here.