UNIVERSIDADE D
COIMBRA

Rita Salomé Carvalho Gama

SISTEMA ROBÓTICO PARA A IMPRESSÃO DIRETA *IN SITU* EM ÓRGÃOS

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, na especialização de Automação, orientada pelo Professor Doutor Rui Pedro Duarte Cortesão e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Outubro de 2021

1 2  9 0

UNIVERSIDADE D
COIMBRA

Sistema Robótico para a impressão direta *in situ* em órgãos

Rita Salomé Carvalho Gama

Coimbra, Outubro de 2021

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Sistema Robótico para a impressão direta *in situ* em órgãos

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, na especialização de Automação, orientada pelo Professor Doutor Rui Pedro Duarte Cortesão e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Orientador:
Prof. Dr. Rui Pedro Duarte Cortesão

Juri:
Prof. Dr. Mahmoud Tavakoli
Prof. Dr. Pedro Manuel Gens de Azevedo de Matos Faia
Prof. Dr. Rui Pedro Duarte Cortesão

Coimbra, Outubro de 2021

Agradecimentos

Em primeiro, quero prestar um agradecimento muito especial aos meus pais, António e Carla, por fazerem de mim o que sou hoje e porque sem eles nada disto era possível. Agradeço à minha irmã Inês e aos meus avós maternos por todo o apoio e pelo amor prestado durante o período de construção deste projeto.

Em segundo, queria expressar o meu profundo agradecimento ao meu orientador, Professor Doutor Rui Cortesão, por me ter concedido esta oportunidade, pelo seu acompanhamento, partilha de experiência e aconselhamento durante o desenrolar desta dissertação.

Também queria deixar um enorme agradecimento ao Engenheiro Hélio Ochoa, pela total disponibilidade, pela paciência e pela partilha de conhecimentos que me foi passando ao longo destes últimos meses.

Aos meus amigos de curso, um obrigada por todos os momentos vividos ao longo dos últimos 5 anos, obrigada pela amizade, pelo apoio, pelos momentos de entreajuda que foram fundamentais para ultrapassar os momentos menos fáceis.

Agradeço ainda a todos aqueles que, de forma direta ou indireta, contribuíram para a conclusão do meu percurso académico.

A todos vós, Muito Obrigada!

Resumo

A escassez de dadores de pele à volta do mundo representa um problema grave à escala mundial. Uma das promessas futuras no tratamento da regeneração de pele, consiste na utilização da bioimpressão 3D em detrimento de doadores. De forma a acelerar o procedimento do processo da bioimpressão 3D surgiu o conceito *in situ*.

Este projeto de dissertação consistiu em avaliar a performance do robô PANDA, desenvolvido pela *Franka Emika*, no processo de impressão direta *in situ* em órgãos.

Através do desenvolvimento de um algoritmo em *Matlab*, foi possível gerar áreas sobre a superfície de um braço para posteriormente serem percorridas pelo robô.

Inicialmente, foram realizados ensaios em espaço livre de modo a avaliar a arquitetura de controlo implementada, controlador de impedância, e o desempenho do robô PANDA, quer em ambiente de simulação quer em ambiente real.

Por último, foi testado a tarefa de impressão direta *in situ* sobre a superfície de um braço de um manequim.

Verificou-se que o controlador de impedância pode ser usado em tarefas de impressão direta *in situ*. Os resultados obtidos em ambiente de simulação e em ambiente real mostraram-se bastante promissores, tendo um erro médio de posição de 1 milímetro e de orientação um erro inferior a 1 grau.

Palavras − chave : Impressão Direta *in situ*, Robô PANDA, *Franka Emika*, Controlador de Impedância, Planeamento de Caminhos e Trajetórias, Nuvem de Pontos.

Abstract

The shortage of skin donors around the world poses a serious worldwide problem. One of the future promises in the treatment of skin regeneration is the use of 3D bioprinting in detriment of donors. In order to speed up the procedure of 3D bioprinting process is, the *in situ* concept emerged.

This work consisted on the evaluation of the performance of the PANDA robot, developed by *Franka Emika*, for of *in situ* direct printing in organs.

Through the development of an algorithm in *Matlab*, it was possible to generate areas on the surface of an arm, later covered by the robot.

Initially, free space tests were performed to evaluate both the implemented control architecture and impedance controller, as work as the performance of the PANDA robot, either in simulation and real environment.

Finally, the process or *in situ* direct printing was tested on the surface of a manikin's arm.

In conclusion, the implemented control can be used in printing tasks. The results obtained in simulation environment and in real environment were very promising, with an average position error of 1 millimetre and an orientation error of less than 1 degree.

Keywords : *In situ* Direct Printing, Panda Robot, *Franka Emika*, Impedance Controller, Path and Trajectory Planning, Point Cloud.

"Eles não sabem, nem sonham
Que o sonho comanda a vida
E que sempre que um homem sonha
O mundo pula e avança
Como bola colorida
Entre as mãos de uma criança"
António Gedeão

Conteúdo

Lista de Abreviaturas

3D 3 Dimension

CAD Computer Aided Design

DH Denavit-Hartenberg

DOF Degrees of Freedom

EL Euler-Lagrange

FCI Franka Control Interface

KDL Kinematics and Dynamics Library

LED Light Emitting Diode

LiDAR Light Detection And Ranging

MTH Matriz Transformação Homogénea

NE Newton-Euler

PC Personal Computer

PRY Pitch Roll and Yaw

ROS Robotic Operation System

URDF Unified Robot Description Format

Lista de Figuras

Lista de Tabelas

1

Introdução

1.1 Contexto e Motivação

A escassez de dadores de pele à volta do mundo para doentes com feridas graves levou à necessidade de criação de novos procedimentos que substituam estes dadores ao mesmo tempo que originam um tratamento rápido e eficaz. O surgimento de componentes artificiais para a substituição da pele tem-se revelado uma ótima alternativa. No artigo *"A Brief Review on 3D Bioprinted Skin Substitutes"*, publicado em 2020 [1], é referida a bioimpressão 3D como uma tecnologia pioneira que se tem revelado bastante eficiente quando se trata da impressão de tecidos para feridas extensas. Uma das grandes vantagens da bioimpressão é ter alta precisão e poder ser um processo de produção altamente personalizado. No entanto, esta tecnologia, apesar de ser bastante promissora, apresenta algumas limitações. Em *"Essential steps in bioprinting: From pre- to post-bioprinting"* [2] é referido que todo o processo de obtenção de construção de tecido humano, utilizando a bioimpressão 3D, envolve vários estágios que passam pelo tecido ser primeiramente impresso dentro de uma bioimpressora, sendo posteriormente passado por um bioreactor para maturação e, só no fim, é que é transferido para o local desejado. Todos esses passos em conjunto tornam o processo de bioimpressão lento e demorado. Como alternativa, surgiu o conceito *in situ*.

A impressão *in situ* é uma técnica promissora de reparação de lesões que pode ser aplicada diretamente na ferida durante a operação cirúrgica, ou seja, os materiais são impressos diretamente na zona afetada reduzindo assim o tempo do processo. No documento *"Development of a Robotic Arm Basead Hydrogel Additive Manufacturing System for In-Situ Printing"* [3] é mensionada a impressão *in situ* como uma técnica de última geração que pode atuar diretamente no local danificado de acordo com as suas características geométricas. Em comparação com a bioimpressão 3D, a impressão *in situ* torna-se mais aplicável à reparação de danos devido ao facto de estes se cingirem normalmente a superfícies curvas ou até mesmo geometrias mais complexas, enquanto que a bioimpressão 3D está limitada a substratos planos.

Com essa ideia em mente surgiu o projeto *Print On Organs* que consiste, no desenvolvimento de um braço robótico para a realização da impressão *in situ* de materiais, diretamente em tecidos danificados de forma customizada e minimamente invasiva.

1.2 Desenvolvimento da robótica colaborativa

Semelhante ao desenvolvimento da bioimpressão 3D, outras áreas da ciência e da tecnologia evoluíram, contribuindo para o melhoramento da robótica colaborativa em vários setores da sociedade.

A utilização da robótica revolucionou muitas indústrias, desde da indústria dos automóveis à indústria alimentar. Hoje em dia, a maioria das indústrias utiliza algum tipo de ferramenta robótica para automatizar procedimentos e aumentar a sua eficiência. Um dos problemas associados aos manipuladores robóticos industriais era a sua incapacidade de trabalhar ao lado do Homem devido a questões de segurança, o que provocava uma desconfiança nos setores industriais.

Esta limitação, juntamente com o desenvolvimento da tecnologia automóvel e elétrica, motivou o desenvolvimento de uma nova geração de manipuladores robóticos e arquiteturas de controlo, que culminou num novo ramo da robótica chamado robótica colaborativa.

Os manipuladores robóticos colaborativos são mais pequenos e mais leves do que os seus homólogos industriais. Possuem atuadores elétricos menos ruidosos e menos volumosos contribuindo dessa maneira para o desenvolvimento de robôs mais pequenos e ambientes de trabalhos mais limpos.

Segundo R. Bloss em *"Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications"* [4], o fator chave para o sucesso dos robôs colaborativos são as arquiteturas de controlo utilizadas. Através da utilização de modelos dinâmicos mais precisos, pode-se ter em consideração as forças de interação do robô com o ambiente. Permitindo desta maneira, controlar o robô e torná-lo mais suave quando interage com o Homem.

Os robôs colaborativos também tiveram impacto no campo medicinal. A primeira utilização de um robô neste campo foi no final dos anos 80, quando o robô PUMA foi utilizado para orientar uma agulha durante uma biopsia ao cérebro [5].

A partir desse dia, começaram a surgir novas investigações em sistemas médicos através da robótica colaborativa.

Mais recentemente, em *"Robotic in situ 3D bio-printing technology for repairing large segmental bone defects"* [6] é utilizado um manipulador robótico colaborativo para a reconstrução óssea através de componentes artificiais. O percurso a ser percorrido durante a impressão 3D foi planeado através um *software* de corte de fonte aberta, *Repetier Host*.

Em suma, a robótica colaborativa através de novos manipuladores e novas arquiteturas de controlo, traz uma nova dinâmica a todas as indústrias ao mesmo tempo que permite a coexistência entre o Homem e o robô no mesmo espaço de trabalho e mais importante, permite a colaboração entre o robô e o Homem.

1.3 Objetivos

Antes de mais, este trabalho consiste em encontrar e desenvolver estratégias que possam vir a ser úteis no processo de impressão direta *in situ*. Por esta razão, o braço robótico PANDA irá ser testado para a realização da impressão direta *in situ* em órgãos. Para isso, é essencial estudar as capacidades e limitações do robô pretendido.

Primeiro, será necessário compreender e estudar a arquitetura de controlo por impedância de modo a evitar colisões indesejadas entre o robô e o ambiente que o rodeia. Seguidamente, é necessário analisar e implementar diferentes tipos de trajetórias e caminhos através de uma qualquer nuvem de pontos fornecida para podermos alcançar o objetivo principal.

Finalmente, é necessário realizar uma demonstração do trabalho desenvolvido.

1.4 Contribuições

O trabalho apresentado dá a possibilidade de testar diferentes tipos de trajetórias no simulador Gazebo, bem como no robô PANDA.

Além disso, oferece uma solução de baixo custo em processos de impressão e portabilidade, devido ao peso reduzido do PANDA.

Por último, abre portas para uma implementação de uma arquitetura de controlo robótico para a impressão direta *in situ* em órgãos.

1.5 Organização do documento

- **Capítulo 1:** Antecedentes, objetivos, contribuições e estrutura do documento;
- **Capítulo 2:** Visão geral do sistema robô PANDA, os possíveis modos de comunicação com o robô e apresentação do *Franka-ROS*;
- **Capítulo 3:** Conceitos teóricos da cinemática robótica, cinemática diferencial e dinâmica.
- **Capítulo 4:** Arquitetura de controlo de impedância implementada no simulador e no verdadeiro robô PANDA;
- **Capítulo 5:** Planeamento de trajetórias. Explicação do funcionamento do algoritmo implementado para a geração de caminhos a serem percorridos;
- **Capítulo 6:** Análise dos ensaios experimentais realizados no simulador Gazebo e no robô real;
- **Capítulo 7:** Demostração da tarefa de impressão direta *in situ* no simulador Gazebo e no robô PANDA;

- **Capítulo 8:** Conclusões finais e sugestões de trabalhos futuros.

2

Braço Robótico PANDA

2.1 Visão geral do sistema

O robô PANDA é um braço robótico colaborativo desenvolvido pela *Franka Emika* [7], uma empresa Alemã sedeada em Munique focada no desenho e desenvolvimento de robôs de alto desempenho.

A versão usada neste projeto é o PANDA Research, que permite ao utilizador ter um controlo direto assim como a possibilidade de programação do mesmo. Esta versão também permite desenvolver uma interface do robô com sensores externos. O robô é constituído por 7 graus de liberdade (DOF) com uma garra removível no seu setor final (*end-effector*) (cf. figura 2.1).

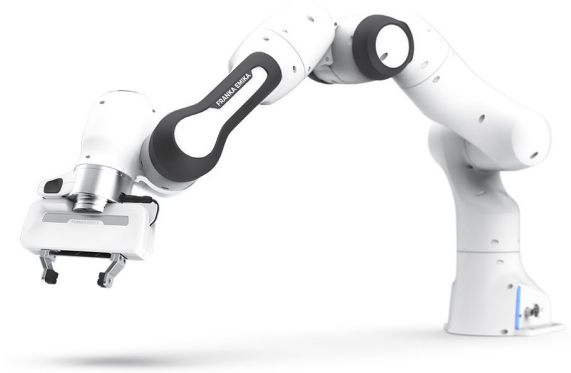


Figura 2.1: Braço robótico PANDA com 7 DOF.

O robô PANDA incorpora as mesmas características que um robô industrial, com um desvio de caminho praticamente inexistente permitindo assim, que a execução do movimento seja precisa, robusta e rápida nos processos de fabricação. O anexo A descreve em pormenor todas as especificações do robô [8].

A figura 2.2 mostra a configuração básica do sistema do robô. Nela é possível observar que o robô além de ser constituído por um braço robótico também é constituído por um *hardware* de controlo, um *PC Host* para conectá-lo via *Desk* e por dispositivos de segurança.

O braço robótico é a parte principal do sistema e é constituído por 7 juntas de rotação, o que torna o robô redundante para qualquer tarefa do espaço 3D. É importante mencionar que na base do robô existem 2 luzes LED para indicar o estado em que se encontra o robô e na parte superior existe um conjunto de botões ao qual chamamos "Piloto". Estes funcionam como uma plataforma de controlo e permitem ao utilizador manipular manualmente o robô. Como mencionado anteriormente, o sistema é também composto por um computador responsável por controlar o braço robótico e estes estão fisicamente conectados através de um cabo de controlo.

Neste sistema está incluído um *PC Host* que basicamente é um computador que o operador usa para aceder à interface e executar o código dos controladores personalizados.

Por último, o robô é constituído por 3 dispositivos de segurança que permitem ao utilizador parar o robô. São eles, um dispositivo de paragem de emergência, que é um botão com dois estados que quando pressionado funciona como um interruptor aberto e quando é libertado fecha e o controlador é energizado. Um dispositivo de ativação externa (*External activation device*), que se comporta de forma semelhante ao dispositivo de paragem de emergência, quando pressionado o robô pára e permanece no modo interativo e quando liberto o robô entra em modo ativo. Caso o botão seja pressionado quando o robô está a realizar qualquer tipo de movimento, este irá parar imediatamente. Finalmente, o dispositivo de inativação externa (*External enabiling devide*) é usado para ignorar o dispositivo de ativação (*External activation device*). Se o botão for pressionado até metade, irá parar o modo ativo e as tarefas podem ser executadas. O robô irá parar se não houver qualquer pressão neste botão ou se este for totalmente pressionado [8, 9].

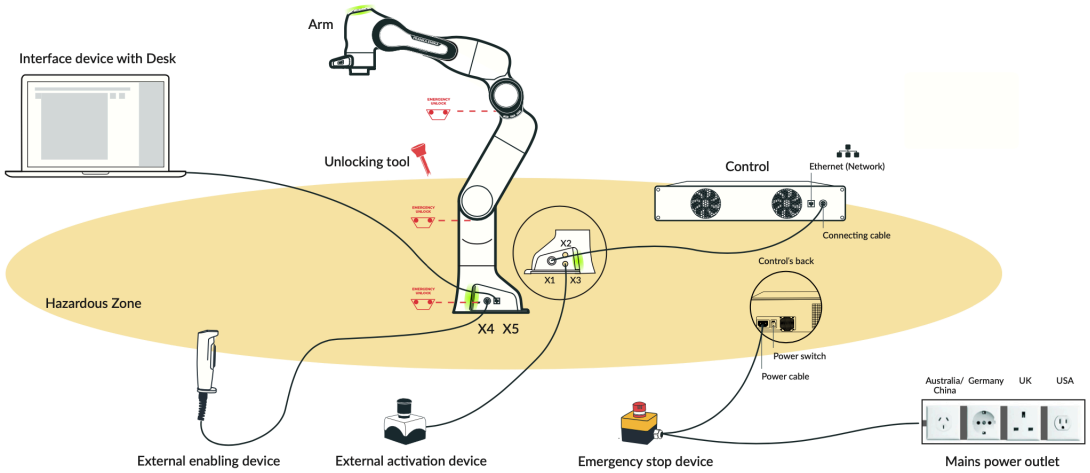


Figura 2.2: Configuração básica do sistema robótico.

2.2 Modos de Comunicação

Uma das formas de comunicar com o robô oferecidas pela *Franka Emika*, é através da utilização da aplicação *web*, *Panda Desk* [10]. Para estabelecer a comunicação entre a

aplicação e o robô, basta ligar ao robô um cabo de internet. O *Panda Desk* permite criar tarefas, que são sequências de programas e consistem em sequências cronológicas de aplicações. As aplicações são blocos de construção de tarefas e descrevem as capacidades básicas do robô, tais como *"grip"*, *"put down"*, *"push button"*, entre outros. Os aplicativos de uma tarefa precisam de ser parametrizados, o que significa que parâmetros como a pose, orientação e velocidade precisam de ser definidos. Uma descrição mais complexa da *Panda Desk* pode ser encontrada na secção 3 *"Further user instruction for Panda system"* em *"PANDA'S Instruction Handbook"* [8].

Outra forma de comunicação é através da *Franka Control Interface* (FCI). A FCI permite uma ligação bidirecional rápida e direta de baixo nível ao braço robótico. Fornece o estado atual do robô e permite o controlo direto com uma estação de trabalho externa ligada através da internet. Através da *libfranka*, a interface de código aberto em C++, é possível enviar valores de controlo em tempo real a $1kHz$ com 5 interfaces diferentes [11].

Para além disso, através do *franka_ros* é possível ligar o robô a todo o ecossistema do Sistema Operacional Robótico (ROS), que integra a *libfranka* no controlador ROS, assim como os modelos URDF e malhas de superfície 3D (*meshes*) detalhadas dos robôs da *Franka Emika*, o que permite visualizações no *RViz* e simulações cinemáticas. Por último, a integração do *MoveIt* é importante uma vez que facilita a movimentação do robô [12].

O modo de comunicação escolhido neste trabalho foi a FCI com recurso ao *franka_ros*, uma vez que o ROS é uma estrutura flexível para escrever *software* robotizado e simplifica a tarefa de criar um comportamento robótico complexo e robusto através de uma grande variedade de plataformas robóticas.

2.3 Franka-ROS

2.3.1 Descrição

Como já foi referido anteriormente, o pacote *franka_ros* integra a biblioteca *libfranka* no controlador ROS. Ele é composto pelos seguintes pacotes:

- *franka_description*: Este pacote contém o formato da descrição unificada do robô (URDF), que é o formato padrão do ROS XML para descrever modelos do robô e malhas de superfícies (*meshes*).
- *franka_gripper*: Este pacote implementa a garra desenhada pela *Franka* para a sua utilização em ROS.
- *franka_hw*: Este pacote fornece a interface de *hardware* para a utilização do robô com o *ros_control* baseado na *libfranka*.
- *franka_control*: Este pacote fornece os nós *franka_control_node* e *franka_control_node* que são nós de *hardware* que servem para controlar o robô.

- *franka_visualization*: Este pacote contém as ferramentas de visualização que permitem visualizar o robô no *RViz*.
- *franka_example_controllers*: Este pacote contém implementado um conjunto de controladores exemplos que permitem controlar o robô através do ROS.
- *franka_msgs*: Este pacote contém mensagens, serviços e todos os tipos de ação que são utilizados nos pacotes *franka_hw* e *franka_control* permitindo assim, publicar o estado em que o robô se encontrar.
- *panda_moveit_config*: Este pacote permite a interação com o ROS *MoveIt*.

2.3.2 Simulador Gazebo

O *franka_ros* não vêm com nenhum tipo de simulador. No entanto, em projetos anteriores desenvolvidos na Universidade de Coimbra foi criado um pacote capaz de usufruir do robô PANDA em ambiente de simulação Gazebo, *franka_simulation*.

Um simulador torna possível testar rapidamente algoritmos, desenhar robôs, realizar testes de regressão, treinar sistemas usando cenários realistas, para além de possibilitar realizar trabalho fora do laboratório. Na figura 2.3 é possível observar o robô PANDA em ambiente de simulação.

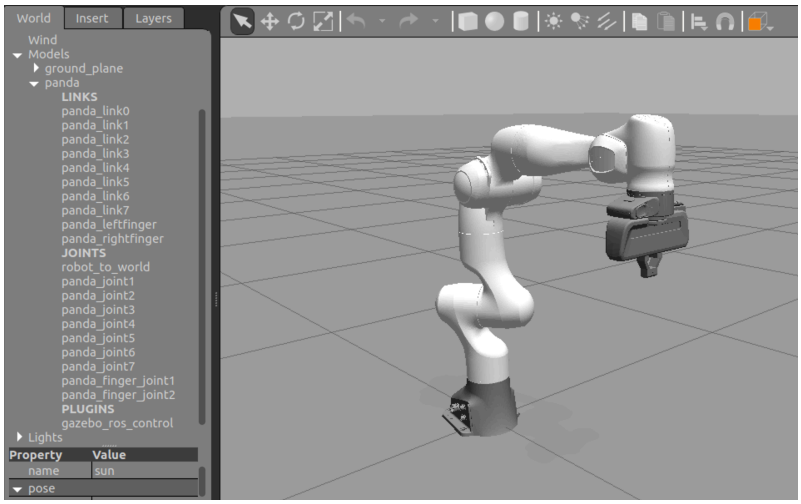


Figura 2.3: Braço robótico PANDA no simulador Gazebo.

Para controlar o modelo do robô no simulador Gazebo é necessário utilizar o pacote *ros_control*. Nele, estão disponíveis três tipos diferentes de controladores: controlador de esforço, controlador de posição e controlador de velocidades. Os ficheiros de configuração para controlar as juntas através do pacote *ros_control* estão disponíveis no pacote *franka_control*. O controlador de trajetórias foi igualmente adicionado para fornecer uma interface para o *MoveIt*.

A figura 2.4 mostra uma visão geral da relação entre o simulador, *hardware*, controla-

dores e transmissões.

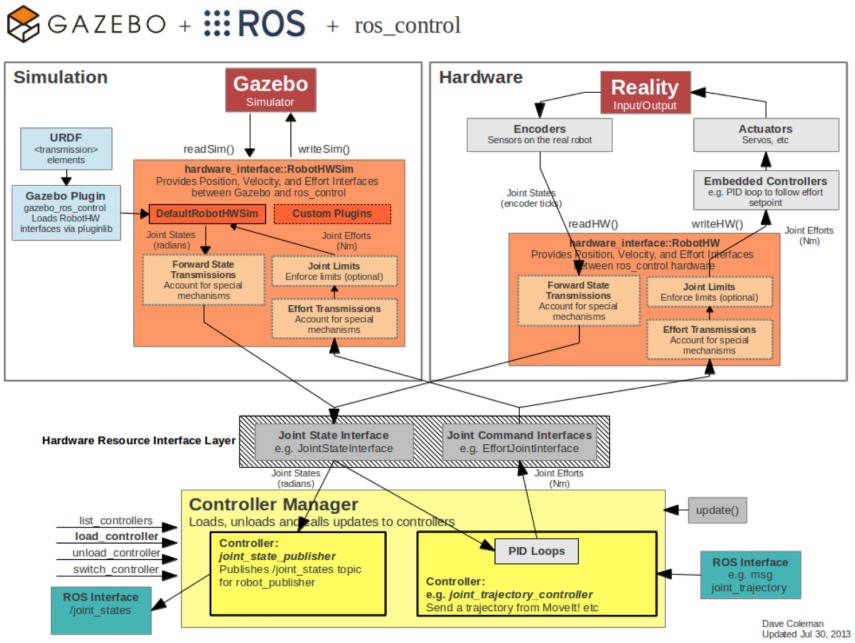


Figura 2.4: Fluxo de dados entre o pacote *ros_control* e o simulador Gazebo.

Mais informações de como funciona o pacote *ros_control* e como ele se relaciona com o simulador Gazebo podem ser consultados nos sites ROS *community* [13, 14].

2.3.3 MoveIt

Conforme foi referido na subsecção 2.3.1 o pacote *franka_ros* fornece um pacote *MoveIt*, *panda_moveit_config*. Este pacote permite lançar o *MoveIt* com um robô virtual, sendo útil para a visualização e teste, ou até mesmo lançar o *MoveIt* com o nó *franka_control* que controla o robô real.

A figura 2.5 mostra a janela do *RViz* com o *plugin MoveIt*. Nele é possível movimentar o *end-effector* do robô através dos marcadores interativos. Enquanto o *end-effector* estiver a ser movimentado, o *MoveIt* executa a cinemática inversa de modo a que as posições das juntas sejam atualizadas. Na parte destacada a vermelho da figura 2.5 é possível planear e executar uma determinada trajetória. Ao clicar no botão *Plan*, o *MoveIt* visualiza a trajetória planeada e analisa se esta é possível de ser concretizada. Caso seja possível, basta clicar no botão *Execute* e a trajetória é executada com sucesso.

Se o robô estiver conectado, ele move-se de acordo com a trajetória visualizada. Caso esteja o robô virtual conectado as articulações do modelo movem-se para o ponto desejado. Além disso, o *plugin MoveIt RViz* também pode ser utilizado para adicionar obstáculos e evitar parâmetros de planeamento.

Todas as configurações sobre o *MoveIt* e *RViz* e de como eles funcionam podem ser

esclarecidas através dos sites do ROS *community* [15, 16].

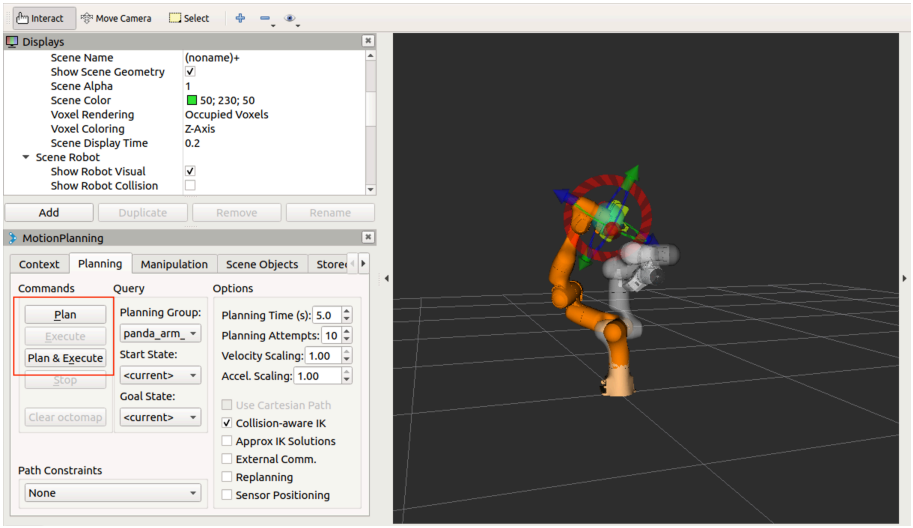


Figura 2.5: Braço robótico PANDA no *plugin MoveIt RViz*.

3

Cinemática e Dinâmica do robô PANDA

Este capítulo centra-se na derivação da cinemática direta, cinemática diferencial e dinâmica do braço robótico PANDA.

Para calcular a cinemática e a dinâmica do PANDA é utilizada uma biblioteca de cinemática e dinâmica (KDL), fornecida pelo *Orocos Project* [17]. A KDL fornece soluções para calcular qualquer coisa desde a cinemática de posição avançada até a dinâmica inversa. Ela inclui suportes para construir uma cadeia KDL a partir de ficheiros URDF. É importante reforçar, que o *franka_ros* inclui um pacote *franka_description* que contém os modelos URDF do PANDA. Isto inclui a geometria física, bem como as propriedades cinemáticas e dinâmicas do braço robótico. Para informações mais detalhadas sobre os ficheiros URDF pode-se consultar a página ROS URDF *wiki* [18].

Devido ao facto da cinemática e dinâmica serem calculadas através da KDL, o enquadramento teórico será cedido nas próximas secções.

3.1 Cinemática Direta

Um manipulador robótico é um dispositivo eletromecânico composto por uma sequência de corpos unidos através de articulações ao qual dá-se o nome de juntas. As juntas podem ser de dois tipos: rotação ou prismáticas. Na extremidade do manipulador pode ser acoplado uma ferramenta para realizar uma determinada tarefa, designada *end-effector*. A estrutura mecânica de um manipulador é caracterizada pelos seus graus de liberdade, DOF, que determinam a postura do manipulador.

A descrição de cinemática do manipulador é um dos aspetos mais importantes do desenvolvimento, análise, controlo, e simulação na robótica, visto que mecanismos robóticos tem na sua essência o movimento no espaço.

A cinemática direta tem como objetivo calcular a *pose*[1] do *end-effector* relativamente ao referencial da base, em função das variáveis de junta.

[1] A *pose* do end-effector equivale à sua posição e orientação

O cálculo da cinemática direta de manipuladores robóticos requer a definição de um conjunto de eixos coordenados desde a base, até ao seu *end-effector*. Para tal, recorre-se à convenção de *Denavit Hartenberg* (DH).

A convenção de DH pode ser interpretada como um conjunto de regras que permitem calcular de forma sistemática a cinemática direta de manipuladores robóticos.

Após ser estabelecido o referencial para cada elo do manipulador, há que definir os parâmetros DH respetivos, para cada uma das juntas do manipulador. Os parâmetros DH são normalmente representados por a_i, d_i, θ_i e α_i e descrevem as translações e rotações necessárias para que referências correspondentes a dois elos consecutivos coincidam.

A posição e orientação do referencial i relativamente ao referencial $i-1$, é obtido através da matriz transformação homogénea (MTH), T_i^{i-1}, que é dada por:

$$T_i^{i-1}(q) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

A função que descreve a cinemática direta do manipulador é conseguida a partir da multiplicação das MTH que relacionam as n juntas do manipulador entre si, desde da base do elemento robótico até ao seu *end-effector*.

$$T_e^b(q) = T_0^b\left(T_1^0\cdots T_n^{n-1}\right)T_e^n \tag{3.2}$$

As regras de convenção de DH podem ser encontras no livro *"Robotics: Modeling, Planning and Control"* [19].

Na figura 3.1 é possível observar a cadeia cinemática do robô PANDA e na tabela 3.1 os parâmetros DH. Mais informações sobre os parâmetros do robô PANDA podem ser consultadas no documento online *"Franka Control Interface"* [11].

Tabela 3.1: Parâmetros de DH do robô PANDA.

Junta	$a\,(m)$	$d\,(m)$	$\alpha\,(rad)$	$\theta\,(rad)$
Junta 1	0	0.333	0	θ_1
Junta 2	0	0	$-\frac{\pi}{2}$	θ_2
Junta 3	0	0.316	$\frac{\pi}{2}$	θ_3
Junta 4	0.0825	0	$\frac{\pi}{2}$	θ_4
Junta 5	-0.0825	0.384	$-\frac{\pi}{2}$	θ_5
Junta 6	0	0	$\frac{\pi}{2}$	θ_6
Junta 7	0.088	0	$\frac{\pi}{2}$	θ_7
Último setor	0	0.107	0	0

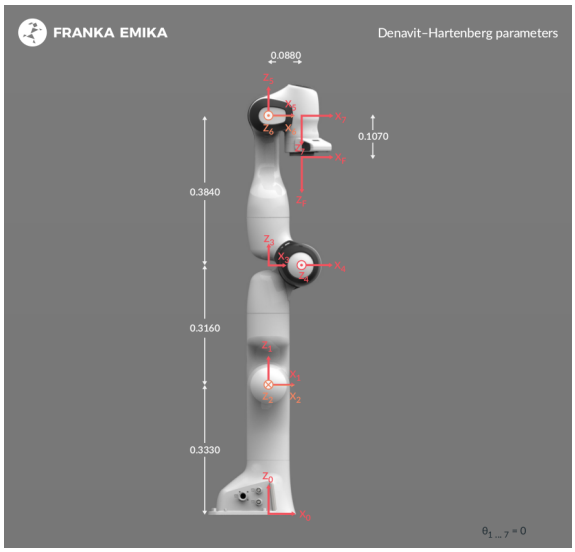


Figura 3.1: Cadeia cinemática do robô PANDA.

3.2 Cinemática Diferencial

[2]A cinemática diferencial estabelece a relação entre as velocidades das juntas e a correspondente velocidade linear e angular do *end-effector*. Este mapeamento é descrito por uma matriz jacobiana, J, que depende da configuração do manipulador. O Jacobiano é uma das ferramentas mais importantes para a caracterização de manipuladores. É usado para encontrar singularidades, analisar redundâncias, determinar algoritmos de cinemática inversa e descrever o mapeamento entre as forças aplicadas no *end-effector* e os torques resultantes das juntas.

Inicialmente, é necessário exprimir a velocidade linear, $\dot{p}_e$, e a velocidade angular, w_e, do *end-effector* em função das velocidades das juntas, $\dot{q}$. Essa relação é dada por:

$$\begin{cases} \dot{p}_e = J_P(q)\,\dot{q} \\ w_e = J_O(q)\,\dot{q} \end{cases} \tag{3.3}$$

onde J_P é uma matriz $(3 \times n)$ que relaciona as velocidades das juntas, $\dot{q}$, com a velocidade linear, $\dot{p}_e$, do *end-effector*, enquanto que, J_O é uma matriz $(3 \times n)$ que relaciona as velocidades das juntas, $\dot{p}_e$, com a velocidade angular, w_e, do *end-effector*. Numa forma mais compacta, a equação 3.3 pode ser dada por:

$$v_e = \begin{bmatrix} \dot{p}_e \\ w_e \end{bmatrix} = J(q)\,\dot{q} \tag{3.4}$$

[2]Todo o conteúdo teórico veiculado neste subcapítulo está de acordo com: Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani and Giuseppe Oriolo. Robotics: *Robotics: Modeling, Planning and Control*. Springer, 2009 [19].

que representa a equação da cinemática diferencial do manipulador. J é uma matriz $(6 \times n)$ que retrata o manipulador geométrico Jacobiano em função das variáveis das juntas.

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix} \tag{3.5}$$

Posteriormente, para calcular o Jacobiano, é preciso escrever as equações da velocidade linear, $\dot{p}_e$, e da velocidade angular, w_e.

$$\begin{cases} \dot{p}_e = \sum_{i=1}^{n} \frac{\partial_{p_e}}{\partial_{q_i}} \dot{q}_i = \sum_{i=1}^{n} J_{P_i} \dot{q}_i \\ w_e = \sum_{i=1}^{n} w_{i-1,i} = \sum_{i=1}^{n} J_{O_i} \dot{q}_i \end{cases} \tag{3.6}$$

A velocidade das juntas, $\dot{q}_i$, pode ser escrita como uma junta de rotação ou como uma junta prismática. Para conseguirmos distinguir uma da outra, escreve-se $(q_i = d_i)$ para uma junta prismática e $(q_i = \vartheta_i)$ no caso de ser uma junta de rotação.

A contribuição para a velocidade linear é dada por:

- Se a junta i for prismática:

$$\dot{q}_i J_{P_i} = \dot{d}_i z_{i-1} \Leftrightarrow J_{P_i} = z_{i-1} \tag{3.7}$$

- Se a junta i for rotação:

$$\dot{q}_i J_{P_i} = w_{i-1,i} \times r_{i-1,e} = \vartheta_i z_{i-1} \times (p_e - p_{i-1}) \Leftrightarrow J_{P_i} = z_{i-1} \times (p_e - p_{i-1}) \tag{3.8}$$

onde, p_e é a distância desde a origem do referencial do *end-effector* até á origem do referencial da base e p_{i-1} é distancia análoga do elo $i-1$.

A contribuição para a velocidade angular é dada por:

- Se a junta i for prismática:

$$\dot{q}_i J_{O_i} = 0 \Leftrightarrow J_{O_i} = 0 \tag{3.9}$$

- Se a junta i for rotação:

$$\dot{q}_i J_{O_i} = \dot{\vartheta}_i z_{i-1} \Leftrightarrow \dot{q}_i J_{O_i} = z_{i-1} \tag{3.10}$$

O robô PANDA apresenta 7 juntas de rotação, de tal modo que o Jacobiano é dado por:

$$J = \begin{bmatrix} J_{P_1} & \cdots & J_{P_n} \\ J_{O_1} & \cdots & J_{O_n} \end{bmatrix} = \begin{bmatrix} J_{P_i} \\ J_{O_i} \end{bmatrix} = \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{3.11}$$

Por último é importante apontar que, a matriz jacobiana depende da forma em que se exprime a velocidade do *end-effector*. As equações acima permitem calcular o Jacobiano geométrico relativamente ao referencial base. Caso se pretenda representar o Jacobiano num referencial diferente, u, basta conhecer a matriz de rotação relativa, R^u. A relação entre as velocidades é:

$$\begin{bmatrix} \dot{p}_e^u \\ w_e^u \end{bmatrix} = \begin{bmatrix} R^u & 0 \\ 0 & R^u \end{bmatrix} \begin{bmatrix} \dot{p}_e \\ w_e \end{bmatrix} \tag{3.12}$$

que substituído na equação 3.4, fica igual a:

$$\begin{bmatrix} \dot{p}_e^u \\ w_e^u \end{bmatrix} = \begin{bmatrix} R^u & 0 \\ 0 & R^u \end{bmatrix} J\dot{q} \tag{3.13}$$

e depois,

$$J^u = \begin{bmatrix} R^u & 0 \\ 0 & R^u \end{bmatrix} J \tag{3.14}$$

onde, J^u é o Jacobiano geométrico no referencial u.

3.3 Dinâmica

Ao contrário da cinemática, a dinâmica descreve a relação entre o movimento do manipulador e as forças e momentos que o causaram. A obtenção de modelos dinâmicos de manipuladores é preponderante quando se pretende simular movimentos ou projetar controladores. Modelos dinâmicos, permitem testar e analisar algoritmos de controlo sem a presença física de um manipulador robótico precavendo assim, a ocorrência de situações indesejáveis. As equações dinâmicas são incorporadas pelos parâmetros dinâmicos do robô como a massa, a inércia, as fricções e outros parâmetros desconhecidos que podem afetar negativamente o desempenho do robô. Além disso, as equações referidas permitem a possibilidade de gerar arquiteturas de controlo adicionais quer no espaço das juntas quer no das tarefas.

Na literatura, encontram-se expostas diversas formas de obter modelos dinâmicos de manipuladores robóticos. Destacam-se como as mais comuns o método de *Euler-Lagrange*

(EL) e o método recursivo de *Newton-Euler* (NE). O método de EL baseia-se numa abordagem energética, enquanto que o método NE baseia-se no balanço das forças.

O método usado neste trabalho para determinar a dinâmica do robô foi o EL, por essa razão será descrito em pormenor.

3.3.1 Método de Euler-Lagrange

[3]A mecânica de *Lagrange* é uma reformulação das leis de *Newton* que tem em conta as relações energéticas.

O *Lagrange*, L, é definido por:

$$L\left(q,\dot{q}\right) = E\left(q,\dot{q}\right) - U\left(q\right) \tag{3.15}$$

onde, $E\left(q,\dot{q}\right)$ e $U\left(q\right)$ são escalares que representam respetivamente a energia cinética e energia potencial do robô, sendo q um vetor $(n \times 1)$ de coordenadas generalizadas, que representam a posição das juntas.

A energia cinética é dada por:

$$E\left(q,\dot{q}\right) = \frac{1}{2}\dot{q}^{T} M\left(q\right)\dot{q} \tag{3.16}$$

onde, $M\left(q\right)$ é a matriz de massa $(n \times n)$ simétrica e positiva. Já a energia potencial é dada por:

$$U\left(q\right) = -g_{c}^{T} r_{c}\left(q\right) m_{c} \tag{3.17}$$

onde, g_c é o vetor (3×1) da aceleração da gravidade, r_c é a matriz $(3 \times n)$ do centro de gravidade das juntas e m_c é o vetor $(n \times 1)$ da massa das juntas.

A mecânica de *Lagrange* é representada da seguinte forma:

$$\tau = \frac{d}{dt}\frac{\partial L\left(q,\dot{q}\right)}{\partial \dot{q}} - \frac{\partial L\left(q,\dot{q}\right)}{\partial q} \tag{3.18}$$

onde, τ é o torque generalizado que atua sobre q.

Pelas equações 3.15, 3.16 e pela equação 3.18 obtêm-se,

$$\frac{d}{dt}\left(M\left(q\right)\dot{q}\right) - \frac{\partial E\left(q,\dot{q}\right)}{\partial q} + \frac{\partial U\left(q,\dot{q}\right)}{\partial q} = \tau \tag{3.19}$$

[3]Todo o conteúdo teórico veiculado nesta secção está de acordo com: Rui Cortesão.*Medical Robotics*. University of Coimbra, 2019/2020 [20].

Podendo, a equação 3.19 ser rescrita como:

$$M\left(q\right)\ddot{q}+\upsilon\left(q,\dot{q}\right)+g\left(q\right)=\tau \tag{3.20}$$

onde, $\upsilon\left(q,\dot{q}\right)$ é o termo de Coriolis e das forças centrípetas e $g\left(q\right)$ é o termo relacionado com a força da gravidade.

A matriz da massa (ou inércia), $M\left(q\right)$, é dada por:

$$M\left(q\right)=\sum_{i=1}^{n}\left[m_{i}J_{\upsilon_{i}}\left(q\right)^{T}J_{\upsilon_{i}}\left(q\right)+J_{w_{i}}\left(q\right)^{T}R_{i}\left(q\right)I_{i}R_{i}\left(q\right)^{T}J_{w_{i}}\left(q\right)\right] \tag{3.21}$$

onde, m_i, $J_{\upsilon_i}\left(q\right)$, $J_{w_i}\left(q\right)$, $R_i\left(q\right)$ e I_i são respetivamente a massa de cada junta, os Jacobianos associados à velocidade linear e angular, a matriz de rotação nas coordenadas da base e o tensor de inércia nas coordenadas de cada junta. É importante referir que todas estas variáveis são referidas ao centro de massa da junta i.

O termo de Coriolis e das forças centrípetas, $\upsilon\left(q,\dot{q}\right)$, é dado por:

$$\upsilon\left(q,\dot{q}\right)=C\left(q,\dot{q}\right)=\dot{M}\left(q\right)\dot{q}-\frac{\partial E\left(q,\dot{q}\right)}{\partial q} \tag{3.22}$$

onde, $C\left(q,\dot{q}\right)$ é a matriz de Coriolis com dimensão $(n\times 1)$ dada por:

$$C\left(q,\dot{q}\right)=\dot{M}\left(q\right)-\frac{1}{2}\dot{q}^{T}\frac{\partial M\left(q\right)}{\partial q} \tag{3.23}$$

Por último, o termo de gravidade $g\left(q\right)$, vetor de dimensão $(n\times 1)$, é representado por:

$$g\left(q\right)=\frac{\partial U\left(q\right)}{\partial q} \tag{3.24}$$

4

Arquitetura de Controlo

Apesar do braço robótico mover-se no espaço livre durante o processo de impressão, existe a possibilidade, mesmo que pequena, de haver algum contacto indesejado com o manipulador. Para tal, existe a necessidade de implementar algumas técnicas de controlo para prevenir esses contactos. É através de uma arquitetura de controlo que se consegue gerir o movimento do robô juntamente com as forças de interação, portanto é necessário implementar um modelo matemático que as relacione. O modelo dinâmico do robô fornece essa relação.

A estratégia de controlo abordada é um controlo de impedância no espaço de tarefa com otimização da postura.

Neste capítulo, aborda-se primeiramente as variáveis do modelo dinâmico do manipulador para controlo de impedância e otimização da postura, e de seguida o controlo de impedância cartesiano com otimização da postura e o controlador de orientação.

4.1 Modelo dinâmico do manipulador PANDA

[1]Como foi referido na secção 3.3 a dinâmica de um robô é dada por:

$$M(q)\ddot{q} + \upsilon(q, \dot{q}) + g(q) = \tau \tag{4.1}$$

- $M(q)$ é a matriz da massa $(n \times n)$;
- n é o número de DOF;
- $\upsilon(q, \dot{q})$ representa as matrizes de Coriolis e da força centrípeta $(n \times 1)$;
- $g(q)$ é o termo de gravidade $(n \times 1)$;
- τ é o torque generalizado que atua sobre q.

[1]Todo o conteúdo teórico veiculado neste subcapítulo está de acordo com: H. Ochoa and R. Cortesao. *"Control Architecture for Robotic-Assisted Polishing Tasks Based on Human Skills"* em IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, vol. 2019-Octob, IEEE, Oct. 2019, pp. 630–637 [21].

4.1.1 Linearização do feedback não linear

O manipulador referente à equação 4.1 é um sistema não linear. Através do artigo *"Computed-Torque Control for Robotic-Assisted Tele-Echography Based on Perceived Stiffness Estimation"* [22] fica-se a saber que ao conhecer o modelo dinâmico do manipulador é possível alcançar um sistema linear e desacoplado através da linearização do feedback não linear.

O torque, τ, do manipulador que atua sobre q é dado por:

$$\tau = \tau_c + \tau_f + \tau_m \tag{4.2}$$

onde, τ_c é o troque referente ao atuador, τ_f é o torque alusivo à fricção e τ_m o torque relativo à interação. Se forem ignoradas as fricções das juntas do manipulador ($\tau_f \approx 0$), define-se τ_m como:

$$\tau_m = J^T(q) F_m \tag{4.3}$$

onde, F_m é a medição das forças no *end-effector* e J^T é a matriz jacobiana associada à transposição. O torque referente ao atuador, τ_c, pode ser definido:

$$\tau_c = C(q, \dot{q}) + g(q) - \tau_m + \tau_c^{'} \tag{4.4}$$

Através das equações 4.1 e 4.4 é possível reduzir o sistema a:

$$M(q)\,\ddot{q} = \tau_c^{'} \tag{4.5}$$

onde, $\tau_c^{'}$ representa o vetor do torque calculado pela lei de controlo.

A equação 4.5 pode ser representada no espaço de tarefa, ficando:

$$\Lambda_p(q)\,\ddot{x}_p - \Lambda_p(q)\,\dot{J}_p(q)\,\dot{q} = F_p \tag{4.6}$$

com,

$$\ddot{x}_p = J_p(q)\,\ddot{q} + \dot{J}_p(q)\,\dot{q} \tag{4.7}$$

ficando,

$$\Lambda_p(q) = \left(J_p(q)\,M^{-1}(q)\,J_p^T(q)\right)^{-1} \tag{4.8}$$

$$\tau_c^{'} = \tau_p = J_p^T(q) F_p \tag{4.9}$$

onde, $\wedge_p(q)$ e $J_p(q)$ são a inércia e a matriz jacobiana associada à tarefa x_p, e τ_p e F_p são respetivamente os vetores de torque e força calculada pelo controlador para executar x_p.

4.1.2 Controlo no espaço nulo

Diz-se que um sistema robótico é redundante quando o número de DOF da tarefa, m, é menor que o número de DOF do robô, n. Ou seja, quando um robô apresenta várias configurações para uma determinada posição do seu *end-effector* no espaço cartesiano.

Matematicamente falando, se $m < n$, o sistema é redundante ao executar x_p e $\tau_c^{'}$ não é único. A relação entre o torque $\tau_c^{'}$, e a força F_p, é dada por:

$$\tau_c^{'} = J_p^T(q) F_p + N_p^T(q) \tau_s \tag{4.10}$$

Esta relação permite a decomposição de $\tau_c^{'}$ em dois vetores desacoplados dinamicamente:

$$\tau_c^{'} = \tau_p + \tau_n \tag{4.11}$$

sendo, τ_p o torque que executa a tarefa principal (ver equação 4.9), e τ_n dado por:

$$\tau_n = N_p^T(q) \tau_s \tag{4.12}$$

o torque de uma tarefa secundária que não interfere com τ_p, realizado no espaço nulo de $J_p^T(q)$:

$$N_p^T(q) = \left[I - \bar{J}_p(q) J_p(q)\right]^T = \left[I - J_p^T \bar{J}_p^T(q)\right] \tag{4.13}$$

onde, $N_p^T(q)$ é o projetor do espaço nulo de x_p, I a matriz identidade e $\bar{J}_p(q)$ é o inverso dinamicamente generalizado de $J_p(q)$ dado por:

$$\bar{J}_p(q) = M^{-1}(q) J_p^T(q) \Lambda_p(q) \tag{4.14}$$

4.1.3 Controlo de impedância

Existem várias arquiteturas de controlo de interação. Como já foi referido anteriormente, a escolhida para este projeto foi o controlo de impedância. Segundo Christian Ott em *"Cartesian Impedance Control of Redundante and Flexible-Joint Robots"* [23] a ideia de controlo de impedância é modelar a dinâmica de interação do robô com o ambiente, como

sendo um sistema de amortecimento da massa (*mass-spring-damper system*). Nas figuras 4.1 e 4.2 [24] é possível observar uma representação da interação do robô com o ambiente utilizando este tipo de sistema.

Neste modelo, a velocidade $\dot{X}$ do *end-effector* e a força F_e aplicada pelo robô estão relacionadas a uma impedância mecânica Z. No domínio de *Laplace*,

$$Z(s) = \frac{F_e(s)}{\dot{X}(s)} \Leftrightarrow sZ(s) = \frac{F_e(s)}{X(s)} \tag{4.15}$$

com,

$$sZ(s) = As^2 + Ds + K \tag{4.16}$$

onde, A, K e D são os parâmetros do sistema de amortecimento da massa (*mass-spring-damper system*).

Se a analise for feita no domínio do tempo para um ponto de equilíbrio, x_p, a força externa, F_e, é dada por:

$$F_e = A(\ddot{x}_p - \ddot{x}_d) + D(\dot{x}_p - \dot{x}_d) + K(x_p - x_d) \tag{4.17}$$

No espaço de tarefa, a força F_p (ver equação 4.9) é na realidade a soma da força comandada, F_c, com a força do ambiente, F_e, obtidas por forças externas aplicadas ao corpo do robô:

$$F_p = F_c + F_e \tag{4.18}$$

Ao fazer-se,

$$F_c = -\Lambda_p(q)\dot{J}_p(q)\dot{q} + F_p^* \tag{4.19}$$

através da equação 4.6, fica-se com:

$$\Lambda_p(q)\ddot{x}_p = F_p^* + F_e \tag{4.20}$$

onde, F_p^* é a nova variável de controlo. Atribuindo a F_p^* a seguinte lei de controlo:

$$F_p^* = \Lambda_p(q)\ddot{x}_p - [A(\ddot{x}_p - \ddot{x}_d) + D(\dot{x}_p - \dot{x}_d) + K(x_p - x_d)] \tag{4.21}$$

Na prática, a modelagem por inércia (*inertia shaping*) é difícil de implementar uma vez que $\ddot{x}_p$ é muito ruidoso, pelo que o termo da massa, A, é considerado:

$$A = \Lambda_p(q) = \left(J_p(q) M^{-1}(q) J_p^T(q)\right)^{-1} \tag{4.22}$$

Por consequente, a equação 4.21 pode ser rescrita da seguinte maneira:

$$F_p^* = \Lambda_p(q) \ddot{x}_p + D(\dot{x}_d - \dot{x}_p) + K(x_d - x_p) \tag{4.23}$$

A partir da equação 4.23, a equação 4.10 é agora dada por:

$$\tau_p = J_p^T(q) F_p = J_p^T\left(-\Lambda_p(q) \dot{J}_p(q) \dot{q} + F_p^*\right) \tag{4.24}$$

Uma vez que a derivada do jacobiano, $\dot{J}_p(q)$, tem uma influência muito pequena sobre o sistema e será ignorada ao longo deste projeto, a equação 4.24 pode ser simplificada da seguinte maneira:

$$\tau_p = J_p^T F_p^* \tag{4.25}$$

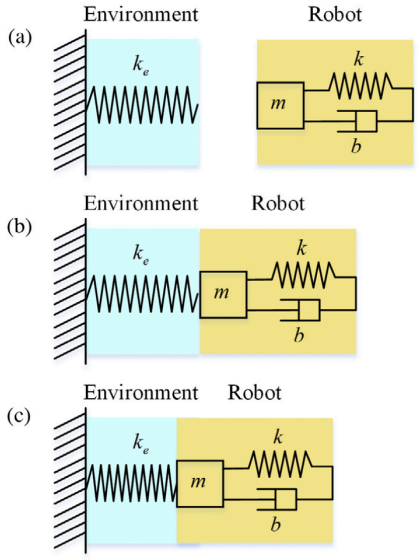


Figura 4.1: Modelo de sistema do robô e ambiente: (a) Sem qualquer contacto entre o robô e o ambiente; (b) Ponto crítico quando o contacto ocorre, mas f = 0; (c) Contacto com $f \neq 0$.

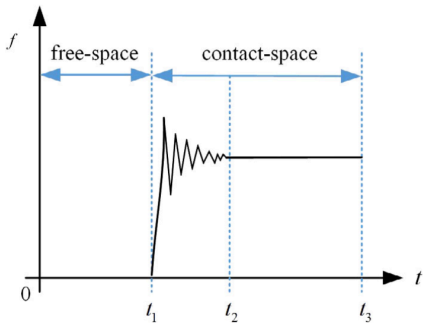


Figura 4.2: Diagrama de forças de contacto entre o robô e o ambiente.

4.1.4 Otimização da postura

É necessário garantir que o robô executa a tarefa que lhe foi atribuída com a máxima eficácia e eficiência. Para conseguir uma eficiência aceitável é importante otimizar o desempenho do robô. A postura do manipulador exerce um papel importante na otimização desse desempenho.

Recorrendo novamente ao artigo *"Computed-Torque Control for Robotic-Assisted Tele-Echography Based on Perceived Stiffness Estimation"*[22], fica-se a saber que quando as juntas ficam perto dos seus limites, podem surgir alguns problemas de singularidade. Uma maneira de evitar esses problemas, é criar uma tarefa secundária que mantêm as juntas redundantes do robô perto do meio do seu valor de alcance através da minimização do gradiente. Isso pode ser conseguido através da projeção de τ_s no espaço nulo da tarefa principal (ver equação 4.10):

$$\tau_s = M\left(q\right) K_N \frac{\partial \upsilon\left(q\right)}{\partial q} \tag{4.26}$$

onde, K_N é a matriz do ganho diagonal e $\upsilon\left(q\right)$ é uma função objetiva dada por:

$$\upsilon\left(q\right) = -\frac{1}{2}\sum_{i=1}^{n}\left(\frac{q_i - \bar{q}_i}{q_{i_M} - q_{i_m}}\right)^2 \tag{4.27}$$

onde, $\bar{q}_i$, q_{i_M} e q_{i_m} são respetivamente o i-ésimo valor médio, máximo e mínimo do intervalo de valores da junta.

4.2 Controlo de impedância cartesiana com otimização da postura

[2]Este subcapítulo descreve o controlador de impedância cartesiana que é o controlador utilizado para controlo de posição e orientação neste projeto.

A força F_p^*, *task force*, é o encadeamento de um vetor da força 3D, f, com um vetor do torque 3D, μ.

$$F_p^* = \begin{bmatrix} f \\ \mu \end{bmatrix} \tag{4.28}$$

[2]Todo o conteúdo teórico veiculado neste subcapítulo está de acordo com: H. Ochoa and R. Cortesao. *"Control Architecture for Robotic-Assisted Polishing Tasks Based on Human Skills"* em IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, vol. 2019-Octob, IEEE, Oct. 2019, pp. 630–637 [21].

4.2.1 Controlo de Posição

De acordo com Cortesão e Ochoa [21], se p_c e p_d forem respetivamente a posição atual e posição desejada do robô, considerando $\ddot{p}_d$ e $\dot{p}_d$ vetores nulos, o vetor da força 3D, f, é calculado da seguinte maneira:

$$f = -D_p \dot{p}_c + K_p \Delta p_{cd} + I_p i_{e_p} \tag{4.29}$$

onde, Δp_{cd} é o erro da posição (ver equação 4.30) e i_{e_p} representa o erro integral da posição (ver equação 4.31). D_p, K_p e I_p são respetivamente a matriz do ganho diagonal referente ao amortecimento da posição, a rigidez e o ganho integral.

$$\Delta p_{cd}\,[k] = p_d\,[k] - p_c\,[k] \tag{4.30}$$

$$i_{e_p}\,[k] = i_{e_p}\,[k-1] + \Delta p_{cd}\,[k] \tag{4.31}$$

onde, k é o número de interações num ciclo de controlo em tempo real com frequência $1kHz$.

4.2.2 Controlo de Orientação

Tal como no controlador de posição, no controlador de orientação é necessário ter em conta o erro. Calcular o erro no controlador de orientação é um pouco mais complexo do que o da posição.

O cálculo do erro de rotação da corrente para a orientação desejada descrita no quadro base, $R_{c\to d}$ é dada por:

$$R_{c\to d} = R_d R_c^{-1} \tag{4.32}$$

onde, R_c é a orientação atual do *end-effector* e R_d a orientação desejada. $R_{c\to d}$ pode ser escrito de diferentes formas: ângulos de *Euler*, representação eixo-ângulo ou quaterniões.

Os quaterniões são uma das técnicas mais populares, no entanto, têm termos fragmentados que podem comprometer questões numéricas. Ao recorrer à representação eixo-ângulo, obtemos resultados semelhantes ao dos quaterniões sem termos fragmentados. Dada a matriz de rotação, R:

$$R = \begin{bmatrix} R_{11} & R_{21} & R_{31} \\ R_{12} & R_{22} & R_{32} \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \tag{4.33}$$

a representação eixo-ângulo associada é:

$$\begin{cases} \theta = \cos^{-1}\left(\frac{R_{11}+R_{22}+R_{33}-1}{2}\right) \\ v = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix} \end{cases} \tag{4.34}$$

onde, θ é o ângulo e v é um vetor unitário. A partir de v, pode ser obtido um vetor de rotação r que não tenha problemas de singularidades ou de cálculos fracionais,

$$r = v\sin\theta = \frac{1}{2}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix} \tag{4.35}$$

A transformação da matriz de rotação num vetor de equação (ver equação 4.35) é representado pela função $R2r\left(\cdot\right)$.

$$r = R2r\left(R\right) \tag{4.36}$$

Ao aplicar $R2r\left(\cdot\right)$ a $R_{c\to d}$, obtemos o vetor rotação $r_{c\to d}$ que representa o erro da orientação Δo_{cd} nas coordenadas da base.

$$\Delta o_{cd} \equiv r_{c\to d} = R2r\left(R_{c\to d}\right) \tag{4.37}$$

Portanto, ao considerar os vetores nulos, a velocidade angular desejada, w_d, e a aceleração desejada, α_d, o vetor do torque 3D, μ, é calculado da seguinte maneira:

$$\mu = -D_o w_c + K_o \Delta o_{cd} + I_o i_{e_o} \tag{4.38}$$

onde, D_o, K_o e I_o são respetivamente a matriz do ganho diagonal referente ao amortecimento da orientação, a rigidez e o ganho integral. i_{e_o} representa o erro integral da orientação. É calculado através da projeção iterativa e da soma do vetor do erro anterior, $\Delta o_{cd}\left[k-1\right]$, ao vetor do erro atual, $\Delta o_{cd}\left[k\right]$. O cálculo depende do vetor do erro ao quadrado.

Para $\delta << 1$,

- Se $\left(\Delta o_{cd}\left[k\right]\right)^T \Delta o_{cd}\left[k\right] < \delta$:

$$i_{e_o}\left[k\right] = \Delta o_{cd}\left[k\right] \tag{4.39}$$

- Caso contrário:

$$i_{e_o}[k] = \left[\left(\left(i_{e_o}[k-1]\right)^T \frac{\Delta o_{cd}[k]}{\|\Delta o_{cd}[k]\|}\right) \frac{\Delta o_{cd}[k]}{\|\Delta o_{cd}[k]\|}\right] + \Delta o_{cd}[k] \tag{4.40}$$

As velocidades angulares e lineares atuais $(\dot{p}_c, w_c)$ são obtidas através da seguinte equação:

$$\begin{bmatrix} \dot{p}_c \\ w_c \end{bmatrix} = J_p(q)\,\dot{q} \tag{4.41}$$

4.2.3 Arquitetura de Controlo

O torque final, τ_c, pode ser simplificado da seguinte maneira:

$$\tau_c = J_p^T F_p^* + N_p^T(q)\,\tau_s + C(q,\dot{q}) + g(q) - \tau_m \tag{4.42}$$

Normalmente, a matriz de amortecimento, D, a matriz de rigidez, K, e a matriz do ganho integral, I, são representadas na estrutura base do robô. Para representar essas matrizes numa qualquer estrutura r pretendida, é necessário fazer algumas transformações adicionais. Essas transformações são:

$$\begin{cases} D = R_d D^r R_d^T \\ K = R_d K^r R_d^T \\ I = R_d I r R_d^T \end{cases} \tag{4.43}$$

Com estas restrições é fácil criar restrições de linha e plano com qualquer posição e orientação.

Na figura 4.3 é possível observar o diagrama de controlador de impedância cartesiano com otimização de postura.

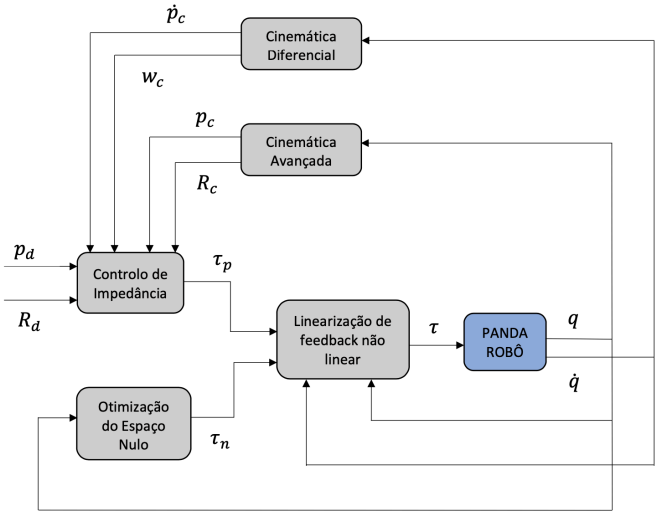


Figura 4.3: Arquitetura de controlo para realizar impressão direta *in situ*. Um controlador de impedância cartesiano com otimização de postura, onde o posicionamento cartesiano é a tarefa principal e a otimização da postura é realizada no espaço nulo.

5

Planeamento de Trajetórias

O planeamento de trajetórias engloba o conjunto de estudos e métodos que permitem definir os regimes de velocidade dos diversos atuadores de forma a fazer o manipulador cumprir os objetivos de movimentação ou de deslocamento planeado.

A tarefa que um braço robótico deve realizar pode ser definida por uma sucessão de pontos onde o braço deve operar. Muitas das vezes, é necessário especificar o trajeto entre dois pontos de forma a evitar obstáculos. Os sistemas robóticos devem executar movimentos suaves de forma a evitar modificações bruscas de posição, velocidade e aceleração.

O principal objetivo deste projeto é obter uma trajetória bem definida através de uma qualquer nuvem de pontos fornecida e assim realizar a impressão direta *in situ* de uma forma simples, rápida, eficaz e controlada. Para tal, é necessário planear muito bem as trajetórias a serem executadas bem como o caminho a ser percorrido. Esse planeamento pode ser feito ou no espaço das juntas ou no espaço de tarefa.

Neste capítulo aborda-se primeiro o planeamento de trajetória no espaço das juntas e posteriormente o planeamento de trajetória no espaço de tarefa. De seguida, fala-se da diferença entre ambos e qual a abordagem que irá ser seguida durante o desenvolvimento do projeto. Por último, aborda-se a estratégia implementada no planeamento do caminho a ser percorrido.

5.1 Planeamento de trajetória no espaço das juntas

O planeamento de trajetórias no espaço das juntas é normalmente utilizado para gerar movimentos suaves e contínuos. Existem vários métodos que nos permitem garantir essa suavidade e continuidade, contudo o mais citado em projetos científicos e utilizado é o de Craig que sugere em *"Introduction to Robotics: Mechanics and Control"* [25] a utilização de dois polinómios de 3º grau entre pontos.

5.1.1 Dois polinómios de 3ª ordem

Em *"Simplified Robotics Joint-Space Trajectory Generation with a via Point Using a Single Polynomial"* [26] recorre-se ao método de Craig de forma a obter um movimento

suave com posição e velocidade contínuas no tempo com uma velocidade inicial e final igual a zero.

Os dois polinómios de 3ª ordem podem ser descritos da seguinte maneira:

$$\begin{cases} \theta_1(t) = a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \\ \theta_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20} \\ \dot{\theta}_1(t) = 3a_{13}t^2 + 2a_{12}t + a_{11} \\ \dot{\theta}_2(t) = 3a_{23}t^2 + 2a_{22}t + a_{21} \\ \ddot{\theta}_1(t) = 6a_{13}t + 2a_{12} \\ \ddot{\theta}_2(t) = 6a_{23}t + 2a_{22} \end{cases} \tag{5.1}$$

onde, θ, $\dot{\theta}$ e $\ddot{\theta}$ representam a posição, velocidade e aceleração respetivamente.

Em conjunto, os dois polinómios requerem oito restrições. As primeiras quatros provêm dos pontos inicias e finais.

$$\begin{cases} \theta_1(0) = \theta_o \\ \dot{\theta}_1(0) = 0 \end{cases} \qquad \begin{cases} \theta_2(t_2) = \theta_f \\ \dot{\theta}_2(t_2) = 0 \end{cases} \tag{5.2}$$

onde, θ_o corresponde à posição inicial e θ_f à posição final.

As restantes restrições derivam do facto de se forçar o primeiro polinómio a terminar na posição θ_V onde o segundo começa. É importante garantir que as velocidades e acelerações dos dois polinómios coincidam, não sendo necessariamente igual a zero.

$$\begin{cases} \theta_1(t_1) = \theta_V \\ \theta_2(0) = \theta_V \end{cases} \qquad \begin{cases} \dot{\theta}_1(t_1) = \dot{\theta}_2(0) \\ \ddot{\theta}_1(t_1) = \ddot{\theta}_2(0) \end{cases} \tag{5.3}$$

As restrições originam oito equações lineares nas oito incógnitas. Três dos coeficientes polinomiais desconhecidos são fáceis de ser encontrados a partir das restrições inicias e restrições temporais:

$$\begin{cases} a_{10} = \theta_o \\ a_{11} = 0 \\ a_{20} = \theta_V \end{cases} \tag{5.4}$$

Assim, consegue-se simplificar a equação que nos permite descobrir as restantes incóg-

nitas:

$$\begin{bmatrix} 1 & t_1 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_2 & t_2^2 \\ 0 & 0 & 1 & 2t_2 & 3t_2^2 \\ 2t_1 & 3t_1^2 & -1 & 0 & 0 \\ 2 & 6t_1 & 0 & -2 & 0 \end{bmatrix} \begin{Bmatrix} a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{Bmatrix} = \begin{Bmatrix} \frac{(\theta_V - \theta_o)}{t_1^2} \\ \frac{\theta_f - \theta_V}{t_2} \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5.5}$$

Ao considerar, $t_1 = t_2 = \frac{t_f}{2} = t_V = T$, obtêm-se:

$$\begin{cases} a_{12} = \frac{-3\left(\theta_f - \theta_V\right) + 9(\theta_V - \theta_o)}{4T^2} \\ a_{13} = \frac{3\left(\theta_f - \theta_V\right) - 5(\theta_V - \theta_o)}{4T^3} \\ a_{21} = \frac{3\left(\theta_f - \theta_o\right)}{4T} \\ a_{22} = \frac{3\left(\theta_f - \theta_V\right) - 3(\theta_V - \theta_o)}{2T^2} \\ a_{23} = \frac{-5\left(\theta_f - \theta_V\right) + 3(\theta_V - \theta_o)}{4T^3} \end{cases} \tag{5.6}$$

5.2 Planeamento de trajetória no espaço de tarefa

Recorre-se ao planeamento de trajetória no espaço de tarefa quando é necessário garantir que o movimento do *end-effector* siga um caminho geometricamente especificado no espaço de trabalho. Esse planeamento pode ser feito através da interpolação de uma sequência de pontos ou através da geração de um movimento analítico.

Em ambos os casos, a sequência temporal dos valores obtidos pelas variáveis do espaço de tarefa é utilizada para obter a sequência correspondente dos valores das variáveis do espaço das juntas, essa correspondência é feita através de um algoritmo de cinemática inversa.

Nos casos em que não existe a necessidade de percorrer todos os pontos do caminho, o planeamento da trajetória pode ser feito através da atribuição de N pontos, mencionando os valores escolhidos para descrever a pose do *end-effector*, x_e, no espaço de tarefa num nomeado instante t_k, com $k = 1, \cdots, N$. À semelhança do que acontece no planeamento de trajetória no espaço das juntas, o caminho a ser percorrido é gerado através de uma função interpoladora vetorial suave entre pontos.

Essa função pode ser calculada aplicando a cada componente x_e qualquer uma das seguintes técnicas de interpolação:

- Interpolação de polinómios com velocidades estabelecidas em pontos de passagem (*via points*);
- Interpolação de polinómios com acelerações continuas entre pontos (*spline*);

- Interpolação de polinómios lineares com misturas parabólicas (*parabolic blends*).

Todas estas técnicas de interpolação podem ser encontras no livro *"Robotics: Modeling, Planning and Control"* [19]

5.3 Planeamento de trajetória no espaço das juntas Vs. Planeamento de trajetória no espaço de tarefa

Em tarefas como a soldadora, a impressão ou o corte é necessário que o caminho especificado seja corretamente seguido. Nestes casos, o planeamento de trajetória no espaço de tarefa torna-se a melhor opção em relação ao planeamento de trajetória no espaço das juntas, em que é necessário definir a trajetória por interpolação de pontos. Para além disso, os movimentos arquitetados no espaço de tarefa, são mais previsíveis, isto é, é mais fácil prever o posicionamento do *end-effector* e saber se ele irá colidir com algum obstáculo ou não.

No entanto, o planeamento de trajetória no espaço das juntas não deve ser menosprezado. Segundo Sahar e Hollerbach em *"Planning of Minimum-Time Trajectories for Robot Arms"* [27] o movimento é mais rápido quando é realizado por segmentos de reta que unem dois pontos definidos no espaço das juntas. Em ambientes de trabalho onde não existe risco de colisão é desejável que o movimento ocorra o mais rápidamente possível, tornando-se assim o planeamento da trajetória no espaço das juntas o mais indicado. Além disso, para movimentos planeados no espaço das juntas os pontos definidos são alcançados melhor precisão: isso deve-se ao facto dos erros gerados pelos modelos cinemáticos, direto e inverso, não serem tidos em conta no sistema. Em *"Minimum Ratio-Locked Profile Times for Robot Trajectories"* [28], Ficher e Mujtaba afirmam que ao descrever uma trajetória linear no espaço das juntas permite gerar movimentos suaves e contínuos do *end-effector*, cuja trajetória não varia com a velocidade do manipulador.

Sintetizando, o planeamento de trajetória no espaço das juntas é ideal quando existe a necessidade de realizar movimentos mais grosseiros ou quando não existe a necessidade de transportar objetos. Contudo, se na área em que o manipulador estiver a operar existir algum risco de colisão ou houver alguma necessidade de transporte de objetos o planeamento de trajetória no espaço de tarefa é o mais indicado uma vez que, consegue garantir que nenhum obstáculo é tocado, e/ou que o objeto transportado não seja danificado.

É importante referir que o planeamento de trajetória no espaço de tarefa possui uma dificuldade adicional que consiste no planeamento de orientação do *end-effector* ao longo da trajetória.

Sendo o objetivo principal deste projeto gerar uma trajetória através de uma nuvem de pontos de modo a ser possível a realização da impressão direta *in situ*, o planeamento de trajetória no espaço de tarefa torna-se o método mais adequado ao processo.

5.4 Planeamento do caminho a ser percorrido para a realização da impressão direta *in situ*

Como foi referido no subcapítulo anterior, o planeamento de trajetória no espaço de tarefa é a abordagem mais indicada para trabalhos que necessitem de percorrer um caminho bastante específico.

Após ser tomada a decisão sobre a forma de como se vai planear a trajetória, é necessário decidir o caminho que o *end-effector* do braço manipulador PANDA irá percorrer, por outras palavras, definir os pontos que irão fazer parte da trajetória. Os pontos que pertencem a uma trajetória podem ser obtidos de várias maneiras, sendo a mais convencional e utilizada a retirada de pontos através de uma nuvem de pontos.

Uma nuvem de pontos, consiste num formato de dados 3D que corresponde a uma coleção de pontos no espaço tridimensional. São normalmente usadas para a deteção de objetos, para produzir modelos CAD (*Computer Aided Design*), entre outros. Cada ponto tem três coordenadas dimensionais em relação ao gerador da nuvem. Os pontos são habitualmente utilizados para criar malhas de superfície (*meshes*) e estimar distâncias e volumes. Neste projeto, uma das nuvens de pontos utilizada é uma nuvem de pontos alusiva à parte inferior de um braço com a mão cortada. Na figura 5.1 é possível observar essa nuvem.

Figura 5.1: Nuvem de pontos de um CAD que representa a parte inferior de um braço com a mão cortada.

A nuvem de pontos ilustrada na figura 5.1 foi obtida através de um *scanner Artec Leo*[1]. Ao digitalizar o braço foi concebido um ficheiro com extensão *.ply* (*Polygon File Format*).

Os ficheiros *.ply* conseguem armazenar objetos como uma lista de polígonos planos que po-

[1]O *scanner Artec Leo* é um *scanner* 3D profissional capaz de fornecer fluxo de trabalho mais intuitivo, tornando a digitalização em 3D fácil e intuitiva. Mais informações sobre o *scanner Artec Leo* podem ser consultadas no site *Artec3D* [29].

dem ter as seguintes propriedades atribuídas: cor e transparência, descrição da superfícies e coordenadas espaciais.

Como se pode observar pela figura 5.1, a nuvem de pontos obtida é muito densa, o que dificulta o processo de filtração da informação necessária para construir um caminho. Portanto, houve a necessidade de reduzir a densidade de pontos na nuvem de modo a obter unicamente a informação suficiente para traçar um caminho. Assim sendo, recorreu-se ao *software Fusion 360*[2]. Este *software* permitiu reduzir o número de malhas de superfície (*meshes*) tornando, desse modo, a nuvem de pontos muito menos densa. Na figura 5.2 é possível observar as etapas de redução até ter sido obtido uma nuvem de pontos aceitável e que contêm unicamente informação necessária e suficiente à realização do trajeto.

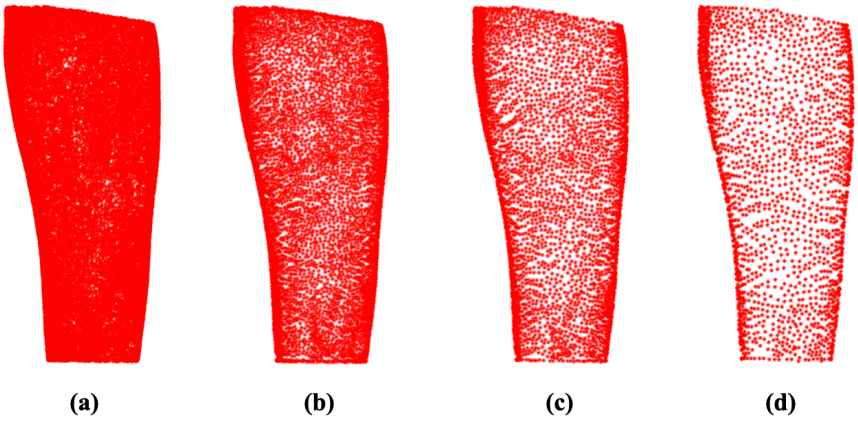


Figura 5.2: Etapas de redução da nuvem de pontos: (a) Nuvem de pontos original; (b) Nuvem de pontos reduzida a 38% dos pontos; (c) Nuvem de pontos reduzida a 20% dos pontos; (d) Nuvem de pontos reduzida a 9.98% dos pontos.

A nuvem de pontos obtida (cf. figura 5.2 (d)) em comparação com a nuvem de pontos original (cf. figura 5.2 (a)) é muito menos densa o que facilita a seleção de pontos que o manipulador deve percorrer. Porém, se a nuvem de pontos for analisada em detalhe (cf. figura 5.3) é possível observar que os pontos que a constituem são desalinhados e desorganizados, dificultando, desse modo, a tarefa de traçar um caminho que seja facilmente executado. Para ultrapassar esse impasse, decidiu-se recorrer a estratégia em que fosse possível obter os pontos da nuvem alinhados e organizados.

[2]O *Fusion 360* é uma plataforma de *software* CAD, CAM, PCB e de modelação 3D baseada na nuvem para o design e fabrico de produtos. Mais informações sobre o *Fusion 360* podem ser consultadas no site da *Autodesk* [30].

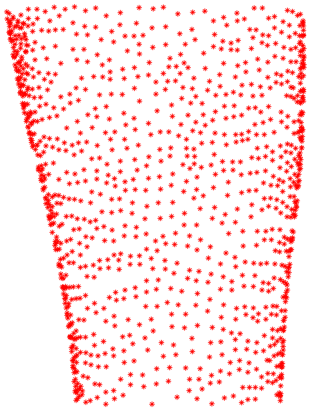


Figura 5.3: Nuvem de pontos reduzida a 9.98% dos pontos ampliada ao detalhe.

Através do *software Matlab*[3] foi possível desenvolver um algoritmo que permite obter os pontos da nuvem, alinhá-los e dispô-los como desejado. O algoritmo desenvolvido tem como ideia principal, projetar os pontos constituintes de uma trajetória sobre a superfície da malha (*mesh*) do braço.

O primeiro passo consistiu na seleção de um ponto da nuvem onde se pretende começar a executar a trajetória. Após ser obtido o ponto inicial, projetou-se uma linha tridimensional com início nesse ponto. A linha tem o mesmo comprimento da área em que se pretende fazer a impressão. Após a linha ser traçada, é dividida em N pontos de acordo com o tempo em que se deseja realizar a trajetória (quanto mais pontos a linha for dividida, mais tempo demorará a ser executada a trajetória).

Posto isto, é necessário projetar os pontos da trajetória na malha de superfície (*mesh*) do braço. Isso é feito através do cálculo da distância euclidiana entre pontos. Para cada ponto da trajetória existe uma distância entre esse ponto e os pontos da nuvem de pontos. O ponto da nuvem que estiver a menor distância do ponto a ser analisado nesse momento, irá ceder ao ponto analisado a sua coordenada z e vetor normal, fazendo com que o ponto seja projetado em cima da malha de superfície (*mesh*). Durante o decorrer do algoritmo, a linha tridimensional é projetada na sua totalidade na malha e, consequentemente, o caminho para realizar a trajetória é encontrado.

Na figura 5.4 é possível analisar todos os passos do algoritmo para a realização de um movimento horizontal.

[3]O *Matlab* é uma plataforma de programação e computação numérica utilizada por milhões de engenheiros e cientistas para analisar dados, desenvolver algoritmos e criar modelos. Mais informações sobre a plataforma *Matlab* podem ser encontradas no site *MathWorks* [31].

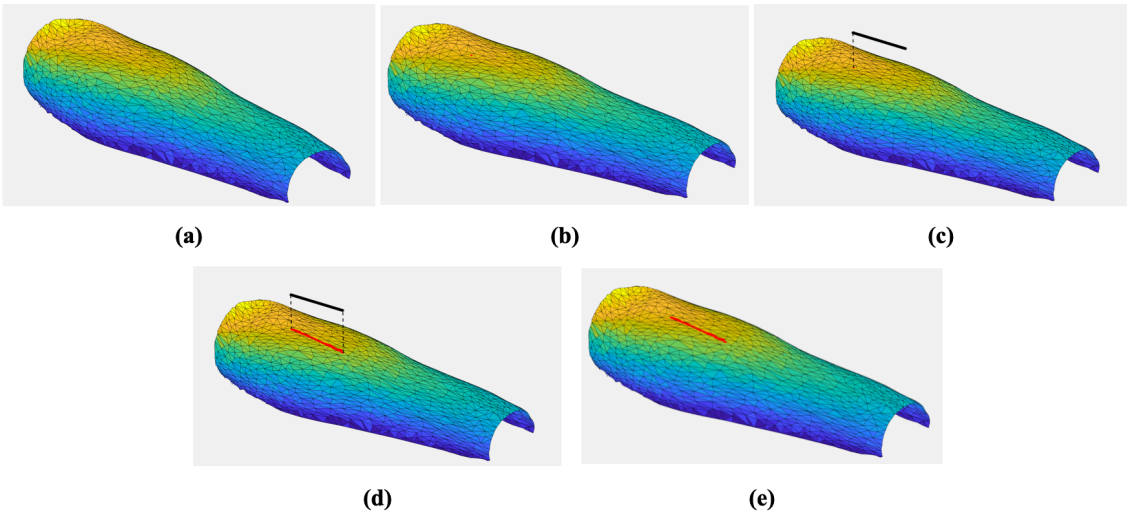


Figura 5.4: Etapas do algoritmo de projeção de linha para a realização de um movimento horizontal: (a) Malha de superfície do braço; (b) Seleção do ponto da nuvem; (c) Linha traçada com origem no ponto de seleção; (d) Projeção dos pontos da linha na malha de superfície; (e) Caminho final.

No entanto, na maioria dos casos, o que se pretende é arranjar um caminho no qual seja possível percorrer uma área na sua plenitude e não uma única trajetória. Nesse sentido, em vez de ser projetada uma única trajetória na nuvem de pontos, são projetadas várias linhas que definem a trajetória até ser cobreta a totalidade da área em questão. A figura 5.5 mostra uma área de 10 centímetros de comprimento constituída por 35 linhas tridimensionais que definem a trajetória com espaçamento de 1 milímetro entre si, o que equivale a ter uma área de 10×3.5 cm.

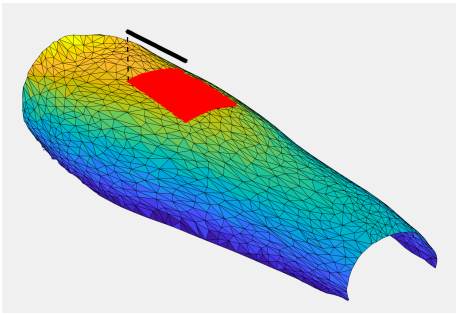


Figura 5.5: Projeção de uma área com 10×3.5 cm constituída por 35 linhas com espaçamento de 1 milímetro entre elas.

6

Resultados Experimentais

Este capítulo é dedicado à arquitetura de controlo implementada neste projeto, à trajetoria e ao caminho planeado. Primeiro foram testados no simulador Gazebo o esquema de controlo e o planeamento da trajetória, sendo depois feitos testes no verdadeiro robô PANDA.

Os ensaios realizados tiveram em conta os seguintes aspetos:

- Tamanho da área a ser percorrida;
- Velocidade;
- Movimento a ser realizado: horizontal ou vertical.

Durante o processo de elaboração do projeto foi criado um repositório com o nome *3DPrint*. Este repositório contém todos os ficheiros necessários para a concretização do projeto. Todos os ensaios e simulações apresentadas neste capítulo e no próximo podem ser replicadas através dele. Para informações de como instalar e de como correr o simulador realizando ensaios, deve-se consultar os anexos B e C. Este projeto também contém um repositório desenvolvido em *Matlab*, é nele que são elaborados os caminhos por onde se pretende percorrer o manipulador. Este repositório também pode ser usado para análise e tratamento de resultados.

6.1 Resultados obtidos através do simulador Gazebo

Antes dos resultados serem apresentados é importante saber como funciona o simulador Gazebo.

Quando o robô é lançado no simulador, é inicializado com um controlador de trajetória em cada uma das juntas do robô. Para o efeito, é necessário recorrer ao nó *move_to_star*, que tem como objetivo enviar torques para as juntas. Seguidamente, é inicializado um controlador de comutação de controlo de trajetórias para o controlo do binário. Isto é possível porque o *ros_control* fornece um *controller_manager* que permite interagir com os diferentes controladores. Sem este controlador de comutação, o robô iniciaria caído no chão do simulador uma vez que, os binários em cada junta seriam iguais a zero. Após

o controlador de comutação ter sido iniciado, o circuito de controlo onde os binários são gerados depende da arquitetura de controlo implementada.

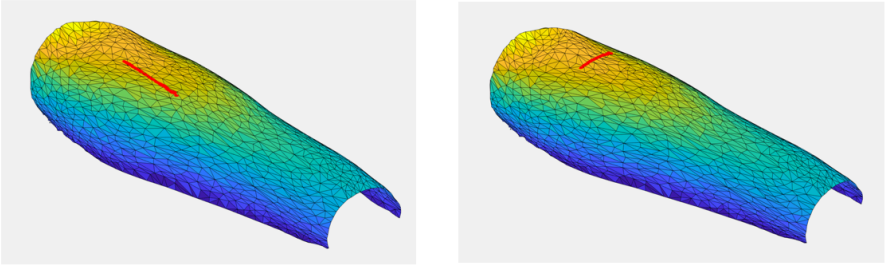
A arquitetura de controlo de impedância permite a interação entre o robô e o ambiente. Este tipo de arquitetura pode ser utilizado para interagir com um humano ou com qualquer outro tipo de material.

Numa primeira fase do projeto, para analisar o desempenho do robô durante o processo de impressão, os testes serão realizados em espaço livre. Mais tarde será demonstrada a tarefa de impressão direta *in situ* sobre um braço de um manequim.

6.1.1 Análise do caminho realizado sobre uma linha tridimensional projetada sobre a superfície de um braço (simulação realizada em espaço livre)

O primeiro ensaio tem como objetivo analisar o comportamento do robô perante diferentes trajetórias retilíneas segundo uma linha tridimensional. Para tal, recorrendo ao algoritmo e à nuvem de pontos apresentados no subcapítulo 5.4 obtiveram-se dois caminhos distintos: um primeiro caminho a ser realizado na vertical e um segundo caminho na horizontal.

Na figura 6.1 é possível observar esses dois caminhos.



(a) **(b)**

Figura 6.1: Projeção dos caminhos na malha da superfície: (a) Caminho a ser realizado segundo uma linha vertical; (b) Caminho a ser realizado segundo uma linha horizontal.

Antes de começar a percorrer a inha tridimensional, foi necessário deslocar o *end-effector* do robô para o ponto inicial da mesma. Para tal, recorreu-se a um polinómio do terceiro grau que é denotado da seguinte maneira:

$$q_d(t) = q_i + \frac{3(q_f - q_i)(t - t_i)^2}{(t_f - t_i)^2} - \frac{2(q_f - q_i)(t - t_i)^3}{(t_f - t_i)^3} \tag{6.1}$$

Nas figuras seguintes é possível observar que o robô apresenta um bom comportamento para movimentos diferentes em espaço livre. Ao analisar as parcelas alusivas à orientação é possível denotar que o movimento da orientação em *Roll* é ligeiramente desfasado da

referência. No entanto, o erro causado é praticamente irrelevante não provocando alterações no desempenho do robô. As figuras 6.2 e 6.3 são referentes à realização do caminho vertical e caminho horizontal respetivamente com duração de 4 segundos.

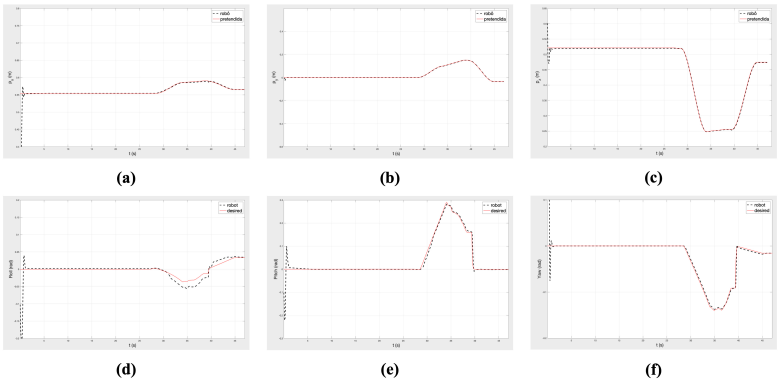


Figura 6.2: Posição e orientação em PRY referentes à realização do movimento vertical com duração de 4 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

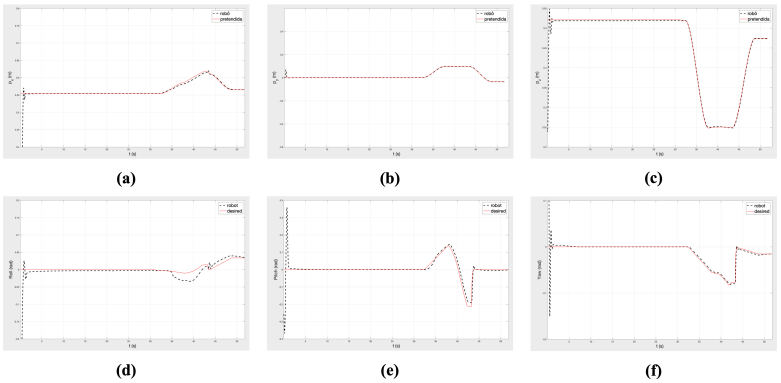


Figura 6.3: Posição e orientação em PRY referentes à realização do movimento horizontal com duração de 4 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

A tabela 6.1 representa os ganhos do controlador utilizados para mover o robô no espaço livre.

Tabela 6.1: Ganhos de controlo por impedância no espaço de tarefa, onde a variável I em (a) e em (b) representam a matriz identidade. (Linha de reta).

(a) Ganhos posição

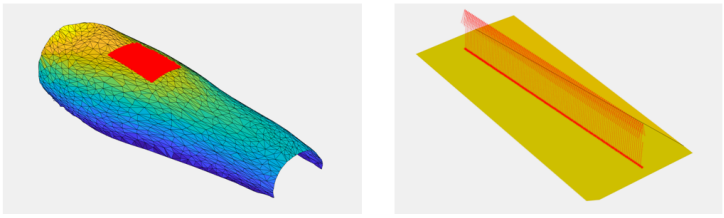
Ganho	Posição
$\boldsymbol{K_p}$	$3000I_{3\times3}$
$\boldsymbol{D_p}$	$100I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$150I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

6.1.2 Análise do caminho realizado sobre uma área de 10 x 3.5 cm projetada sobre a superfície de um braço (simulação realizada em espaço livre)

Neste segundo ensaio, o objetivo principal é fazer com que o *end-effector* do robô PANDA percorra uma área com 10×3.5 *cm*. À semelhança do que foi feito anteriormente, recorreu-se à nuvem de pontos e ao algoritmo do subcapítulo 5.4 para definir a área desejada. É importante relembrar que para cada ponto existe um vetor normal correspondente. É através desses vetores normais que a orientação do robô é gerada. Na figura 6.4 é possível observar a área pretendida assim como os vetores normais correspondentes a cada ponto.

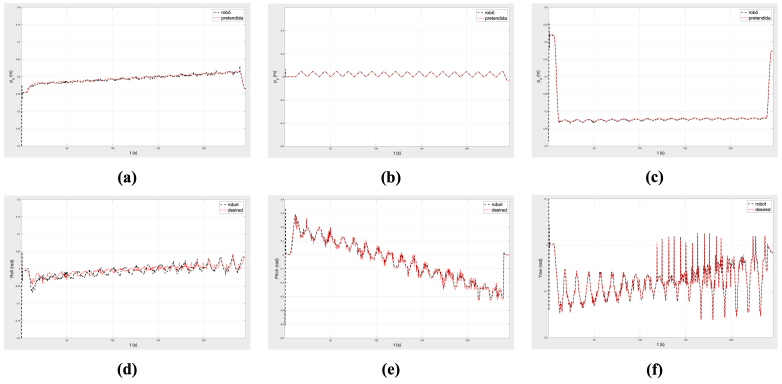


(a) **(b)**

Figura 6.4: Área de 10×3.5 *cm*: (a) Nuvem de pontos que constituem a área; (b) Vetores normais correspondentes a cada ponto.

Semelhantemente ao que se fez anteriormente, recorreu-se à equação 6.1 para deslocar o *end-effector* para o ponto inicial e, assim, começar a percorrer a área desejada.

Na figura 6.5 é possível observar que o robô consegue percorrer a área pertendida com um erro praticamente inexistente. No entanto, ao analisar as parcelas referentes à orientação é visível que, apesar de o robô conseguir realizar as orientações, existem mudanças demasiado violentas. Isto deve-se ao facto de os vetores normais retirados da nuvem de pontos apresentarem algum ruído.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

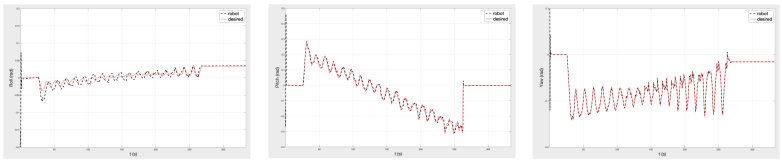
Figura 6.5: Posição e Orientação em PRY referentes à realização do caminho da área de 10×3.5 *cm* com duração de 210 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

De modo a reduzir o ruído verificado foi aplicado aos vetores normais um filtro de banda passa-baixo.

$$\begin{cases} \check{v}_k = v_k \\ \check{v}_{k+1} = \frac{a\check{v}_k + bv_{k+1}}{|a\check{v}_k + bv_{k+1}|} \end{cases} \tag{6.2}$$

Onde $\check{v}_k$ é o valor do vetor normal filtrado, v o vetor normal e a e b representam a amplitude e frequência respetivamente.

Ao comparar as figuras 6.5 (d), 6.5 (e) e 6.5 (f) com a figura 6.6, é evidente a existência de uma melhoria significativa nas orientações elaboradas pelo robô. Com a passagem do filtro, o robô passou a realizar movimentações mais suaves evitando assim movimentos bruscos desnecessários.



(a) (b) (c)

Figura 6.6: Orientação em PRY referentes à realização do caminho da área de 10×3.5 cm com duração de 210 segundos após o uso do filtro de banda passa-baixo com $a = 1000$ e $b = 1$. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

A tabela 6.2 representa os ganhos do controlador de impedância em ambiente de simulação.

Tabela 6.2: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área de 10×3.5 cm).

(a) Ganhos posição

Ganho	Posição
$\boldsymbol{K_p}$	$3000I_{3\times3}$
$\boldsymbol{D_p}$	$100I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

6.1.3 Análise do caminho realizado sobre uma área de 10 x 1 cm projetada sobre a superfície de um braço (simulação realizada em espaço livre)

O último ensaio realizado em simulação tem como intuito avaliar o desempenho do robô perante velocidades distintas numa área 10×1 cm, como se pode observar na figura 6.7.

Como foi referido na subcapítulo 5.4, o tempo que o robô demora a percorrer uma área depende do número de pontos com que a área é definida. Cada linha tridimensional é constituída por N pontos dependendo da velocidade em que se deseja percorrê-la. Isto

deve-se ao facto do robô, quando se apresenta para realizar a trajetória da linha tridimensional, a cada milésimo de segundo ler um ponto e deslocar o seu *end-effector* para a posição desejada. Em termos práticos, uma área constituída por 10 linhas tridimensional com 2000 pontos cada uma, demorará 20 segundos a ser percorrida.

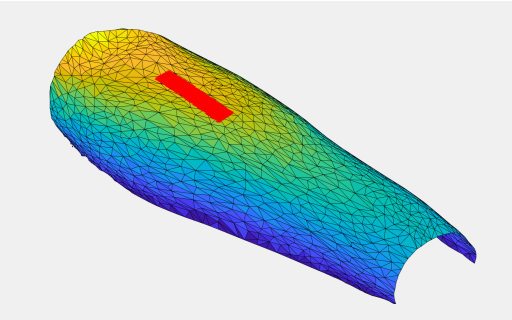


Figura 6.7: Nuvem de pontos que constituem a área 10×1 *cm* projetada na malha de superfície.

Semelhante ao que foi feito na subseção 6.1.2, antes de o robô começar a percorrer a área, os vetores normais foram todos passados pelo filtro de banda passa-baixo.

As figuras 6.8, 6.9 e 6.10 representam os resultados obtidos para diferentes velocidades. Ao analisar as figuras juntamente com a tabela de erros (ver tabela 6.3), constata-se que o robô detém um bom desempenho em velocidades altas como em velocidades baixas. Contudo, é notório que, o robô em velocidades mais baixas apresenta um erro médio menor quando comparado com o observado em velocidades mais altas.

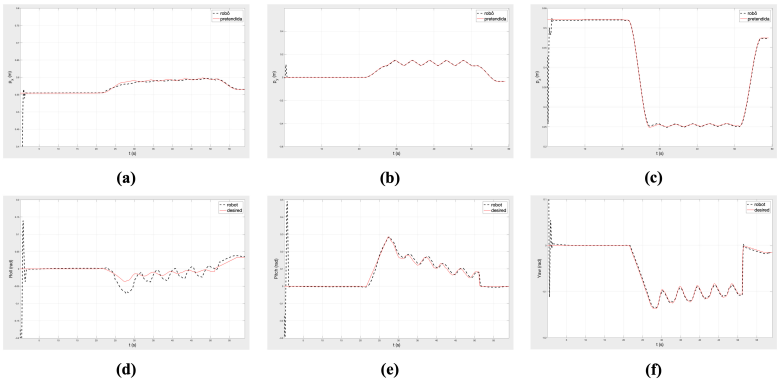


Figura 6.8: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 20 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

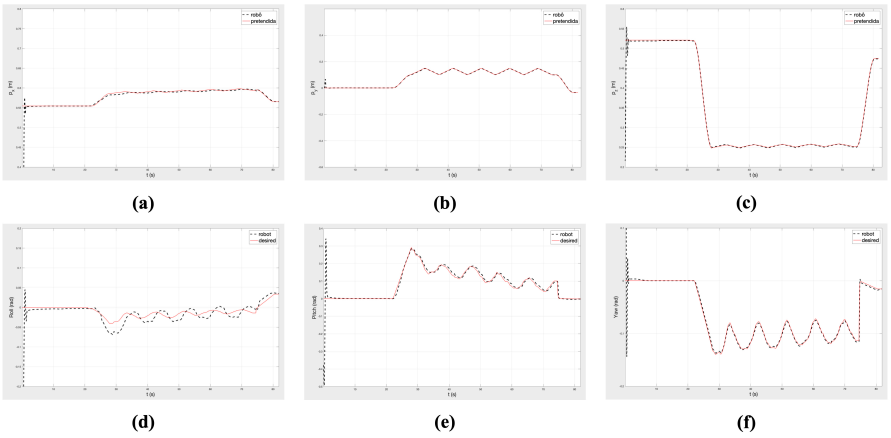


Figura 6.9: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 40 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

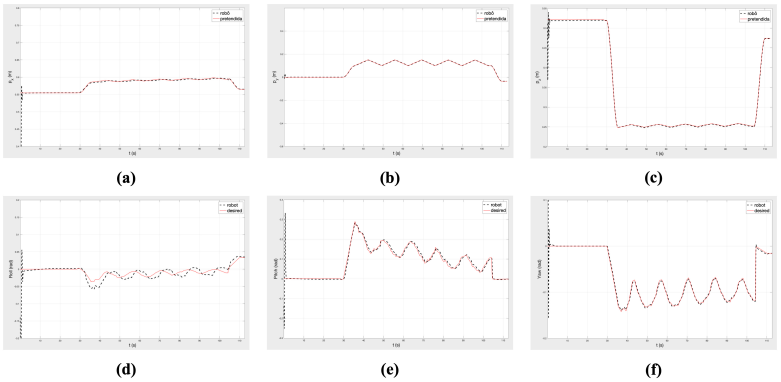


Figura 6.10: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 60 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

Tabela 6.3: Tabela da média dos erros quadrádicos obtido nas diferentes velocidades no simulador Gazebo.

Erro/Tempo (s)	**20*s***	**40*s***	**60*s***
$\boldsymbol{e_x}$ $\boldsymbol{(m)}$	0.003	0.002	0.001
$\boldsymbol{e_y}$ $\boldsymbol{(m)}$	0.001	0.001	0.001
$\boldsymbol{e_z}$ $\boldsymbol{(m)}$	0.002	0.001	0.001
$\boldsymbol{Roll(rad)}$	0.011	0.011	0.009
$\boldsymbol{Pitch(rad)}$	0.012	0.010	0.009
$\boldsymbol{Yaw}$ $\boldsymbol{(rad)}$	0.004	0.004	0.003

A tabela 6.4 representa os ganhos do controlador utilizados para mover o robô no espaço livre.

Tabela 6.4: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área 10×1 cm).

(a) Ganhos posição

Ganho	Posição
$\boldsymbol{K_p}$	$3000I_{3\times3}$
$\boldsymbol{D_p}$	$100I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$150I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

6.2 Resultados obtidos através do robô PANDA

Chegou o momento de avaliar e comparar os resultados experimentais obtidos no robô real com os resultados obtidos no simulador Gazebo. Tal como acontece no simulador, os ensaios experimentais são realizados em espaço livre.

6.2.1 Análise do caminho realizado sobre uma linha tridimensional projetada sobre a superfície de um braço (trajetória realizada em espaço livre)

Os seguintes resultados experimentais apresentados são feitos nas mesmas condições dos resultados simulados. Ou seja, identicamente ao que se foi feito na subseção 6.1.1, irá ser avaliado a performance do robô para movimentos distintos (cf. figura 6.1).

Semelhante aos resultados obtidos na simulação, através das figuras 6.11 e 6.12 podemos afirmar que o robô PANDA aparenta dispor de um bom comportamento para movimentos diferentes em espaço livre. Conforme sucedeu nos resultados em simulação, o movimento da orientação em *Roll* é ligeiramente desfasado da referência. Também a orientação em *Pitch* sofreu um ligeiro desvio. No entanto, continua a não haver interferências no desempenho do robô.

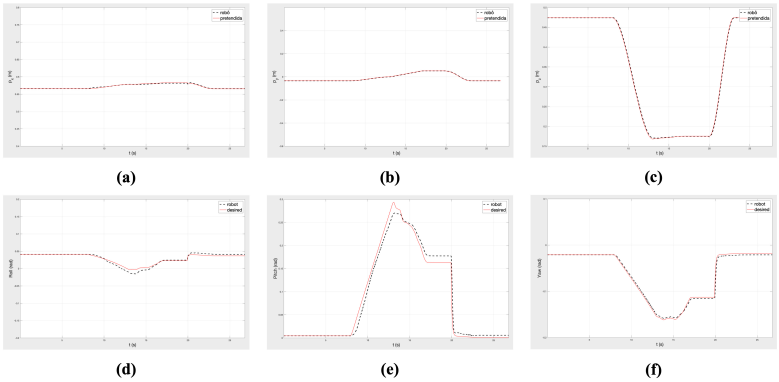


Figura 6.11: Posição e orientação em PRY referentes à realização do movimento vertical com duração de 4 segundos: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

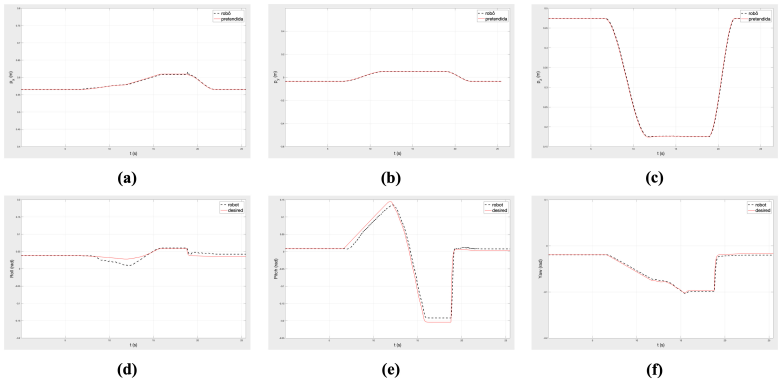


Figura 6.12: Posição e orientação em PRY referentes à realização do movimento horizontal com duração de 4 segundos: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

A tabela 6.5 representa os ganhos do controlador utilizados para mover o robô no espaço livre.

Tabela 6.5: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Linha de reta).

(a) Ganhos posição

Ganho	Posição
$\boldsymbol{K_p}$	$5000I_{3\times3}$
$\boldsymbol{D_p}$	$140I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

6.2.2 Análise do caminho realizado sobre uma área de 10 x 3.5 cm projetada sobre a superfície de um braço (trajetória realizada em espaço livre)

Agora, são apresentados os resultados experimentais referentes ao caminho realizado sobre uma área com 10×3.5 *cm* (cf. figura 6.4). Analogamente à simulação, também neste caso passou-se os vetores normais pelo filtro de banda passa-baixo (ver equação 6.2) de modo a evitar movimentações bruscas do robô.

Como já era esperado, o robô apresentou um bom comportamento durante todo o percurso, tendo como erro médio um valor aceitável, na ordem de 1 milímetro em posição e inferior a 1 grau na orientação (cf. figura 6.13).

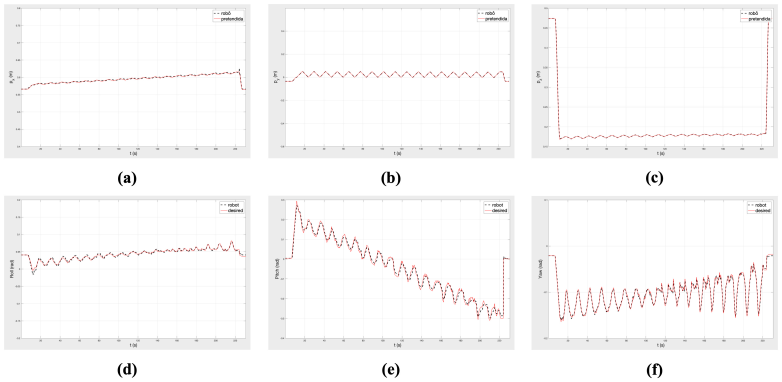


Figura 6.13: Posição e orientação referentes à realização do caminho da área 10×3.5 cm com duração de 210 segundos após o uso do filtro de banda passa-baixo com $a = 1000$ e $b = 1$: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

A tabela 6.6 representa os ganhos do controlador utilizados para mover o robô no espaço livre.

Tabela 6.6: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área 10×3.5 cm).

(a) Ganhos posição

Ganho	Posição
$\boldsymbol{K_p}$	$5000I_{3\times3}$
$\boldsymbol{D_p}$	$140I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

6.2.3 Análise do caminho realizado sobre uma área de 10 x 1 cm projetada sobre a superfície de um braço (trajetória realizada em espaço livre)

Por último, realizou-se o ensaio experimental referente à simulação realizada na subseção 6.1.3. Como tem sucedido até esta fase, os resultados experimentais são obtidos nas mesmas condições que a simulação. Percorreu-se uma área com 10×1 cm (cf. figura 6.7) em diferentes velocidades.

Como já era esperado, o robô PANDA apresenta um bom desempenho para velocidades altas como baixas. Ao observar a tabela 6.7, pode-se afirmar que se o objetivo principal for percorrer uma área com o menor erro possível. Assim melhor solução é percorrer essa área com uma velocidade baixa.

As imagens 6.14, 6.15 e 6.16 representam os resultados obtidos para diferentes velocidades no robô real.

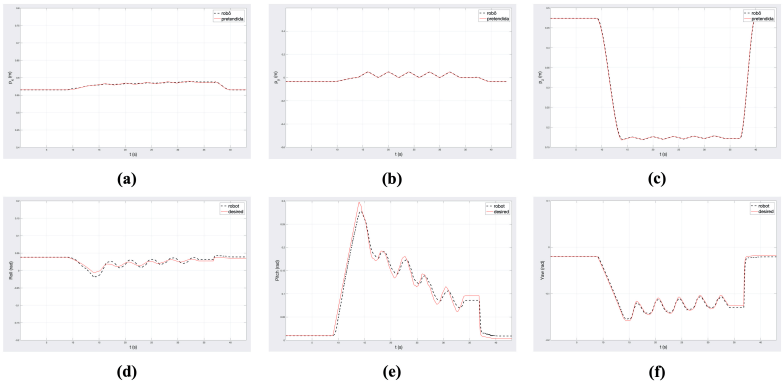


Figura 6.14: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 20 segundos: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

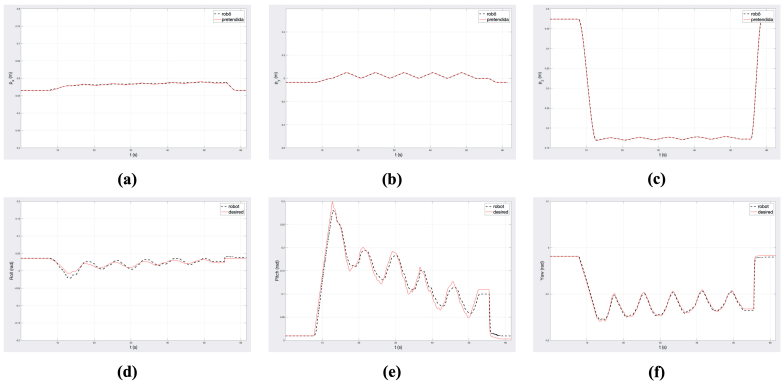


Figura 6.15: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 40 segundos: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

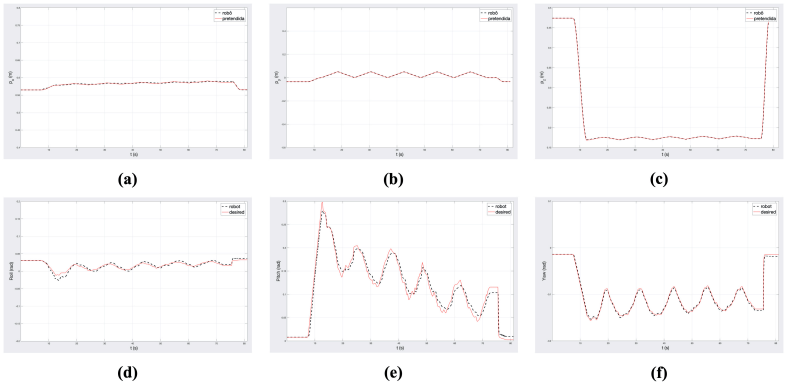


Figura 6.16: Posição e Orientação em PRY referentes à realização do caminho da área de 10×1 *cm* com duração de 60 segundos: (a) p_x; (b) p_y; (c) p_z; (d) *Roll*; (e) *Pitch*; (f) *Yaw*.

Tabela 6.7: Tabela da média dos erros quadrádicos obtido nas diferentes velocidades no robô PANDA.

Erro/Tempo (s)	**20s**	**40s**	**60s**
$\boldsymbol{e_x}$ $(\boldsymbol{m})$	$9.90e^{-04}$	0.001	0.001
$\boldsymbol{e_y}$ $(\boldsymbol{m})$	0.001	$8.54e^{-04}$	$7.60e^{-04}$
$\boldsymbol{e_z}$ $(\boldsymbol{m})$	0.002	0.001	0.001
$\boldsymbol{Roll(rad)}$	0.005	0.005	0.004
$\boldsymbol{Pitch(rad)}$	0.009	0.009	0.009
$\boldsymbol{Yaw(rad)}$	0.004	0.004	0.003

Ao comparar a tabela 6.7 com a tabela 6.3 é notório que existe uma ligeira discrepância entre os erros médios obtidos no simulador e os obtidos no robô real. Essa discrepância deve-se ao facto de os ganhos utilizados no robô real serem superiores aos ganhos utilizados na simulação, permitindo assim uma diminuição do erro médio.

A tabela 6.8 representa os ganhos do controlador de impedância em ambiente de real.

Tabela 6.8: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área 10×1 cm).

(a) Ganhos posição

Ganho	**Posição**
$\boldsymbol{K_p}$	$5000I_{3\times3}$
$\boldsymbol{D_p}$	$140I_{3\times3}$

(b) Ganhos orientação

Ganho	**Posição**
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

No capítulo seguinte, será testada a tarefa de impressão direta *in situ* sobre um braço de um manequim. Primeiro os testes serão realizados em ambiente de simulação e posteriormente serão testados no robô PANDA.

7

Impressão direta *in situ*

Neste capítulo irá ser testada a tarefa de impressão direta *in situ*. Para tal, foi necessário decidir o local onde se realizará a impressão.

Para realizar testes em laboratório, a melhor maneira de testar a impressão direta *in situ* foi sobre um braço de um manequim. Após ter-se tido acesso a esse braço, bastou conseguir obter a nuvem de pontos para se poder realizar a impressão no local desejado.

Existem duas possibilidades de obter a nuvem de pontos:

- Através de um *scanner* 3D com boa presisão e exatidão;
- Através da manipulação manual do *end-effector* do robô.

Numa primeira fase, tentou-se obter a nuvem de pontos através da manipulação manual do *end-effector* do robô. Essa tarefa revelou-se complexa e difícil de executar.

Primeiro, foi necessário que o robô estivesse em compensação gravítica. Segundo, o braço onde se vai realizar a obtenção dos pontos necessitou de estar bem preso de modo a que seja impossível de ser movido. Só depois é que se pode percorrer o braço ou uma zona do braço com o manipulador de modo a obter a nuvem de pontos. O grande problema desta estratégia é que a nuvem de pontos obtida apresentou demasiado ruído causado pela incerteza do movimento realizado para a captura dos pontos, além de apresentar algumas falhas de pontos em certas zonas.

Na figura 7.1 é possível observa o processo assim como a nuvem de ponto obtida.

(a) (b)

Figura 7.1: Captura da nuvem de pontos através da manipulação manual do *end-effector* do robô: (a) Manipulação manual da garra do robô; (b) Nuvem de pontos obtida correspondente à zona abaixo do cotovelo até ao pulso.

Apesar de não se ter acesso a um *scanner* 3D, teve-se acesso a um Tablet que continha um sensor LiDAR[1] (*Light Detection And Ranging)*) e através da aplicação *3D Scanner* foi possível obter a nuvem de pontos do braço do manequim.

Na figura 7.2 é possível observar a aplicação em funcionamento assim com a nuvem de pontos obtida da mesma.

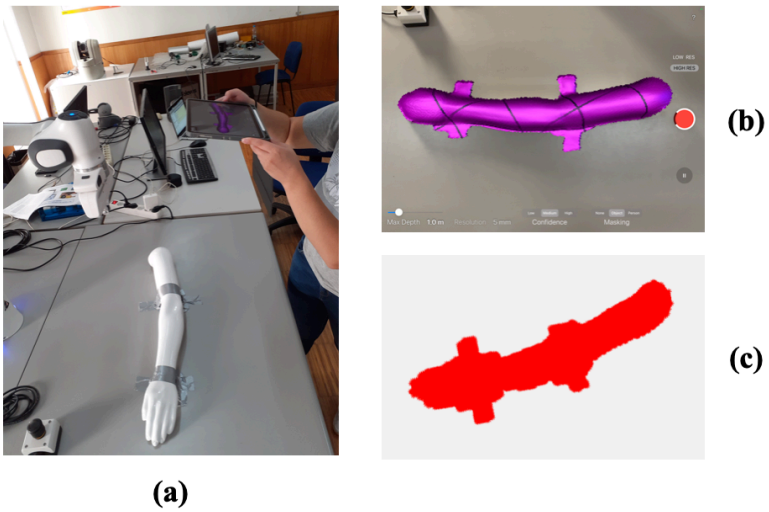


Figura 7.2: Captura da nuvem de pontos através da aplicação *3D Scanner*: (a) Tablet utilizado para a geração da nuvem de pontos; (b) Aplicação *3D Scanner*; (b) Nuvem de pontos obtida através da aplicação.

Recorreu-se ao *software Blender*[2] para fazer a rotação dos pontos e obter a nuvem de pontos na posição deseja. Foi também através deste *software* que se passou as coordenadas da nuvem de pontos[3] para metros uma vez que, a aplicação usada fornecia os pontos em centímetros.

Após obter-se a nuvem de pontos nas unidades corretas e com a rotação desejada, decidiu-se a região onde iria realizar a impressão direta *in situ*, tendo sido escolhida a região abaixo do cotovelo até ao pulso, assim limitou-se a nuvem de pontos a essa mesma região. (cf. figura 7.3).

A razão de não se ter feito o processo de redução da nuvem de pontos como foi feito no subcapítulo 5.4 é que, ao reduzir a nuvem de pontos, iria perde-se informação sobre as características do braço do manequim. Logo, para se conseguir manter as características do braço e reduzir a informação unicamente à necessária, optou-se por utilizar outra

[1] O LiDAR é uma tecnologia que permite determinar a distâncias desde um emissor laser até um objeto ou superfície utilizando um feixe laser por impulso, obtendo uma nuvem de pontos 3D do cenário em que se encontra com uma altíssima velocidade e precisão.

[2] O *Blender* é um programa de computador de código aberto, desenvolvido pela *Blender Foundation*, para modelagem, animação, texturização, composição, renderização e edição de vídeo. Mais informações sobre o Blender podem ser consultadas no site *Blender* [32]

[3] A nuvem de pontos fornece as coordenadas espaciais de cada ponto assim como o vetor normal correspondente.

estratégia.

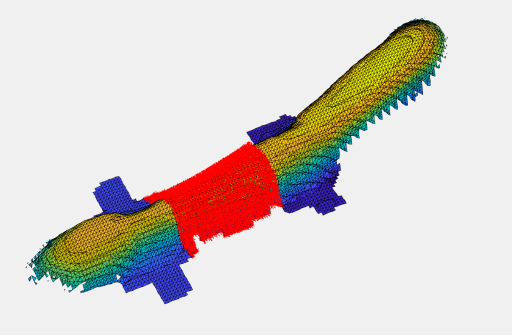


Figura 7.3: Redução da nuvem de pontos à zona onde se vai realizar a impressão 3D.

Em seguida, projetou-se a área onde iria ser realizada a impressão direta *in situ*. Recorrendo à estratégia utilizada nos ensaios experimentais (ver subcapítulo 5.4) projetou-se uma área de 6×2 *cm* em cima da malha de superfície do braço do manequim.

Na figura 7.4 é possível visualizar essa área.

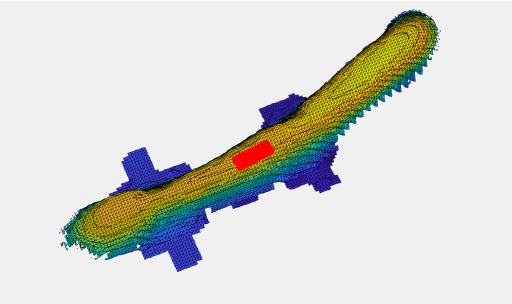


Figura 7.4: Área com 6×2 *cm* projetada sobre a malha de superfície do braço do manequim.

7.1 Ensaio da impressão direta *in situ* no simulador Gazebo

Para possibilitar que o ensaio da impressão direta *in situ* no simulador Gazebo seja o mais próximo da realidade, substitui-se a garra do robô pela peça com o qual se irá realizar a impressão.

Por uma questão de facilidade, considerou-se o chão do simulador Gazebo como sendo o tampo da mesa onde foram colocados o robô PANDA e o braço do manequim.

A figura 7.5 mostra o novo formato do robô PANDA em ambiente de simulação juntamente com o braço do manequim.

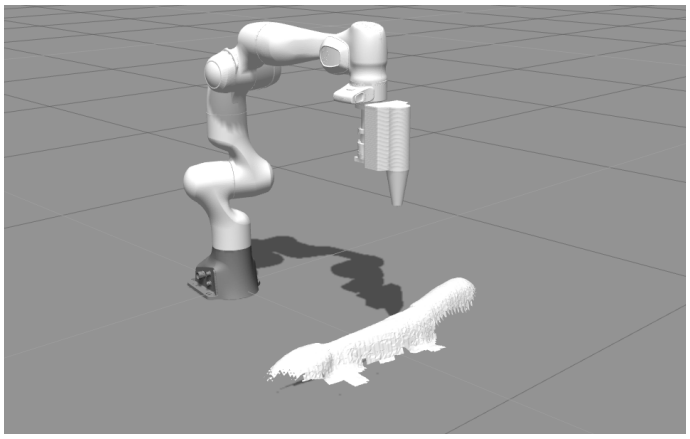


Figura 7.5: Ambiente do simulador Gazebo onde foram colocados o robô PANDA e o braço do manequim.

Ainda antes de se iniciar o ensaio experimental, projetaram-se as referências que vão ser enviadas para o robô num gráfico. Como se pode observar pela figura 7.6 os vetores normais e a posição em z apresentam demasiado ruído, o que poderá causar problemas no ato da impressão.

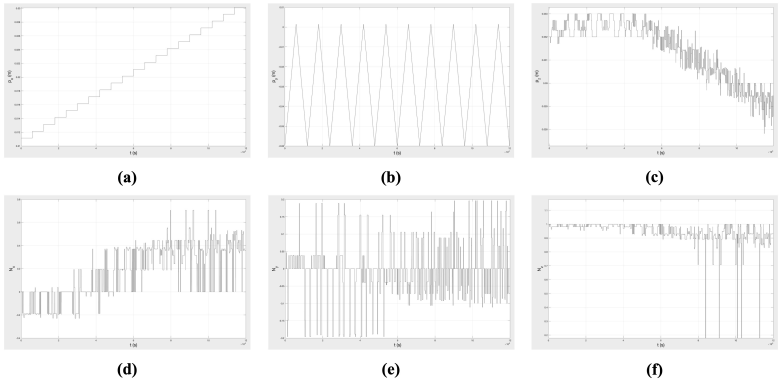


Figura 7.6: Referências obtidas posteriormente a ser traçado o caminho horizontal a ser percorrido pela peça de impressão 3D: (a) p_x; (b) p_y; (c) p_z; (d) N_x; (e) N_y; (f) N_z.

Semelhante ao que se fez na subseção 6.1.2, passou-se as referências por um filtro *butter* passa-baixo de modo a obter um movimento um pouco mais suave. É importante referir que, as referências ao serem passadas por um filtro sofrem um desvio em fase causando um ligeiro desequilibro entre elas. De modo a corrigir esse desfasamento, recorreu-se à função *filtfilt* do *Matlab*. Na figura 7.7 é possível observar a diferença das referências antes do filtro e depois do filtro.

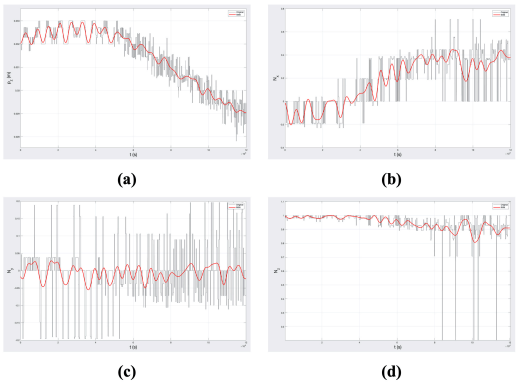


Figura 7.7: Referências obtidas posteriormente a ser traçado o caminho horizontal a ser percorrido pela peça de impressão 3D passadas por um flitro *butter* passa-baixo: (a) p_x; (b) p_y; (c) p_z; (d) N_x; (e) N_y; (f) N_z.

Após se ter estabelecido todas as configurações necessárias, deu-se inicio aos dos ensaios experimentais.

É relevante referir que o processo de impressão direta requer que se encontre acoplada à peça de impressão uma bisnaga. O conteúdo da bisnaga irá ser expelido por uma agulha que se encontra na extremidade da peça de impressão. Por essa razão, a peça de impressão tem de estar 3 milímetros acima da zona onde se vai realizar a impressão direta *in situ*.

Recorreu-se à equação 6.1 para deslocar a peça de impressão para o ponto inicial da nuvem de pontos e, assim, começar a percorrer a área desejada. Num primeiro ensaio a área foi percorrida segundo um movimento horizontal, e num segundo ensaio essa área foi percorrida segundo um movimento vertical.

Ao analisar os gráficos das figuras 7.8 e 7.9 juntamente com a tabela 7.1, é seguro afirmar que o robô apresenta um bom desempenho em simulação durante todo o processo de impressão com uma margem de erro máximo de 1 milímetro. Não existindo praticamente nenhuma diferença entre o movimento horizontal e o movimento vertical.

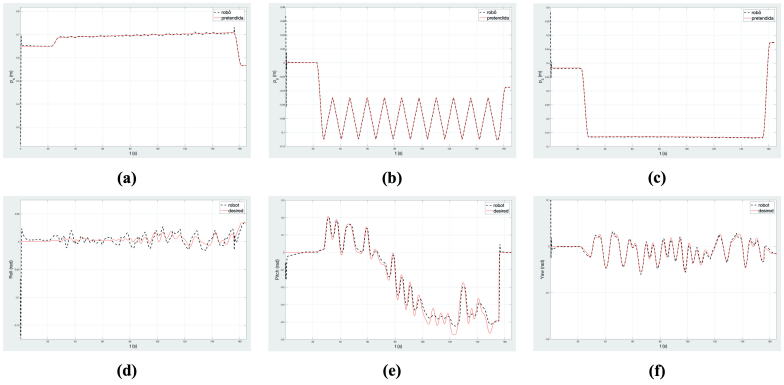


Figura 7.8: Posição e Orientação em PRY referentes à realização do movimento horizontal durante a impressão direta in situ numa área de 6×2 cm com duração de 120 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) N_x; (e) N_y; (f) N_z.

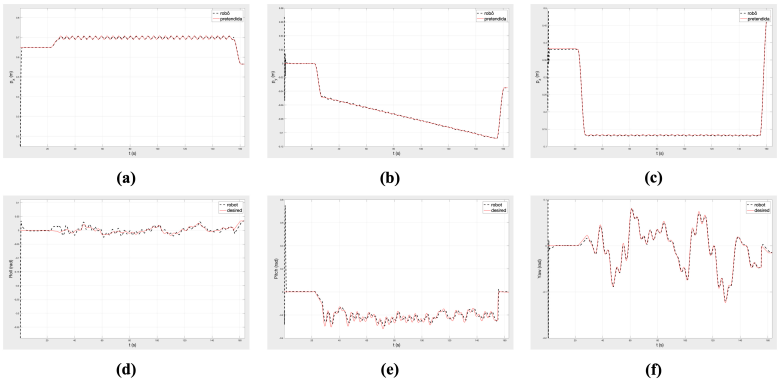


Figura 7.9: Posição e Orientação em PRY referentes à realização do movimento vertical durante a impressão direta *in situ* numa área de 6×2 cm com duração de 120 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) N_x; (e) N_y; (f) N_z.

Tabela 7.1: Tabela da média dos erros obtido nos diferentes movimentos realizados no simulador Gazebo.

Erro/Tempo (s)	Horizontal	Vertical
$\boldsymbol{e_x}$ $\boldsymbol{(m)}$	0.001	0.001
$\boldsymbol{e_y}$ $\boldsymbol{(m)}$	$9.38e^{-04}$	$5.70e^{-04}$
$\boldsymbol{e_z}$ $\boldsymbol{(m)}$	0.001	0.001
$\boldsymbol{Roll(rad)}$	0.007	0.007
$\boldsymbol{Pitch(rad)}$	0.027	0.025
$\boldsymbol{Yaw(rad)}$	0.003	0.003

Os ganhos utilizados para percorrer a área 6×2.5 cm no simulador estão representados na tabela 7.2

Tabela 7.2: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área 6×2 cm).

(a) Ganhos posição

Ganho	**Posição**
$\boldsymbol{K_p}$	$4500I_{3\times3}$
$\boldsymbol{D_p}$	$200I_{3\times3}$

(b) Ganhos orientação

Ganho	**Posição**
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

7.2 Ensaio da impressão direta *in situ* realizada pelo robô PANDA

A última etapa deste projeto corresponde a uma demostração do ensaio da impressão direta *in situ* realizada pelo verdadeiro robô PANDA.

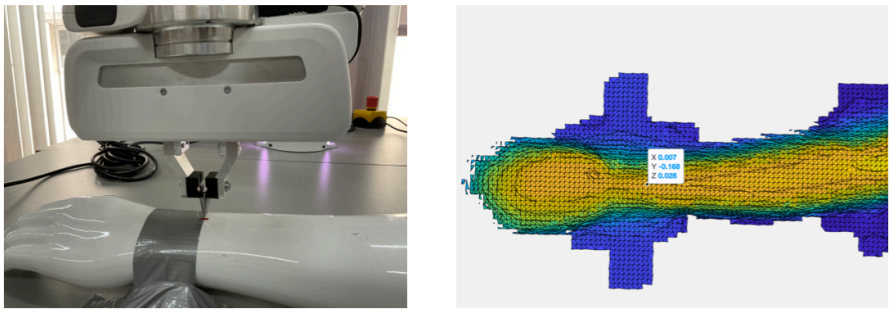
Infelizmente, não foi possível obter a peça com a qual se irá realizar a impressão. Em alternativa, colocou-se na garra do manipulador, uma ponta pontiaguda de uma lapiseira

de modo a simular a ponta da peça utilizada na impressão direta *in situ*, de forma a criar um ambiente mais perto da realidade.

Ao colocar a ponta da lapiseira entre os dedos da garra, existiu a necessidade de fazer alguns acertos no eixo ortonormado da garra. Isto quer dizer que, se teve de deslocar o eixo ortonormado para a ponta da lapiseira, em vez de o lacalizar na ponta da garra. Para isso, mediu-se o comprimento da ponta da lapiseira e deslocou-se o comprimento obtido.

Após se ter feito esta correção, houve a necessidade de fazer a calibração entre os pontos da nuvem de pontos e os pontos do robô. Ou seja, teve-se de passar as coordenadas da nuvem de pontos, que estavam nas coordenadas da câmara, para as coordenadas do robô. Para tal, colocou-se a ponta da lapiseira, presa na garra, numa zona que posteriormente era facilmente identificável na nuvem de pontos, e retirou-se as coordenadas desse ponto. De seguida, pesquisou-se o ponto correspondente na nuvem de pontos e recolheu-se as coordenadas. Por fim, calculou-se o *offset* entre elas e somou-se esse *offset* a todos os pontos da nuvem.

Na figura 7.10 é possível observar o processo de calibração.



(a) **(b)**

Figura 7.10: Calibração das coordenadas dos pontos da nuvem com as coordenadas dos pontos do robô: (a) Panda na zona de calibração; (b) Coordenada na nuvem de pontos na zona de calibração.

Após a calibração ser feita, executou-se a demonstração do processo da impressão direta *in situ*.

A área a ser percorrida foi a mesma que foi usada na simulação e foram realizados vários ensaios até se ter atingindo um resultado favorável.

Os gráficos apresentados na figura 7.11 representam o trajeto realizado pelo robô durante 120 segundos seguindo um movimento horizontal.

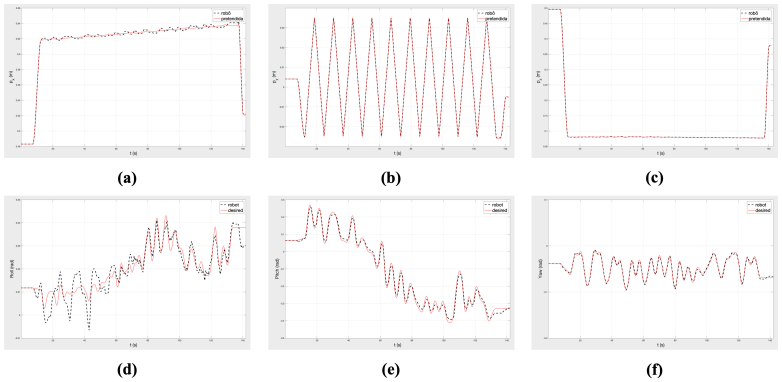


Figura 7.11: Posição e Orientação em PRY referentes à realização do movimento vertical durante a impressão direta *in situ* numa área de 6×2 cm com duração de 120 segundos. (Simulador Gazebo): (a) p_x; (b) p_y; (c) p_z; (d) N_x; (e) N_y; (f) N_z.

Ao comparar a figura 7.11 com a figura 7.8 é notório que o robô apresenta um bom comportamento muito semelhante à simulação, tendo um erro quadrático médio na ordem de 1 milímetro em posição e um erro inferior a 1 grau na orientação.

Contudo, apesar do seu desempenho ser favorável, ele realiza o movimento ligeiramente afastado do local onde deveria realizar a impressão direta *in situ*, devido a incorreções na captura da nuvem dos pontos através do sensor LiDAR, observando-se erros na nuvem de pontos na ordem dos 1 a 2 centímetros.

Na tabela 7.3 estão representados os ganhos usados no controlador de impedância.

Tabela 7.3: Ganhos de controlo de impedância no espaço de tarefa, onde a variável I em (a) e (b) representam a matriz identidade. (Área 6×2 cm).

(a) Ganhos posição

Ganho	Posição
$\boldsymbol{K_p}$	$5000I_{3\times3}$
$\boldsymbol{D_p}$	$50I_{3\times3}$

(b) Ganhos orientação

Ganho	Posição
$\boldsymbol{K_0}$	$200I_{3\times3}$
$\boldsymbol{D_0}$	$10I_{3\times3}$

8

Conclusão

Apesar das dificuldades que foram surgindo durante o decorrer do desenvolvimento deste projeto, é legítimo afirmar que o objetivo principal deste trabalho foi alcançado.

Conseguiu-se desenvolver um algoritmo que, ao receber uma nuvem de pontos arbitrária consegue traçar caminhos simples que posteriormente irão ser percorridos pelo robô PANDA.

Os resultados obtidos no simulador Gazebo são bastantes promissores, permitindo preparar e testar diferentes trajetórias em diferentes nuvens de pontos antes de serem feitos testes no robô real.

O controlador de impedância implementado também apresentou bons resultados de performance em ambos os ambientes (simulação e real), tendo como erro médio máximo um valor na ordem de 1 milímetro na componente posição e um erro inferior a 1 grau na componente associada à orientação. Resultados importantes quando se trata de robôs customizados para a indústria da medicina robótica.

Embora os resultados alcançados no verdadeiro robô PANDA serem bastante favoráveis, existe um erro associado ao modo de obtenção da nuvem de pontos, que faz com que o robô realize o seu movimento ligeiramente afastado do local pretendido. Os sensores LiDARs são capazes de atingir uma precisão de alcance de 0.5 a 10 milímetros com uma exatidão de mapeamento até 1 centímetros nas coordenadas x e y e 2 centímetros na coordenada z, o que para certas situações é o suficiente, mas quando se trata de tratamentos cirúrgicos, são imprecisões demasiados elevadas e prejudiciais. O ideal é obter a nuvem de pontos com sensores, scanners ou câmaras de alta resolução, precisão e exatidão na ordem dos micrômetro ou nanômetro.

Por último, este projeto de dissertação abre portas para futuros investigadores que queiram trabalhar na área da robótica colaborativa.

Contudo, muitas adaptações, ensaios e experiências foram deixadas para o futuro. Por isso, são deixadas algumas sugestões:

- Melhorar o algoritmo de geração de caminhos, torná-lo capaz de elaborar caminhos mais complexos, não geométricos e menos genéricos;

- Realizar testes e ensaios no robô PANDA com a verdadeira peça de impressão;
- Obter uma nuvem de pontos através de uma câmara ou scanner de alta resolução, exatidão e precisão e, realizar teste com os resultados obtidos dessa nuvem.

Bibliografia

[1] Fateme Fayyazbakhsh and Ming C. Leu. A brief review on 3d bioprinted skin substitutes. *ScienceDirect*, 48:790–796, 2020.

[2] Pallab Datta, Ananya Barui, Yang Wu, Veli Ozbolat, Kazim K. Moncal, and Ibrahim T. Ozbolat. Essential steps in bioprinting: From pre- to post-bioprinting. *Biotechnology Advances*, 36:1481–1504, 2018.

[3] Xiao Li, Qin Lian, Dichen Li, Hua Xin, and Shuhai Jia. Development of a robotic arm based hydrogel additive manufacturing system for in-situ printing. *Applied Sciences (Switzerland)*, 7, 2017.

[4] R. Bloss. Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications. *Industrial Robot: An International Journal*, 43:463–468, 2016.

[5] Y. Kwoh, J. Hou, E. Jonckheere, and S. Hayati. A robot with improved absolute positioning accuracy for ct guided stereotactic brain surgery. *IEEE Transactions on Biomedical Engineering*, 35:153–160, 1988.

[6] Lan Li, Jianping Shi, Kaiwei Ma, Jing Ji, Peng Wand, Huixin Liang, Yi Cao, Xingson Wang, and Quing Jiang. Robotic in situ 3d bio-printing technology for repairing large segmental bone defects. *Journal of Advance Research. How Science Improves Society*, 30, 2020.

[7] Franka Emika. `https://www.franka.de`.

[8] Franka Emika GmbH. *PANDA'S Instruction Handbook*.

[9] Hélio Ochoa and Rui Cortesão. Tooling4g manual, 2020.

[10] Web Panda Desk. `https://world.franka.de`.

[11] Franka Emika. *Control Interface Documentation*.

[12] Franka Emika. *Official ROS Packages for Panda*.

[13] ROS community. ROS Control. `http://wiki.ros.org/ros_control`.

[14] ROS community. ROS Control and Gazebo. `http://wiki.ros.org/gazebo`.

[15] ROS community. ROS MoveIt. `https://moveit.ros.org`.

[16] ROS community. ROS RViz. `http://wiki.ros.org/rviz`.

[17] The Orocos Project. `https://orocos.org`.

[18] ROS community. ROS URDF. `http://wiki.ros.org/urdf`.

[19] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modeling, Planning and Control*. 2009.

[20] Rui Cortesão. Medical robotics, 2020.

[21] Hélio Ochoa and Rui Cortesão. Control architecture for robotic-assisted polishing tasks based on human skills. *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, pages 630–637, 2019.

[22] Luís Santos and Rui Cortesão. Computed-torque control for robotic-assisted tele-echography based on perceived stiffness estimation. *IEEE Transactions on Automation Science and Engineering 15.3*, pages 1337–1354, 2018.

[23] Christian Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. 2008.

[24] Jinjun Duan, Gan Duan, Yahui, Chen, Ming, Dai, and Xianzhong. Adaptive variable impedance control for dynamic contact force tracking in uncertain environment. *Robotics and Autonomous Systems*, 102:54–65, 2018.

[25] J. J. Craig. *Introduction To Robotics: Mechanics and Control*. 3 edition, 2005.

[26] Williams and Robert L. Simplified robotics joint-space trajectory generation with a via point using a single polynomial. *Journal of Robotics*, 2013.

[27] Sahar G. and J.M. Hollerbach. Planning of minimum-time trajectories for robot arms. *IEEE International Conference on Robotics and Automation*, 1985.

[28] W.D. Lfisher and M.S Mujtaba. Minimum ratio-locked profile times for robot trajectories. *IEEE International Conference on Robotics and Automation*, 1988.

[29] Artec3d. `https://www.artec3d.com`.

[30] Autodesk. Fusion 360. `https://www.autodesk.pt/products/fusion-360/overview`.

[31] MathWorks. Matlab. `https://www.mathworks.com/products/matlab.html`.

[32] Blender. `https://www.blender.org`.

A

PANDA Especificações

FRANKA EMIKA

DATASHEET [1]
ROBOT ARM & CONTROL

Release Version: April 2020

HARDWARE

Arm

Degrees of freedom	7
Payload	3 kg
Workspace	see backside
Maximum reach	855 mm
Force/ Torque sensing	link-side torque sensors in all 7 axes
Expected nominal lifetime [3,4]	20,000 h
Joint position limits	A1, A3, A5, A7: -166°/166° A2: -101°/101° A4: -176°/-4° A6: -1°/215°
Mounting flange	DIN ISO 9409-1-A50
Installation position	upright
Weight	~ 17.8 kg
Moving mass	~ 12.8 kg
Protection rating	IP30
Ambient temperature [2]	15 – 25 °C (typical) 5 – 45 °C (extended)
Air humidity	20 – 80 % non-condensing
Power consumption	• max. ~ 350 W • typical application ~ 60 W
Interfaces	• ethernet (TCP/IP) for visual intuitive programming with Desk • input for external enabling device • input for external activation device or safeguard • Control connector • Connector for end effector

Control

Controller size (19")	355 x 483 x 89 mm (D x W x H)
Supply voltage	100 – 240 V_{AC}
Mains frequency	47 – 63 Hz
Power consumption	~ 80 W
Active power factor correction (PFC)	yes
Weight	~ 7 kg
Protection rating	IP20
Ambient temperature	15 – 25 °C (typical) 5 – 45 °C (extended)
Air humidity	20 – 80 % non-condensing
Interfaces	• ethernet (TCP/IP) for internet and/or shop-floor connection • power connector IEC 60320-C14 (V-Lock) • Arm connector

1. Technical data are subject to change.
2. Lifetime and performance can potentially be reduced when operating outside the typical temperature range.
3. Based on ISO 9283 (Annex A), specified values refer to a workspace of 0.4 x 0.4 x 0.4 m centered at [0.515, 0.0, 0.226] m, with the Z-Axis of the flange oriented parallel to earth-gravity and the elbow positioned upwards.
4. Nominal conditions (66% load).

SOFT-ROBOT PERFORMANCE

Motion

Joint velocity limits	A1, A2, A3, A4: 150°/s A5, A6, A7: 180°/s
Cartesian velocity limits	up to 2 m/s end effector speed
Pose repeatabillity	<+/- 0.1 mm (ISO 9283)
Path deviation [3]	<+/- 1.25 mm

Force

Sensing [3]

Force resolution	<0.05 N
Relative force accuracy	0.8 N
Force repeatability	0.15 N
Force noise (RMS)	0.035 N
Torque resolution	0.02 Nm
Relative torque accuracy	0.15 Nm
Torque repeatability	0.05 Nm
Torque noise (RMS)	0.005 Nm

1 kHz Control [3]

Minimum controllable force (Fz)		0.05 N
Force controller bandwidth (-3 dB)		10 Hz
Force range [N]	Nominal case	Local best case
Fx	-125 – 95	-150 – 115
Fy	-100 – 100	-275 – 275
Fz	-50 – 150	-115 – 155
Torque range [Nm]	Nominal case	Local best case
Mx	-10 – 10	-70 – 70
My	-10 – 10	-16 – 12
Mz	-10 – 10	-12 – 12

Interaction

Guiding force	~ 2 N
Collision detection time	<2 ms
Nominal collision reaction time [3,4]	<50 ms
Worst case collision reaction time [3]	<100 ms
Adjustable translational stiffness	0 – 3000 N/m
Adjustable rotational stiffness	0 – 300 Nm/rad
Monitored signals	joint position, velocity, torque cartesian position, velocity, force

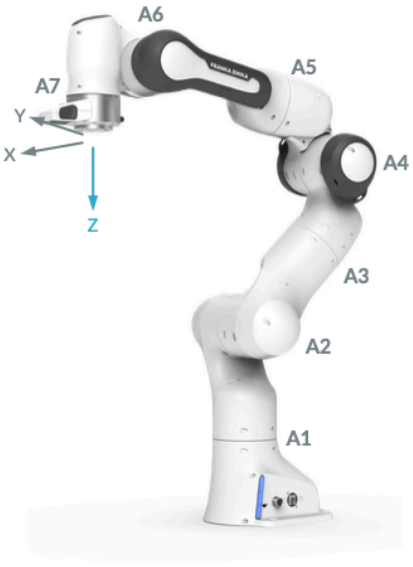
ADD-ONS

Safety retrofit option with safety-rated PLC	PLd Cat. 3 • Safe torque off (STO) • Safe OSSD inputs
Fully integrated end effectors	• 2-finger gripper • Vacuum gripper
Fast mounting	Clamping Adapter
Demonstration	Pop-up Box
Research interface	1kHz Franka Control Interface (FCI)
Fieldbuses	Modbus/TCP, OPC UA

Additional technical specifications

Respect torque limits for each joint at all times:

- Axes 1 & 2: allowed, repeatable peak torque <= 87 Nm
- Axes 3 & 4: allowed, repeatable peak torque <= 87 Nm
- Axes 5, 6, 7: allowed, repeatable peak torque <= 12 Nm



The mechanical zero position of each joint is reached when the two triangles on each side of the gap between the Arm segments align.

Joints' mechanical zero positions

No planned maintenance or service intervals within expected nominal lifetime.

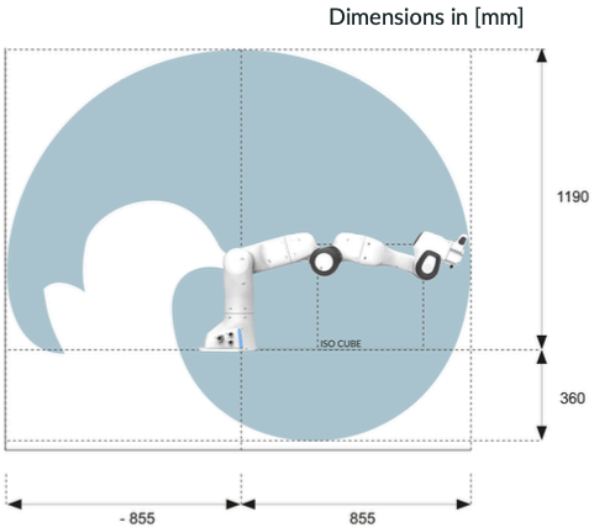
Planned maintanance

Panda robot system's weighted emission sound pressure level at workstations does not exceed 70 dB (EU Machinery Directive 2006/42/EC).

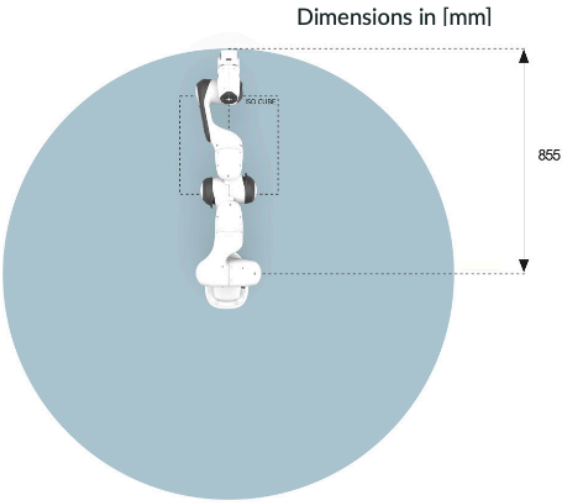
Noise emission

Reachable Space of Panda

Side-view: reachable space for the end effector flange



Top-view: reachable space for the end effector flange



B

Instalação Linux

Neste apêndice é explicado como instalar o repositório 3DPrint num sistema *Linux*.

O primeiro passo a ser feito, é decidir qual a distribuição ROS que se pretende usar. As recomendadas são:

- *Ubuntu 16.04 LTS Xenial Xerus e ROS Kinetic Kame*;
- *Ubuntu 18.04 LTS Bionic Beaver e ROS Melodic Morina*

De seguida é necessário instalar a *libfranka*. Para tal, basta seguir os seguintes passos:

user@hostname$ sudo apt install build-essential cmake git libpoco-dev libeigen3-dev

user@hostname$ git clone –recursive https://github.com/frankaemika/libfranka

user@hostname$ cd ibfranka

user@hostname$ cmake -DCMAKE_BUILD_TYPE=Release ..

user@hostname$ cmake –build .

user@hostname$ cd /path/to/desired/folder

user@hostname$ mkdir -p catkin_ws/src

user@hostname$ catkin_ws

user@hostname$ source /opt/ros/melodic/setup.sh

user@hostname$ catkin_init_workspace src

Após a *libfranka* ser instalada é necessário clonar o repositório 3DPrint.

user@hostname$ git clone –recursive https://github.com/RitaGama/3DPrint.git

Por último, instalar dependências em falta:

user@hostname$ rosdep install –from-paths src –ignore-src –rosdistro melodic -y --skip-keys libfranka

user@hostname$ sudo apt-get install libsfml-dev

user@hostname$ catkin_make -DCMAKE_BUILD_TYPE=Release -DFranka_DIR:PATH=/path/to/libfranka/build

user@hostname$ source devel/setup.sh

Após todas as instalações terem sido feitas, é importante configurar o *kernel* em tempo real, caso contrário não será possível controlar o robô PANDA.

Para instalar o *kernel Linux* em tempo real, basta seguir o tutorial disponível na documentação da FCI (`https://frankaemika.github.io/docs/installation_linux.html?highlight=real%20time%20kernel`).

Posteriormente ao *kernel* ser instalado, é necessário configurar a rede do robô PANDA uma vez que, o computador da estação de trabalho que comanda o robô deve estar sempre ligado à *LAN port* do controlador e não à *LAN port* do braço robótico, (ver figura 2.2). Para configurar a rede basta seguir o tutorial disponível na documentação da FCI (`https://frankaemika.github.io/docs/installation_linux.html?highlight=real%20time%20kernel`).

C

Como utilizar o repositório 3DPrint

C.1 Simulador Gazebo

Para lançar o robô PANDA no simulador Gazebo é preciso abrir o terminal e digitalizar na linha de comandos:

roslaunch franka_simulation cartesian_impedance_controller.launch

Após comando ser executado, irão abrir 3 janelas diferentes.

- Simulador Gazebo
- *RViz*
- Reconfiguração Dinâmica

Esta última janela, permite afinar os ganhos do controlador online.

Para lançar o robô PANDA no simulador Gazebo para a realização da impressão direta *in situ* é necessário abrir dois terminais.

No primeiro terminal é lançado o *roslaunch*:

roslaunch franka_simulation print_controller.launch

No segundo terminal é onde se executa a tarefa de impressão, com a linha de comando:

rosrun franka_simulation simulation_print_node

C.2 Robô PANDA

Para configurar o robô PANDA recorre-se ao uso da pasta *franka_spacenav*, para tal digita-se na linha de comandos:

roslaunch franka_spacenav cartesian_impedance_controller.launch

Para visualizarmos o robô PANDA a percorrer o caminho desejado, abre-se um segundo terminal e digita-se:

rosrun franka_spacenav print_node

C.3 Matlab

O repositório *3DPrint* recebe os ficheiros gerados do repositório *3DPrint_Matlab*.

Este ficheiro permite gerar dados, tratá-los e posteriormente analisá-los.

O repositório *3DPrint_Matlab* pode ser obtido através do seguinte link (`https://github.com/RitaGama/3DPrint_Matlab.git`).