FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Planeamento de trajetórias em Manipuladores em ambientes industriais

Pedro Miguel Santos Tavares

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Pedro Luís Cerqueira Gomes da Costa (Professor Doutor)

Co-orientador: José Luís Sousa de Magalhães Lima (Professor Doutor)

28 de Julho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Planeamento de Trajetórias em Manipuladores em Ambientes Industriais"

foi aprovada em provas realizadas em 16-07-2015

o júri

Presidente Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Luís Paulo Gonçalves Reis
Professor Associado do Departamento de Sistemas de Informação da Escola de Engenharia da Universidade do Minho

Professor Doutor Pedro Luís Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Pedro Miguel Santos Tavares

Faculdade de Engenharia da Universidade do Porto

Resumo

O desenvolvimento científico e tecnológico associado ao mundo da robótica encontra-se em constante evolução, motivado pela necessidade de encontrar soluções às atuais alternativas e pela ambição do ser humano em desenvolver sistemas de deslocamento cada vez mais competentes. Neste contexto surge a necessidade de estudo de algoritmos capazes de planear de forma eficaz e segura toda a movimentação de um robot. O principal objetivo deste projeto é o desenvolvimento e implementação de um planeamento de trajetória em manipuladores e em ambiente industrial, sem nunca esquecer os passos intermédios inerentes a este processo, nomeadamente o reconhecimento do cenário envolvente ao robot. Este documento apresenta, assim, a história e parte da evolução já realizada pela comunidade robótica neste domínio, bem como as soluções existentes até este momento, tentando, paralelamente, mostrar quais os pontos passíveis de serem melhorados. Posteriormente é proposta uma solução flexível e genérica implementada para operações de *pick-and-place*, com o uso de um planeador de trajetória inovador. Este planeador assemelha-se aos planeadores da família A* e tem associado a necessidade de discretização de espaço e de definição de equações de cinemática que regem a movimentação do robot. Ao longo do documento será ainda descrita a importância deste projeto e da evolução do ramo robótico como uma necessidade para o mundo industrial e empresarial, sendo apresentados dois exemplos, o *European Robotic Challenge* (EuRoC) e o *Amazon Picking Challenge*.

Abstract

The scientific and technological development together with the world of robotics is constantly evolving, driven by the need to find new solutions and by the ambition of the human beings to develop systems with increasingly efficiency. Consequently, it is necessary to study planning algorithms capable of effectively and safely move a robot within a given scene. The main objective of this project is to develop and to implement a path planning methodology for automatic manipulators in the industrial environment. The intermediate steps inherent to this process, such as the recognition of the scene, will also be investigated in this project. This document presents the history and key discoveries in this area that have been reported by the robotics community, as well as the currently available solutions for path planning problems. Proposed improvements to currently available methods will be also described in this project and the use of the new path planning algorithm develop during the project will also be explained. This planner is similiar to the ones associated to the A* family and has associated the need of space discretization and the definition of kinematics equations that rule the robot's movement. Throughout the document, the importance of this project and the development of novel robots will be further described taken into consideration the need for robots in the industrial setting. It will also be presented two use cases: the European Robotic Challenge (EuRoC) and the Amazon Picking Challenge.

Agradecimentos

Nesta secção queria aproveitar para agradecer ao meu orientador, Pedro Costa, e ao meu co-orientador, José Lima, pela oportunidade de participar neste projeto e por toda a ajuda e dedicação que sempre disponibilizaram. Bem sei que também gostariam de me agradecer todas as novelas e conversas de barbeiro (por ser mais másculo que cabeleireiro) que permitiam abstrair por cinco minutos do projeto para logo depois recomeçar no mesmo com muita mais motivação.

Uma palavra para outros Professores que me deram a mão para todos os desafios que tive ao longo do curso entre eles António Paulo Moreira e Armando Sousa pela disponibilidade e propostas que permitiram crescer como *Engenheiro* e ao Professor Adriano Carvalho por me ter constantemente desafiado e me ter incutido o espírito de querer sempre melhorar.

Aqui, torna-se importante inserir um especial obrigado também às pessoas que contribuíram para o desenvolvimento de todo o projeto desta tese, nomeadamente as pessoas dos laboratórios I-108, I-109 e I-110. Sei que fui muito *chatinho* em várias ocasiões, mas a sua persistência e interesse em ajudar foi uma mais-valia para recordar.

Num aspeto mais íntimo queria agradecer aos meus pais, irmã e avós por me ajudarem em todo o percurso académico e pessoal e por me aturarem em fases de grande incoerência emocional. Muitas vezes preteri da sua excelente companhia em prol desta tese e, bem sei, que o lugar vazio foi sentido. Ainda assim, sempre tiveram uma palavra amiga ou um abraço a dar na altura certa.

Um especial obrigado à minha companheira Andreia, que ao longo de todo este trajeto que me ajudou em todos os momentos bons e maus, que me ajudou a ultrapassar obstáculos que me pareceram em parte impossíveis, me incentivou a querer ser sempre o melhor e me fez valorizar cada momento como único. Todos as etapas que alcancei, em parte lhe devo.

Aos meus GRANDES amigos: Rui Barros e Miguel Abreu. Rui por me aturar em momentos de desespero e por ter sempre a porta aberta para aquilo que um verdadeiro amigo deve fazer (ceder um sofá, um jogo de bilhar ou de FIFA). Miguel por se revelar mais que um colega, um amigo para a vida. Os nossos jantares semanais onde se partilhava mais que vivências comuns nunca serão esquecidos como um grande incentivo para esta etapa.

Neste percurso muitos mais me marcaram pela sua energia, disponibilidade e paciência para me aturar. Aos rapazes Luís "Ursula" Leça (ou Bessa dependendo das ocasiões), Valter "Besta" Costa, Ricardo "Brasileirada" Carvalho, Hugo "Longroiva" Costa, Tiago "Monstro" Cunha, Filipe "Ordinário" Lopes, Peter "SALTY ONION" Cebola, João "Terceira Idade" Vaz, Filipe "Glorioso Líder" Santos, um muito obrigado pela companhia para os finos, pelas noites passadas em claro e pelo companheirismo que sempre demonstraram e às donzelas Ursula (fiel companheira do Luís), Susana "Mãe" Neves, Diana "Bambu" Neves, Cláudia "Mini Javali" Rocha por alegrarem todos os dias desta longa caminhada.

A todos, um sincero obrigado!

Pedro Tavares

"Tudo parece impossível até ser feito"

Nelson Mandela

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

θ_i	Configuração de Articulação de um Robot Manipulador
T_y^x	Matriz de Transformação Linear
A*	Algoritmo A*
DH	Método Denavit e Hartenberg
EuRoC	European Robotic Challenge
IFR	International Federation of Robotics
OMPL	Open Motion Planning Library
OpenRAVE	Open Robotics Automation Virtual Environment
RGBD	Red, Green, Blue, Depth
ROS	Robot Operative System
SMEs	Small and Medium-Size enterprises (pequenas e médias empresas)
UR5	Robot Universal 5
YAML	Extensão de ficheiros tipo XML (Yet Another Markup Language)

Capítulo 1

Introdução

Os manipuladores robóticos móveis têm suscitado um interesse substancial por parte da comunidade de engenharia associada à pesquisa e desenvolvimento tecnológico; assim como, por parte da comunidade industrial enquanto cliente capaz de receber e adotar esta tecnologia que potencializa a maximização de eficiência dos processos automatizados e dos processos de produção em geral. Exemplos de aplicação destes robots incluem [1, 2, 3]:

- Transporte de equipamento.
- Operações de *pick-and-place*.
- Na prática médica e em indústrias farmacêuticas [2].
- Operações perigosas e inacessíveis ao ser humano. Por exemplo, vários robots têm sido desenvolvidos para fins militares e para aplicações de defesa, incluindo deteção de explosivos, vigilância e operações de logística ou de salvamento [4].

A Federação Internacional de Robótica (IFR, *International Federation of Robotics*), na sua mais recente revisão do crescimento na área da robótica, além de confirmar a tendência crescente do desenvolvimento de soluções e processos da robótica industrial, prevê que este crescimento se mantenha durante os próximos anos, sendo este considerável crescimento fundamental para o contínuo desenvolvimento da indústria (ver figura 1.1) [5]. Mais ainda, o mesmo estudo aponta para um crescimento do mercado robótico, o que revalida o considerável interesse que esta área tem suscitado, desde logo, pelas suas vantagens de eficiência em processos automatizados, mas também pela capacidade de integração em todos os ambientes industriais.

Os robots atuais são desenvolvidos de forma a serem capazes de se adaptarem a diversas aplicações e cenários, permitindo assim uma interação frequente com o operador humano. Estes são também essenciais em tarefas de elevada força/precisão e são desenvolvidos com o objetivo de serem financeiramente rentáveis e sustentáveis em pequenas áreas e com cargas de trabalho reduzidas [6]. Para além disso, têm ainda sido feitos esforços no sentido de se desenvolver robots capazes de responder a mudanças de forma, cor ou tamanho dos diversos objetos com os quais

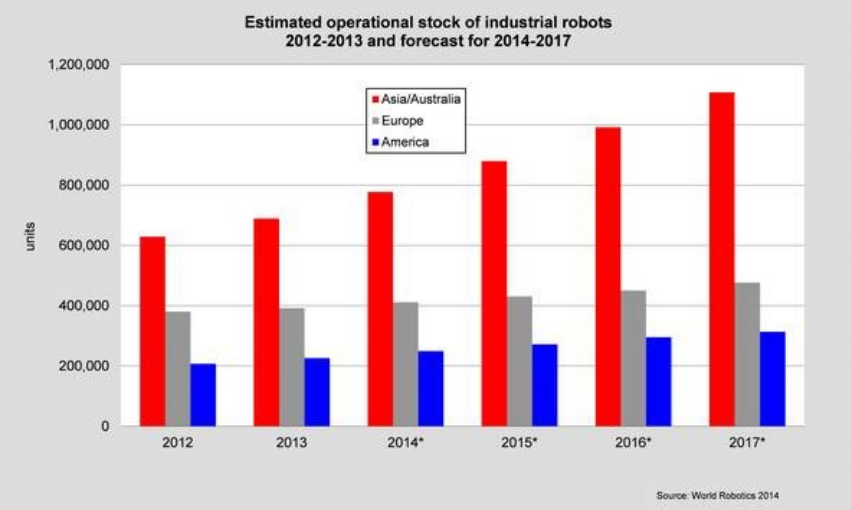

Figura 1.1: Evolução do número de robots operacionais nos últimos anos e sua perspetiva futura [5]

terão que trabalhar, ajustando-se a diferentes configurações e redefinindo movimentos, ações, entre outros.

Atualmente o crescimento organizacional é dependente do desenvolvimento de pequenas e médias empresas (*small and medium-size enterprises, SMEs*) [7]. Assim, robots atuais e futuros são desenvolvidos por forma a serem facilmente adaptados a uma grande variedade de aplicações e a permitirem uma interação frequente e segura com o operador humano. Simultaneamente estes robots têm de surgir como uma solução viável para as *SMEs* em termos de serem lucrativos em pequenas zonas de trabalho e com cargas de trabalho reduzidas [8].

Contudo, e apesar da tremenda e crescente utilidade desta tecnologia, ainda existem desafios e limitações na área de robótica que requerem trabalho futuro. Por exemplo, muitos são os robots sem a flexibilidade de ajuste a diferentes tarefas. Por outro lado, o código que coordena as respostas dos robots não é frequentemente implementado usando ferramentas de construção modular como o *Robotic Operative System* (ROS) ou o *Open Robotics Automation Virtual Environment* (OpenRAVE) [9, 10, 11]. O ROS é um conjunto de bibliotecas e ferramentas de *software*, nomeadamente drivers e algoritmos de última geração, que podem ser incorporadas em aplicações robóticas. O *OpenRAVE* oferece um ambiente desenhado para teste, desenvolvimento e implementação de algoritmos associados ao planeamento de movimento em aplicações robóticas de parâmetros realísticos; isto é, planeamento de trajetória e execução desse caminho. O principal objetivo desta ferramenta é a simulação e análise da cinemática e das informações geométricas associadas ao planeamento do movimento. Para além de serem ferramentas modulares, outra vantagem do uso de ROS ou *OpenRAVE* reside no facto de estes serem acessíveis a qualquer engenheiro ou programador, pois são *open-source* e *user-friendly*.

Aspetos importantes a considerar quando se desenvolve um novo robot incluem a seleção da metodologia para reconhecimento do ambiente envolvente e a capacidade de resposta rápida e efi-

ciente a esse mesmo ambiente. Ambos contribuem para que o robot execute um bom planeamento de trajetória, e é nesta dimensão que surge a motivação para a resolução do problema discutido neste projeto.

1.1 Problema

O problema principal a focar neste trabalho relaciona-se com a necessidade de desenvolvimento de estratégias de planeamento de trajetória de um robot, procurando maximizar o proveito de cada estratégia em ambientes industriais e aplicando essas soluções a situações reais.

De forma a resolver o problema apresentado, este projeto procederá à implementação de um planeamento de trajetória de um robot desenvolvido em ambiente ROS, com recurso às bibliotecas *OpenRAVE*. Pretende-se que tal planeamento tenha ainda em conta a possibilidade do ambiente poder ser irregular, com a existência de objetos estáticos e em movimento. É ainda importante que o planeamento do robot seja simples e rápido, de modo a que a cooperação e a colaboração com outros manipuladores e com seres humanos seja possível e sustentada.

1.2 Análise do Problema

Pode-se dividir a análise do problema exposto na secção anterior em várias etapas:

1. Compreensão das ferramentas de desenvolvimento modular como ROS e OpenRAVE.
2. Discretização espacial do espaço de robot e identificação das equações de cinemática que regem a movimentação do robot.
3. Necessidade de ter uma arquitetura modular e flexível capazes de ser transversal a várias aplicações robóticas.
4. Criar algoritmos robustos e eficientes.

Para resolver estes problemas foi definida a seguinte ordem de trabalhos:

1. Estudo do Estado-da-Arte.
2. Estudo da ferramenta ROS e biblioteca OpenRAVE.
3. Estudo do planeamento de trajetórias em manipuladores.
4. Estudo da Cinemática de manipuladores robóticos.
5. Desenvolvimento e implementação do planeamento de trajetórias manipuladores.
6. Desenvolvimento do sistema de controlo.
7. Teste e validação final.

Tendo em conta a divisão do problema nas fases descritas acima, torna-se fundamental, numa fase inicial, a realização de uma pesquisa bibliográfica que visa o levantamento do estado-da-arte no campo da robótica, especificamente investigação das diversas ferramentas de planeamento de trajetórias previamente descritas e em desenvolvimento. Paralelamente, visto a necessidade de se desenvolver um robot flexível e fácil de se associar a vários sistemas, torna-se necessária a realização de um levantamento de quais as plataformas modulares mais adequadas a este projeto. Por ser um dos requisitos deste trabalho, as plataformas modulares ROS e *OpenRAVE* serão o foco deste projeto, sendo, respetivamente, uma ferramenta de desenvolvimento de *software* e uma biblioteca de suporte ao desenvolvimento de *software* [12, 13].

Numa fase mais avançada de análise do problema, é também importante o estudo e a escolha de um ambiente integrado de desenvolvimento (IDE, *Integrated Development Environment*), ou seja uma ferramenta de integração de desenvolvimento de *software*, bem como um ambiente de simulação para testes desse mesmo desenvolvimento. De igual importância é o desenvolvimento de formas de reconhecimento do ambiente em que o robot se insere. Dado que um dos objetivos deste projeto é o desenvolvimento de robots desenhados para ambientes industriais, na fase de desenvolvimento do projeto é também importante validar estas ferramentas em manipuladores direcionados para esses mesmos ambientes.

1.3 Estrutura da Dissertação

Este documento está organizado em seis secções. Na primeira secção (capítulo 2) serão apresentados os conteúdos que resultam de uma pesquisa bibliográfica do estado-da-arte relacionado com o projeto, desde a história de manipuladores industriais, as suas principais operações e os tipos existentes até às formas de reconhecimento dos ambientes industrias e aos algoritmos de planeamento de trajetória. Na secção seguinte (capítulo 3) serão apresentadas os fundamentos de espaço de configurações e cinemática associada a robótica. Será ainda aprofundado estes tópico para um robot específico. Seguidamente, no capítulo 4 é proposta uma abordagem flexível e genérica para as aplicações de *pick-and-place*. Mais ainda, são apresentados exemplos de aplicação, mais propriamente duas competições de robótica onde tal abordagem pode ser expressa. No capítulo 5 são apresentadas as soluções desenvolvidas ao longo de todo o projeto desenvolvido, nomeadamente o planeador inovador e as soluções implementadas para os casos de aplicação descritos no capítulo anterior. Imediatamente a seguir (capítulo 6) descreve os resultados produzidos pelas soluções implementadas, comparando em alguns casos com algoritmos e soluções já existentes no mundo industrial. Finalmente, na última secção (capítulo 7) serão apresentadas as conclusões retiradas deste projeto e as perspetivas para o desenvolvimento futuro do mesmo.

Capítulo 2

Revisão Bibliográfica

O desenvolvimento robótico está associado ao progresso tecnológico e industrial desde 1938, ano em que se desenvolveu uma pequena estrutura semelhante a uma grua capaz de fazer pequenas construções de forma automática [14]. No entanto, o robot industrial é apenas reconhecido duas décadas mais tarde. Em 1959, George Devol e Joseph Engelberger desenvolveram o primeiro robot industrial oficial [15]. Como é expectável, o robot então desenvolvido era demasiado pesado (2 toneladas), usava atuadores hidráulicos e a sua precisão era reduzida. Para além disso, o facto de não ser um robot inteligente, no sentido em que apenas replicava sempre o mesmo movimento, permite-nos perceber a evolução até à atualidade. Num artigo publicado pela IFR em 2012, descrevem-se excertos dessa evolução no mundo da robótica ao longo dos anos, nomeadamente a introdução da inteligência na robótica e a condensação e simplificação na construção dos robots [15].

Atualmente é frequente associar-se o uso de manipuladores robóticos a operações de *pick-and-place*. Estas operações requerem que o robot manipule e movimente objetos num dado ambiente. Exemplos da aplicabilidade destes robots na área industrial incluem o empacotamento de medicamentos, sistemas de automação no manuseamento de produtos na indústria alimentar [16] e automatização de linhas de produção no ramo automóvel [17].

Dado que o objetivo deste projeto é definir e desenvolver uma metodologia capaz de planear trajetórias de manipuladores robóticos em ambientes industriais, as secções seguintes deste documento explicam as abordagens de *pick-and-place* já implementadas e a sua generalização (Secção 2.1), bem como o tipo de manipuladores existentes (Secção 2.2), as formas de reconhecimento do ambiente do robot (Secção 2.3) e, por fim, os algoritmos de planeamento de trajetória já implementados (Secção 2.4).

2.1 Operações de *pick-and-place*

Ao longo dos anos, várias têm sido as técnicas desenvolvidas com foco nas operações de *pick-and-place* e na sua otimização. Por operação de *pick-and-place* entende-se operações que resultam na movimentação de um objeto por intermédio da atuação de um manipulador robótico. Esta

movimentação pode representar um simples processo de transporte, pode representar assemblagem de peças ou ainda o deslocamento de peças ao longo de uma linha de produção.

Para clarificar e simplificar a operação, em 2000, Mattone e colaboradores descreveram os dois problemas fundamentais inerentes às técnicas de *pick-and-place*, separando-os na forma mais eficiente [18, 19]:

1. Sensorização, deteção e classificação dos objetos a serem manipulados;
2. *Gripping* (o nome advém da palavra *gripper* que diz respeito ao instrumento capaz de recolher as peças por parte do robot, normalmente trata-se de uma garra).

Em 2005, Gecks e Henrich desenvolveram um algoritmo para realizar as operações de *pick-and-place* de forma segura e em tempo real, usando processamento de imagem de várias câmaras estacionárias [20]. Em 2014, Daoud e colaboradores desenvolveram um método que permitia a cooperação entre robots, com cada robot realizar a operação de *pick-and-place* em tempo real de forma a maximizar o ritmo de produção [21]. Esforços para melhorar o *gripping* também têm sido realizados por diversos investigadores. Por exemplo, Wang e colaboradores propuseram uma estratégia dividida em três passos [22]:

1. Alinhamento do *gripper* com a peça no plano da imagem capturada por um sistema de câmara de vídeo;
2. Alinhamento do *gripper* com a peça na direção normal ao plano da imagem;
3. Fecho do *gripper* até a peça estar segura pelo mesmo. Embora a estratégia de Wang e colaboradores tenha sido realizada em micropeças, esta pode ser generalizada para todas as operações.

Para além dos dois passos principais do processo de *pick-and-place* (1. Sensorização, deteção e classificação dos objetos a serem manipulados; e 2. *Gripping*) é ainda necessária a implementação de um controlo de toda a operação. Em 2002, Son desenvolveu um algoritmo neuronal que juntamente com um modelo do processo fuzzy permitia a realização das tarefas de inserção de partes e transporte de partes, num ambiente que continha obstáculos. A ideia por trás desta metodologia é que ligando a informação do algoritmo neuronal à informação obtida através dos sistemas de pressão e de visão associados ao sistema, as condições de assemblagem onde a parte se inseria pudessem ser identificadas. Son mostrou, com esta informação, que a tarefa de assemblagem é conseguida usando um controlo fuzzy baseado no modelo previamente referido, que controlava os diversos atuadores do sistema, evitando situações de erro durante a mesma assemblagem [23, 24].

Todas as soluções e desenvolvimentos discutidos anteriormente foram propostos para responder a processos simples de *pick-and-place*. Contudo, existem ainda operações especiais de *pick-and-place*, onde se adiciona processos para além da identificação da peça e do *gripping* da mesma até ao destino pretendido. Exemplos mais complexos incluem os problemas de *bin-picking* (operações de *pick-and-place*, em que os objetos encontram-se misturados, entrelaçados e/ou desorganizados). Para este problema, Ghita e os seus colaboradores desenvolveram um sensor rápido

e preciso baseado em profundidade para uma aplicação de *bin-picking*. Este sensor juntamente com um sistema de visão de três componentes é capaz de reconhecer e avaliar o posicionamento dos objetos 3D [25]. A primeira componente faz uma segmentação da cena, seguindo um método baseado no número de cantos dos objetos, a segunda componente tenta reconhecer o objeto com base num modelo guardado em base de dados e finalmente as características do objeto reconhecido são avaliadas usando uma abordagem de reconhecimento de imagem baseada em vetores próprios [25].

2.2 Tipo de Manipuladores

A evolução desde o primeiro manipulador até à atualidade é exponencial. Nos anos mais recentes, o crescimento de inovações e pontos de interessa na área tem sido particularmente notório. Presentemente estes manipuladores robóticos são classificados e distinguidos pela sua geometria construtiva, pelo que existem dois grandes tipos de manipuladores: em série e em paralelo. A arquitetura em série distingue-se pela sucessão de corpos rígidos desde da base do robot até ao ponto de ferramenta (entenda-se por este ponto de ferramenta, o atuador que permita a manipulação dos objetos). A arquitetura em paralelo consiste num robot composto por uma ferramenta com vários graus de liberdade e uma base fixa, ligado entre si por cadeias cinemáticas independentes. Na arquitetura paralela, o robot é controlado por atuadores em igual ordem de grandeza que os seus graus de liberdade. A vantagem deste tipo de arquitetura paralela reside na elevada velocidade e aceleração, precisão, rigidez e reduzida inércia, quando comparada com a arquitetura em série [26].

Vários são os manipuladores que seguem a arquitetura em paralelo, por exemplo:

- Robot Delta [27];
- Robot Par4 [28];
- DexTAR [29];
- Sketchy [30];

Apesar das vantagens associadas aos robots com arquitetura em paralelo, os robots com arquitetura em série são adequados e suficientes para grande parte das operações de *pick-and-place* que são pertinentes a este projeto. Assim e atendendo ao custo inferior dos robots em série comparado com os robots em paralelo, este projeto utilizará robots com configuração em série. Abaixo apresentam-se as principais características dos robots com arquitetura em série. Exemplos deste tipo de arquitetura são os manipuladores antropomórficos (Figura 2.1), esféricos (Figura 2.2), SCARA (Figura 2.3), cilíndricos (Figura 2.4) ou cartesianos (Figura 2.5).

Existem ainda robots móveis que se podem considerar para manipuladores industriais, basta terem uma base móvel, por exemplo o robot TAPAS [33]. No entanto, em ambiente industrial e para cooperação com outros manipuladores e/ou operadores humanos, os manipuladores móveis

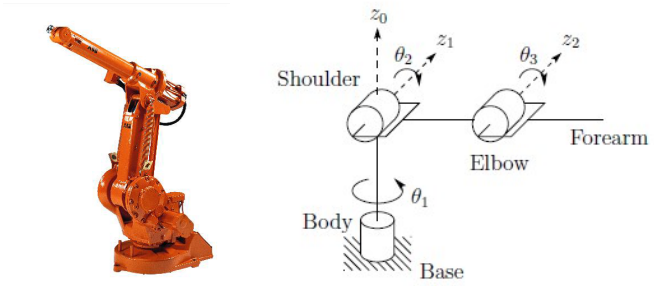

Figura 2.1: Robot antropomórfico (ABB IRB1400) [31]

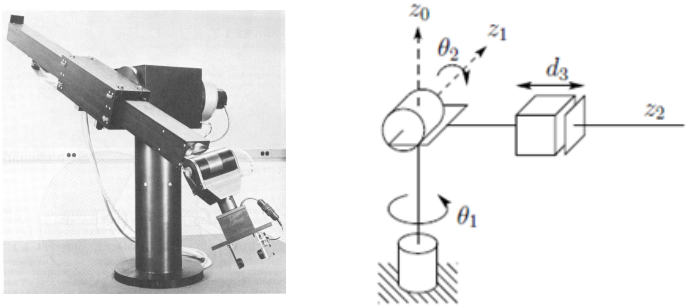

Figura 2.2: Robot esférico (The Stanford Arm) [31]

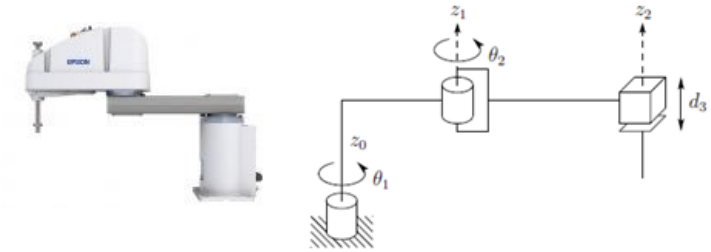

Figura 2.3: Robot SCARA (Epson G20) [31, 32]

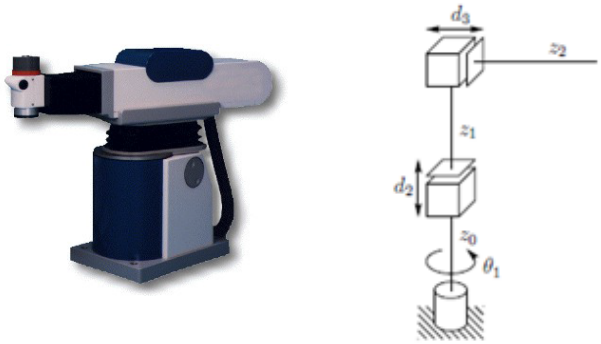

Figura 2.4: Robot Cilindrico (Seiko RT3300) [31]

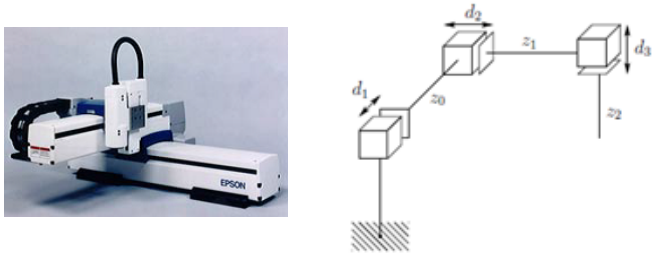

Figura 2.5: Robot Cartesiano (Epson Cartesian Robot) [31]

são menos fiáveis que os manipuladores anteriormente apresentados, principalmente pela questão da precisão, que em manipuladores fixos é superior. Desta forma, o desenvolvimento sobre manipuladores móveis não é de extrema relevância.

É importante também salientar a evolução constante das ferramentas associadas a todos os manipuladores [34]. Grande parte dos robots atuavam por intermédio de um punho robótico e/ou simplesmente por abertura e fecho de garras (figura 2.6). Contudo, e como já referido anteriormente, problemas como *bin-picking* têm conduzido ao desenvolvimento de novas ferramentas, mais flexíveis e eficazes a manipular certos objetos, nomeadamente o aparecimento do VERSABALL, um *gripper* que funciona por sucção de ar e que consegue assim oferecer maior diversidade e possibilidades na manipulação de objetos (figura 2.7).

Figura 2.6: Exemplos de mãos robóticas [35]

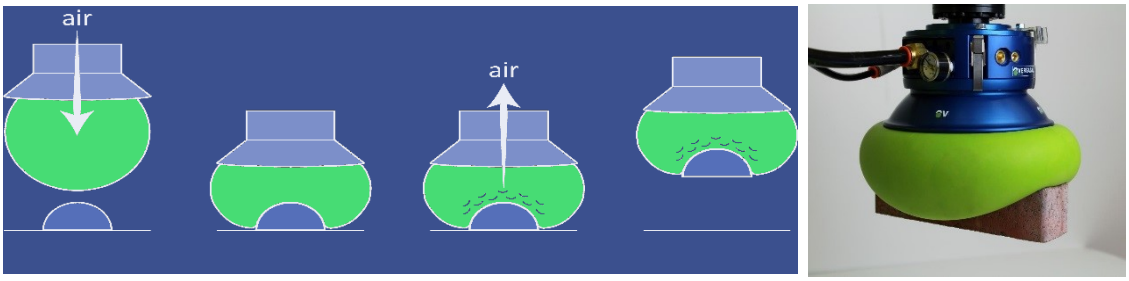

Figura 2.7: VERSABALL - Uma nova forma de Grasping [36]

2.3 Formas de Reconhecimento do Ambiente

Todas as aplicações que utilizem operações de *pick-and-place* têm, naturalmente, associadas um ambiente envolvente bem como localizações de objetos e zonas de destinos diferentes. A necessidade de reconhecer todo este ambiente conduz a um primeiro passo em qualquer abordagem a este tipo de operações que passa pela criação de um modelo do mapa onde o robot se localiza, bem como os objetos, obstáculos e outros corpos presentes nesse mapa.

Existem diversas formas de reconhecimento de posição ou ambiente, por exemplo *encoders*, balizas ou marcos e respetivos equipamentos de interpretação, ou sonares. No entanto, em aplicações industriais podem ser distinguidas três formas de localização com mais preponderância:

- Triangulação câmara-laser;
- Sistema de múltiplas câmaras de vídeo;
- Sistema RGBD (*Red, Green, Blue e Depth* – sistema que alia as capacidades das câmaras normais a um sistema de determinação de profundidade).

Todos estes sistemas têm a vantagem de além de identificarem e localizarem o robot, conseguirem recriar modelos 3D de todo o ambiente envolvente.

2.3.1 Triangulação Câmara-Laser

Sistemas de triangulação câmara-laser são um dos métodos mais comummente utilizados em robots industriais, principalmente devido à precisão que estes sistemas conseguem oferecer assim como à sua elevada velocidade computacional.

Podem-se considerar duas abordagens para este processo:

1. a câmara e o laser estão fixos numa calha linear, onde a câmara e o laser são usados como uma solução compacta que necessita apenas de uma única calibração (Figura 2.8);
2. a câmara está estática e apenas o laser se desloca ao longo da calha (Figura 2.9).

A abordagem *2* não é vantajosa uma vez que requer a calibração do plano do laser a cada *frame* da câmara, contudo é mecanicamente mais simples e económica.

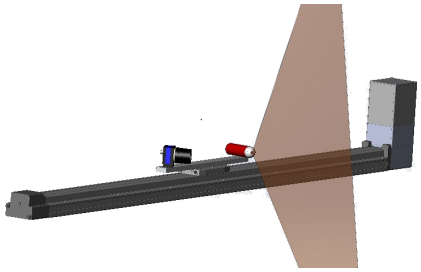

Figura 2.8: Abordagem 1 – Câmara e Laser Fixos numa calha linear [37]

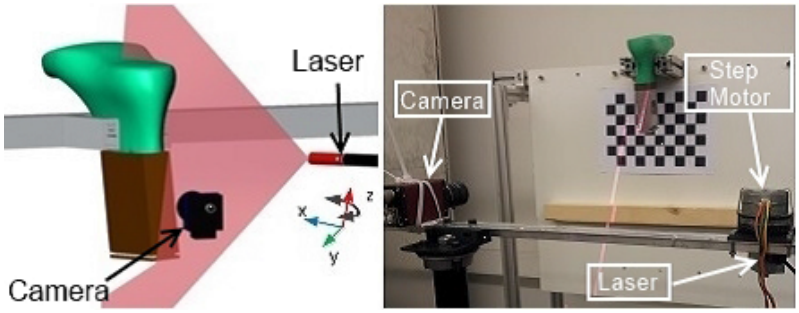

Figura 2.9: Abordagem 2 - Câmara Fixa e Laser Móvel [37]

2.3.2 Sistemas de múltiplas câmaras de vídeo

A necessidade e as vantagens associadas à criação de um modelo 3D do ambiente envolvente ao robot levaram ao desenvolvimento de algoritmos e aperfeiçoamentos matemáticos de certas expressões de dedução. Os sistemas de múltiplas câmaras de vídeo, além da capacidade de observar vários ângulos do ambiente, permitiram por meio de expressões matemáticas deduzir várias propriedades dos objetos e do meio onde o robot se insere. Em 2004, Barreto e os seus colaboradores abordaram a dificuldade em reconhecer objetos e as suas propriedades, tendo proposto uma solução de múltiplas câmaras para resolução destas questões [38]. Basicamente o cruzamento da informação entre as várias câmaras, bem como as expressões matemáticas apresentadas ao longo do artigo de Barreto e colaboradores, permitiram recriar o desejado modelo 3D. Uma revisão do método e um sumário da informação relativo a esta abordagem encontra-se no trabalho de Sturm e seus colaboradores, publicado em 2011 [39].

2.3.3 Sistemas RGBD

Os sistemas apresentados anteriormente (secções 2.3.1 e 2.3.2) baseavam-se no processamento da imagem adquirida, de modo a estimar a profundidade de cada ponto da dita imagem no mundo real e a partir daí, em conjunto com outros algoritmos de processamento, determinar, de uma forma precisa, o modelo 3D do ambiente. Os sistemas RGBD tem a vantagem de não ser necessário estimar a profundidade por meio de um algoritmo, visto que acoplado ao sistema de vídeo RGB está um sistema capaz de determinar a profundidade em cada segmento da imagem captada ao longo do tempo. Isto permite avançar o passo da estimação de profundidade e obter resultados de forma muito mais rápida, precisa e eficiente. Outra vantagem é o facto de ser uma solução mais económica que os métodos expostos anteriormente. Por exemplo uma solução de câmara com um laser *range-finder* (ou seja, uma solução câmara-laser capaz de determinar o modelo 3D) pode custar perto dos 10.000€ enquanto uma simples câmara Kinect (câmara RGBD associada à empresa Microsoft e à marca XBOX, Figura 2.10) consegue-se a um preço bastante mais acessível, por volta dos 150€. Por estes motivos são já vários os estudos e projetos baseados na tecnologia RGBD. Recentemente, a Intel apresentou um programa cujo objetivo é desenvolver metodologias

para potenciar o uso deste sistema [40]. Por outro lado, vários estudos de investigação na comunidade robótica têm sido reportados e têm como objetivo simplificar soluções já implementadas recorrendo a estes sistemas [41, 42, 43].

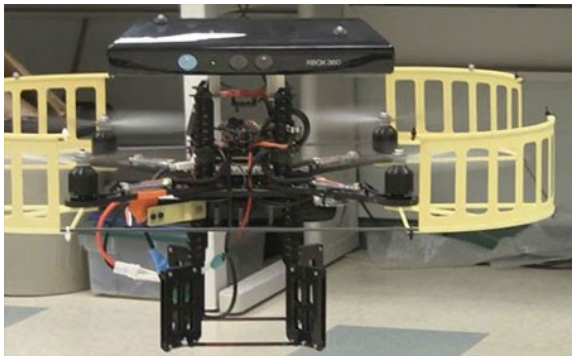

Figura 2.10: Robot equipado com o Kinect [44]

2.4 Pesquisa e Planeamento de Trajetória

A partir do ambiente que envolve o robot é possível definir qual a trajetória que deve ser cumprida pelo mesmo para se movimentar entre dois pontos. Para este efeito existem várias alternativas. Por exemplo, pode-se optar por algoritmos não sofisticados e na sua essência reativos, no sentido em que apenas respondem a estímulos, sem ter certeza total de qual a trajetória a adotar [45]. No entanto, em ambiente industrial faz todo o sentido predeterminar a trajetória antes de movimentar o robot, uma vez que se tratam geralmente de ambientes não estruturados, com variações de cores, forma, entre outros, ao longo de toda a cena. De forma a cumprir este objetivo é importante considerar algoritmos completos. Assim, torna-se necessário, que o robot seja capaz de primeiro fazer o levantamento das configurações que este pode assumir, para depois se aplicar um método de definir quais as configurações que o robot deve assumir nas posições de destino, e finalmente definir-se a trajetória que o robot deve realizar por intermédio de algoritmos já testados e comprovados (A*, Dijkstra, entre outros).

Relativamente ao espaço de configurações, Yong e colaboradores em 1992 enumeraram as diversas formas de determinar qual o espaço de configurações que o robot pode ter, ou seja, as posições que o robot pode assumir ao longo da sua trajetória [46]. Cada técnica apresentada pretende descobrir quais as configurações de cada junta do robot, criando assim uma área de trabalho inerente ao robot, e por aplicação das diversas técnicas, consegue-se extrair a área onde se insere obstáculos e na qual o robot não se pode movimentar. Uma vez definida qual a área acessível ao robot, deve-se determinar qual a configuração no destino pretendido. Ao considerar mapas sem obstáculos a dificuldade deve ser reduzida simplesmente à configuração mais rápida de alcançar o destino final. Contudo quando se inserem obstáculos no caminho surge a necessidade de se utilizar métodos capazes de os contornar. Latombe em 1990 definiu que existem essencialmente três métodos para contornar obstáculos: (1) métodos com base na divisão da cena em células, de forma

a serem criados os caminhos possíveis entre duas quaisquer células (a que o robot se encontra e a do destino); (2) métodos com base na criação de campos potenciais, onde o destino funciona como polo atrativo e os obstáculos como polos repulsivos; e (3) métodos probabilísticos que consistem em determinar caminhos entre diversas células, baseando-se em caminhos já efetuados anteriormente, ou seja, funciona como uma forma de adaptação [47]. Todos estes métodos conduzem à produção de grafos ou de elementos que permitem então aos algoritmos de planeamento de trajetória definir qual o trajeto a seguir.

Recentemente têm sido utilizados dois algoritmos em concreto para o planeamento de trajetórias em manipuladores: *Dynamic Roadmaps* e *Elastic Roadmaps*. *Roadmaps* é uma representação do espaço livre do robot sob a forma de uma rede de curvas uni-dimensionais que permite reduzir o planeamento de trajetórias a um problema de ligações entre as configurações iniciais e finais do mesmo espaço. A variante *Dynamic Roadmaps* assenta na ideia de ambiente que se altera ao longo do tempo. Para estes casos, a noção de espaço de configuração é substituída pelo binómio espaço de configuração – espaço no qual é adicionada uma nova dimensão, a dimensão temporal [48]. A variante *Elastic Roadmaps* baseia-se em dois princípios:

1. A *framework* do *Elastic Roadmaps*, o papel do planeador é determinar uma aproximação do estado da conetividade do espaço livre, sendo esta conetividade usada para gerar um sistema híbrido com controladores de especificação de tarefas, sendo estes controladores os responsáveis pelo movimento do robot;

2. Com a convulsão entre a conetividade do espaço de configuração e a estrutura cinemática do robot permite descrever e identificar os controladores híbridos mais eficientes [49].

Aplicando uma destas variantes, é possível decompor o espaço envolvente ao robot em pontos de conetividade acelerando o futuro processo de pesquisa para finalização do planeamento de trajetória.

2.4.1 Algoritmos de Pesquisa para Planeamento de Trajetória

Por algoritmo de planeamento de trajetória entende-se o conjunto de instruções que rege a movimentação do robot entre dois pontos. Os métodos anteriormente referidos nem sempre criam a totalidade do percurso, mas apenas os pontos que se podem obter ao longo do caminho. Nomeadamente os métodos probabilísticos e os métodos de divisão por células geram apenas grafos com o espaço livre de configuração. Assim, é necessário um método de pesquisa de grafos.

Tipicamente, o objetivo destas pesquisas e dos algoritmos apresentado de seguida, aproximam-se do problema do caminho mínimo. As soluções mais diretas de adotar baseiam-se na iteração de movimento, entre o ponto inicial (entenda-se configuração inicial) até se atingir a configuração final. Exemplos disso são os algoritmos *Breadth-First* e *Depth-First Search* (BFS e DFS) [50].

Pelo facto de estes algoritmos não conhecerem o grafo no início da execução, classificam-se como algoritmos sem informação. Contudo, por procurar todas as alternativas, o tempo de execução e a memória exigida para o seu processamento está longe de ser ideal.

Um algoritmo importante de referir nesta fase é o algoritmo Dijkstra. Dijkstra é um algoritmo que considera o custo associado a cada movimento do robot. Embora seja semelhante ao BFS, permite acelerar o processo na medida que estima o qual a decisão ótima em cada momento, motivo pelo qual tem mais aplicação em robótica [51]. Para tal o algoritmo, designa um nó inicial (configuração inicial) e a cada iteração determina um nó u, que minimize a distância do nó inicial a u. Iterativamente, também, avalia todas as adjacências v do vértice de configuração u em que está e verifica-se se a distância entre o nó inicial e qualquer das adjacências v é menor que a distância anterior. Em caso afirmativo o novo nó ótimo é atualizado, passando a ser um nó v, caso contrário permanece como sendo u. O algoritmo acaba quando analisar todos os nós possíveis no grafo. Surgiram já outras formas de abordar a implementação do algoritmo que o adaptam de forma a proteger situações críticas de processamento ao longo da execução do algoritmo [52]. A vantagem em relação aos anteriores passa precisamente pela velocidade com que pode ser executado, contudo por ter de percorrer todos os nós do grafo, também não é ideal.

Aqui se insere a vantagem de algoritmos com heurística (regra), nos quais já se conhece o grafo e por isso se designam como algoritmos com informação, e nos quais se utiliza a heurística para estimar qual a melhor trajetória a utilizar, por exemplo o algoritmo A*.

O algoritmo A* é bastante semelhante ao algoritmo Djisktra, no qual se insere uma heurística de custo. Este algoritmo usa um método de exploração de grafos com uma dada regra. Com o objetivo de escolher qual o melhor trajeto a seguir, usa uma função de distância ajustada com o custo de deslocamento entre o ponto de partida e o ponto que se pretende atingir. Com esta modificação e o facto de neste algoritmo existir um conhecimento prévio do nó de destino, a escolha a cada iteração do nó a seguir é feita com base na distância entre esse nó e o nó de destino.

Capítulo 3

Espaço de Configurações e Cinemática

Neste capítulo serão abordadas as temáticas que envolvem o espaço envolvente ao robot e a sua cinemática, explicando quais as equações que regem o movimento do mesmo robot e como dividir e analisar o espaço que o envolve.

Tipicamente para determinar corretamente o trajeto a seguir deve-se compreender as consequências de cada passo desse trajeto. Adicionalmente os métodos de reconhecimento do ambiente consegue definir as coordenadas, em espaço cartesiano, de destino e origem, contudo o robot é controlado em espaço de articulações. Assim, definir um espaço de configurações onde se distinga inequivocamente os trajetos livres por onde o robot pode passar permite definir um trajeto viável para o robot, associando as posições cartesianas a posições de articulações. Outro aspeto prende-se com a cinemática de qualquer manipulador que permite determinar quais as poses adquiridas pelo robot e descobrir se poses futuras se inserem no espaço pretendido.

3.1 Espaço de Configurações

A necessidade de discretizar o ambiente envolvente ao robot conduz à necessidade de criar um espaço de configurações. Para um manipulador robótico, um espaço de configurações q, pode ser definido como um conjunto de valores que define a posição atual do robot. Estes valores são tipicamente associados aos valores das articulações do mesmo manipulador. Assim facilmente se passa de um problema de três dimensões redundante para um problema de N dimensões (o número de dimensões do manipulador robótico) mas bem definido, onde cada estado de configuração corresponde a um único ponto no espaço global do robot.

Tendo em conta o espaço de trabalho do robot, pode-se definir a dimensão total do espaço de configuração, ao qual se denominará C. C pode-se dividir em C_{livre} ou $C_{ocupado}$, sendo o primeiro as configurações que o robot pode assumir e o segundo as configurações que se desejam limitar ou onde se insere obstáculos.

Por motivos de simplificação da complexidade a discretização do sistema é feita por forma a reduzir o número de posições possíveis para o robot. Embora seja inserido erro no sistema

globalmente, o mesmo erro pode ser corrigido posteriormente com algoritmos de correção e o tempo de execução do projeto é consideravelmente reduzido.

Várias são as técnicas de criação deste espaço de configuração sendo que maioritariamente funcionam como técnicas de *matching* entre a posição do robot e a posição de obstáculos.

3.2 Cinemática Direta

Cinemática direta é o processo de determinar qual a posição do robot atendendo ao estado das suas articulações. Os manipuladores para os quais este projeto foi direcionado eram constituídos por articulações rotacionais, de forma que o parâmetro variável são as posições angulares dos mesmos. Para conseguir determinar esta posição do manipulador o primeiro passo é definir sistemas de coordenadas associadas a cada um dos segmentos que ligam duas articulações. Com base nesses sistemas de coordenadas consegue-se encontrar relações entre os diversos segmentos do robot por meio de uma rotação e uma translação [53, 54, 55]. Tipicamente entre um referencial x e outro y, pode-se definir então uma matriz de transformação linear definida como:

$$T_y^x = \begin{bmatrix} R_y^x & \vec{P}_y^x \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

R_y^x é a matriz que define a rotação entre os referenciais x e y, sendo $\vec{P}_y^x$, o vetor de translação entre os mesmo referenciais. Mais ainda, a transformação total entre base e ponto final da ferramenta pode ser expresso como um produtório destas transformações entre referenciais, ou seja:

$$T_N^0 = T_1^0 * T_2^1 * T_3^2 * ... * T_N^{N-1} \tag{3.2}$$

Para conseguir definir cada uma destas transformações pode-se optar pelo método DH proposto por Denavit e Hartenberg [56]. As transformações homogéneas que relacionam os referenciais podem ser representadas pelos parâmetros DH, os quais se definem da seguinte forma:

- a_i - distância entre o eixo i-1 e a origem do referencial i, o_i;
- α_i - ângulo formado entre o eixo i-1 e o eixo i no plano normal a x_i;
- d_i - distância entre a origem do referencial i-1, o_{i-1} e o eixo x_i;
- q_i - ângulo formado entre o eixo x_{i-1} e o eixo x_i no plano normal a z_{i-1}.

Baseando-se nestes valores é possível então determinar as matrizes de transformação homogénea de todos os referenciais que são definidas genericamente como:

$$\begin{bmatrix} cos(\theta_i) & -sin(\theta_i)*cos(\alpha_i) & sin(\theta_i)*cos(\alpha_i) & a_i*cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)*cos(\alpha_i) & -cos(\theta_i)*cos(\alpha_i) & a_i*sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

3.3 Cinemática Inversa

A cinemática inversa de um robot permite identificar o estado de cada articulação para uma qualquer posição e orientação espacial. Simultaneamente, a aplicação de cinemática inversa permite a transformação do espaço cartesiano para o espaço de juntas do robot. Maioritariamente os manipuladores são controlados por instruções no espaço de juntas, ou seja, é definido a posição das articulações e não a posição do ponto final do mesmo. Assim, a aplicação da cinemática inversa é obrigatória para o controlo de movimentação.

3.4 O robot UR5

Para efeito de testes do algoritmo foi escolhido o robot Universal 5 (UR5). O UR5 é um robot dedicado ao trabalho cooperativo com operários humanos. Adicionalmente é um robot flexível e facilmente adaptável a qualquer tipo de aplicação. Este robot tem seis articulações (um braço robótico tradicional dotado com um punho esférico, Figura 3.1), todas elas rotacionais.

Figura 3.1: Robot Universal 5 (UR5)

O primeiro passo foi determinar as equações que regem a cinemática deste robot. Para tal elaborou-se uma tabela com informação das articulações segundo o método DH (ver Tabela 3.1 e figura 3.2).

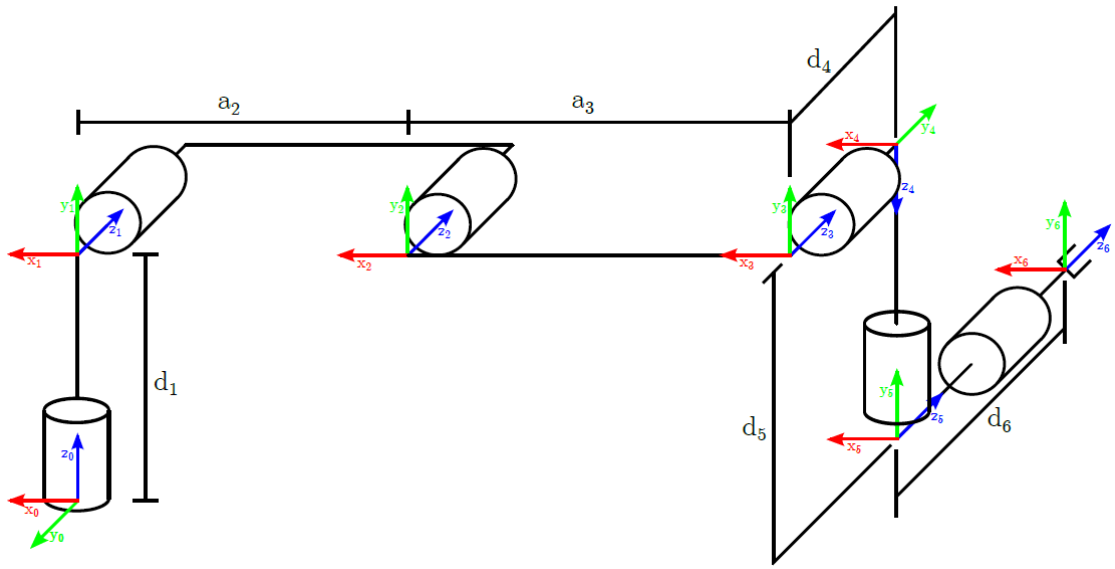

Figura 3.2: Referenciais para cinemática do UR5 [57]

Tabela 3.1: Parâmetros resultantes do método DH no UR5

Articulação	a_i	α_i	d_i	θ_i
1	0	π / 2	d_1	θ_1
2	-a_2	0	0	θ_2
3	-a_3	0	0	θ_3
4	0	π/2	d_4	θ_4
5	0	-π/2	d_5	θ_5
6	0	0	d_6	θ_6

O primeiro passo da determinação da cinemática inversa é determinar o valor de θ_1. Para tal é necessário considerar a 5ª articulação do robot e a sua posição comparada com a origem do robot (P_5^0). Assim surge a necessidade de translação da articulação 6 para a articulação 5. Esta translação ocorre segundo o eixo z e é de valor -d_6. Tendo em conta as equações 3.2 e 3.3 tem-se:

$$P_5^0 = T_6^0 * \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.4}$$

Considerando a vista superior sobre o robot tem-se:

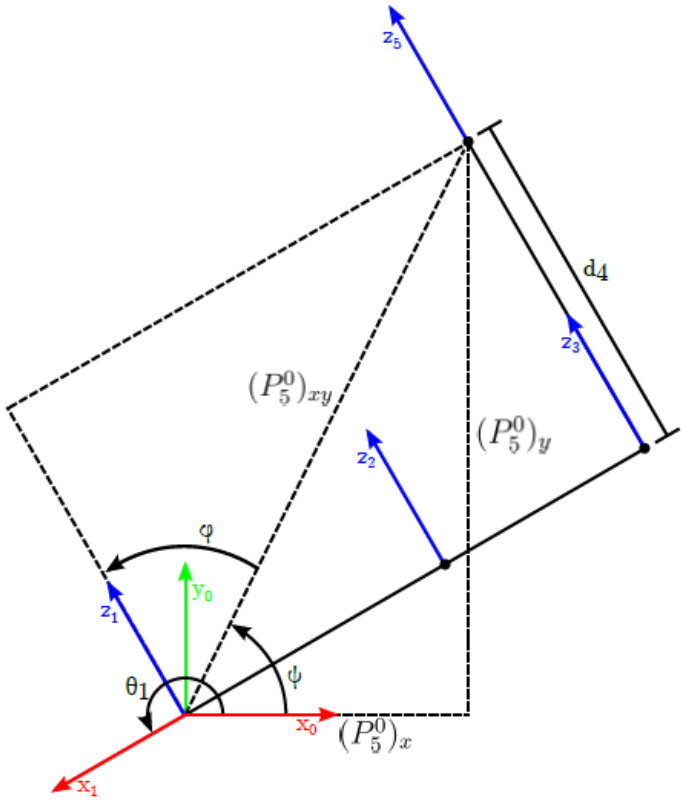

Figura 3.3: Vista superior sobre o robot UR5 [62]

Tendo-se determinada a posição da articulação 5, (P_5^0), tem-se definido o valor de θ_1 como:

$$\theta_1 = \psi + \phi + \frac{\pi}{2} \tag{3.5}$$

$$\psi = atan2((P_5^0)_y, (P_5^0)_x) \tag{3.6}$$

$$\phi = \pm arccos\left(\frac{d_4}{\sqrt{(P_5^0)_x^2 + (P_5^0)_y^2}}\right) \tag{3.7}$$

Calculado θ_1 consegue-se determinar θ_5. Recorrendo novamente à vista superior, incluindo desta vez a articulação 6, consegue-se expressar o ponto resultante da transformação (P_6^0) em função de θ_5 (ver figura 3.4).

$$(P_5^0)_x * sin(\theta_1) - (P_5^0)_y * cos(\theta_1) = d_6 * cos(\theta_5) + d_4 \tag{3.8}$$

$$\theta_5 = \pm arccos\left(\frac{(P_5^0)_x * sin(\theta_1) - (P_5^0)_y * cos(\theta_1) - d_4}{d_6}\right) \tag{3.9}$$

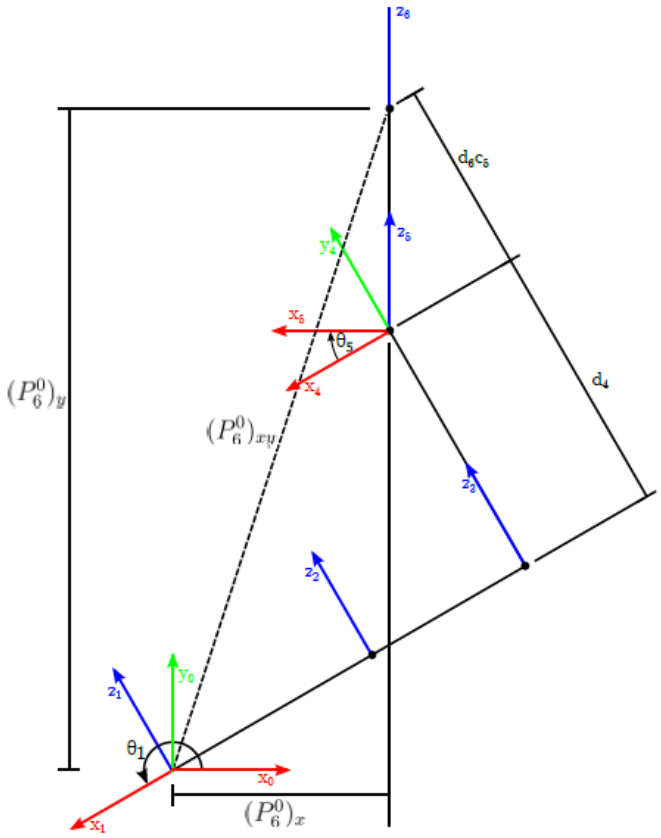

Figura 3.4: Vista superior total do UR5 [57]

Recorrendo à equação 3.2 podemos definir uma transformação entre as articulações 1 e 6 (T_6^1) como o produto entre $(T_1^0)^{-1}$ e T_6^0. Recordando também a equação 3.3 pode-se relacionar a terceira coluna da matriz que expressa a transformação entre as articulações 1 e 6 com os valores de θ_5 e θ_6:

$$-sin(\theta_6) * sin(\theta_5) = (T_6^1)_{1,2} \tag{3.10}$$

$$cos(\theta_6) * sin(\theta_5) = (T_6^1)_{0,2} \tag{3.11}$$

$$\theta_6 = atan2\left(\frac{-(T_6^1)_{1,2}}{sin(\theta_5)}, \frac{(T_6^1)_{0,2}}{sin(\theta_5)}\right) \tag{3.12}$$

Calculando estas articulações, pode-se utilizar o princípio do desacoplamento para redefinir o manipulador como um manipulador planar com duas articulações rotacionais (Figura 3.5).

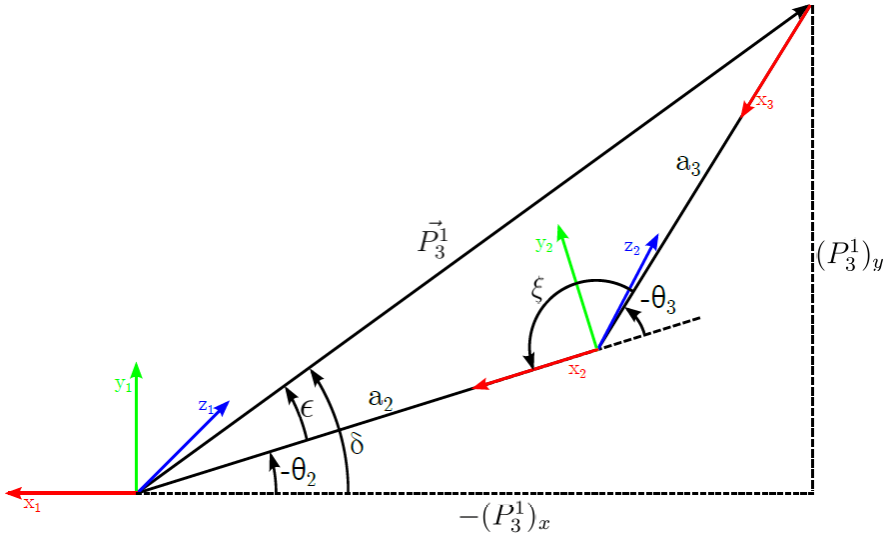

Figura 3.5: Decomposição das articulações 2 e 3 [57]

Analisando a figura 3.5, consegue-se definir θ_3 em função da posição P_3^1 e dos valores de a_2 e a_3:

$$cos(\xi) = -cos(\pi - \xi) = -cos(-\theta_3) = cos(\theta_3) \tag{3.13}$$

$$cos(\xi) = \frac{\left\|P_3^1\right\|^2 - a_2^2 - a_3^2}{2 * a_2 * a_3} \tag{3.14}$$

$$\theta_3 = arccos\left(\frac{\left\|P_3^1\right\|^2 - a_2^2 - a_3^2}{2 * a_2 * a_3}\right) \tag{3.15}$$

Pela mesma figura determina-se θ_2. Este ângulo pode ser definido como a negação da subtração de dois ângulos (δ e ε):

$$\delta = atan2((P_3^1)_y, (P_3^1)_x) \tag{3.16}$$

Pela lei dos senos tem-se:

$$\frac{sin(\xi)}{\left\|P_3^1\right\|} = \frac{sin(\varepsilon)}{a_3} \tag{3.17}$$

$$\varepsilon = arcsin\left(\frac{sin(\xi) * a_3}{\left\|P_3^1\right\|}\right) \tag{3.18}$$

$$\theta_2 = -(\delta - \varepsilon) \tag{3.19}$$

Conhecidas todas as articulações excetuando a 4, é fácil de determinar todas as transformações entre articulações T_j^i, extraindo-se assim a transformação T_4^3. Baseado, novamente, na equação 3.2 tem-se:

$$T_4^3 = T_1^3 * T_4^1 = (T_2^1 * T_3^2)^{-1} * T_1^6 * (T_5^4 * T_6^5)^{-1} \tag{3.20}$$

Isolando-se a primeira coluna da matriz de transformação consegue-se identificar θ_4:

$$\theta_4 = atan2((T_4^3)_{1,0}, (T_4^3)_{0,0}) \tag{3.21}$$

Por outro lado, a cinemática direta surge também como o resultado inverso desta dedução, sendo que é possível preencher a matriz de transformação entre base e origem com os seguintes valores:

Tabela 3.2: Designações da Matriz de Transformação Homogénea

Matriz de Transformação	$Coluna_1$	$Coluna_2$	$Coluna_3$	$Coluna_4$
$Linha_1$	x_x	y_x	z_x	$(P_0^6)_x$
$Linha_2$	x_y	y_y	z_y	$(P_0^6)_y$
$Linha_3$	x_z	y_z	z_z	$(P_0^6)_z$
$Linha_4$	0	0	0	1

Por questões de simplificação, abordar-se-á termos de $cos(\theta_i)$ como c_i e $sin(\theta_i)$ como s_i.

$x_x = c_6 * (s_1 * s_5 + ((c_1 * c_{234} - s_1 * s_{234}) * c_5)/2 + ((c_1 * c_{234} + s_1 * s_{234}) * c_5)/2) - (s_6 * ((s_1 * c_{234} + c_1 * s_{234}) - (s_1 * c_{234} - c_1 * s_{234})))/2;$

$x_y = c_6 * ((s_1 * c_{234} + c_1 * s_{234}) * c_5)/2 - c_1 * s_5 + ((s_1 * c_{234} - c_1 * s_{234}) * c_5)/2) + s_6 * ((c_1 * c_{234} - s_1 * s_{234})/2 - (c_1 * c_{234} + s_1 * s_{234})/2);$

$x_z = (s_{234} * c_6 + c_{234} * s_6)/2 + s_{234} * c_5 * c_6 - (s_{234} * c_6 - c_{234} * s_6)/2;$

$y_x = -(c_6 * ((s_1 * c_{234} + c_1 * s_{234}) - (s_1 * c_{234} - c_1 * s_{234})))/2 - s_6 * (s_1 * s_5 + ((c_1 * c_{234} - s_1 * s_{234}) * c_5)/2 + ((c_1 * c_{234} + s_1 * s_{234}) * c_5)/2);$

$y_y = c_6 * ((c_1 * c_{234} - s_1 * s_{234})/2 - (c_1 * c_{234} + s_1 * s_{234})/2) - s_6 * (((s_1 * c_{234} + c_1 * s_{234}) * c_5)/2 - c_1 * s_5 + ((s_1 * c_{234} - c_1 * s_{234}) * c_5)/2);$

$y_z = (c_{234} * c_6 + s_{234} * s_6)/2 + (c_{234} * c_6 - s_{234} * s_6)/2 - s_{234} * c_5 * s_6;$

$z_x = c_5 * s_1 - ((c_1 * c_{234} - s_1 * s_{234}) * s_5)/2 - ((c_1 * c_{234} + s_1 * s_{234}) * s_5)/2;$

$z_y = -c_1 * c_5 - ((s_1 * c_{234} + c_1 * s_{234}) * s_5)/2 + ((c_1 * s_{234} - s_1 * c_{234}) * s_5)/2;$

$z_z = (c_{234} * c_5 - s_{234} * s_5)/2 - (c_{234} * c_5 + s_{234} * s_5)/2;$

$(P_0^6)_x = -(d_5 * (s_1 * c_{234} - c_1 * s_{234}))/2 + (d_5 * (s_1 * c_{234} + c_1 * s_{234}))/2 + d_4 * s_1 - (d_6 * (c_1 * c_{234} - s_1 * s_{234}) * s_5)/2 - (d_6 * (c_1 * c_{234} + s_1 * s_{234}) * s_5)/2 + a_2 * c_1 * c_2 + d_6 * c_5 * s_1 + a_3 * c_1 * c_2 * c_3 - a_3 * c_1 * s_2 * s_3)$;

$(P_0^6)_y = -(d_5 * (c_1 * c_{234} - s_1 * s_{234}))/2 + (d_5 * (c_1 * c_{234} + s_1 * s_{234}))/2 - d_4 * c_1 - (d_6 * (s_1 * c_{234} + c_1 * s_{234}) * s_5)/2 - (d_6 * (s_1 * c_{234} - c_1 * s_{234}) * s_5)/2 - d_6 * c_1 * c_5 + a_2 * c_2 * s_1 + a_3 * c_2 * c_3 * s_1 - a_3 * s_1 * s_2 * s_3)$;

$(P_0^6)_z = d_1 + (d_6 * (c_{234} * c_5 - s_{234} * s_5))/2 + a_3 * (s_2 * c_3 + c_2 * s_3) + a_2 * s_2 - (d_6 * (c_{234} * c_5 + s_{234} * s_5))/2 - d_5 * c_{234}$.

Finalmente com a análise da cinemática inversa torna-se evidentes várias soluções para uma mesma posição cartesiana e orientação. Elas correspondem às combinações entre ombro à esquerda e ombro à direita, cotovelo para cima e cotovelo para baixo e punho para cima e punho para baixo (Figura 3.6).

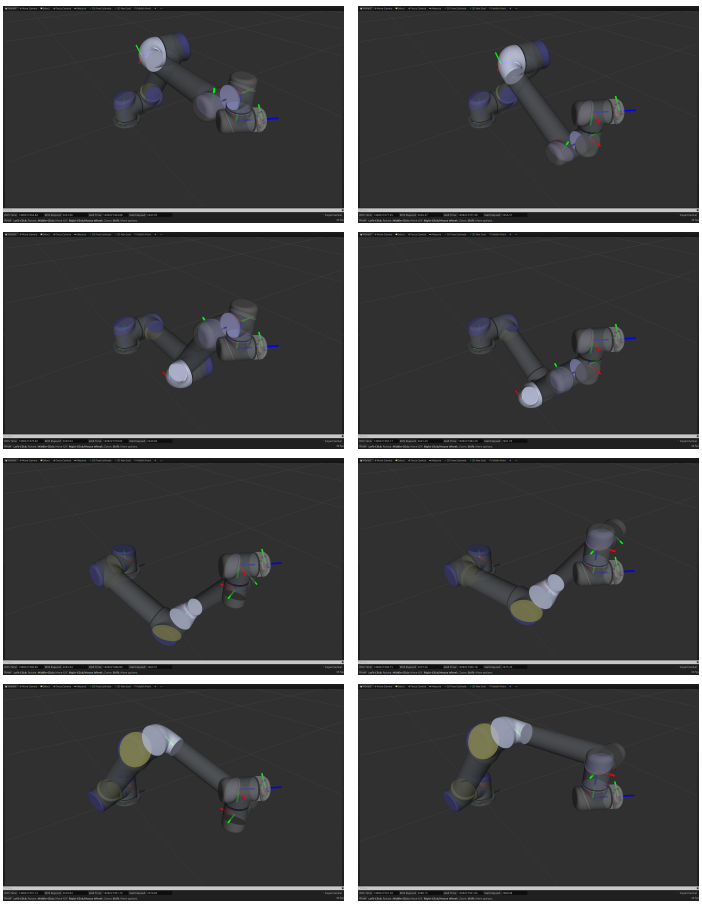

Figura 3.6: Várias configurações do robot UR5

3.5 Resumo do Capítulo

Neste capítulo foi descrita a importância de dois fundamentos das operações de *pick-and-place*: o espaço de configurações e a compreensão da cinemática. Mais ainda, foi particularizado o caso do robot utilizado na maioria das aplicações do projeto, o robot UR5.

O espaço de configurações permite a manipulação do espaço de trabalho do robot por meio de uma discretização. Esta discretização é o que torna possível a avaliação de cada célula do espaço como espaço livre para movimentação ou como espaço ocupado, ou seja, onde o robot não tem possibilidade de se movimentar.

O cálculo da cinemática incide sobre este espaço discretizado. Este cálculo associa o robot a cada célula desse espaço, percebendo também a pose do mesmo robot no mundo.

Capítulo 4

Operação Pick-and-Place: Arquitetura do Sistema e Desafios

Neste capítulo será apresentada a abordagem proposta para as operações de *pick-and-place* (secção 4.1), assim como casos de aplicação da mesma, neste caso competições de robótica (secção 4.2). Para este fim serão evidenciados os diferentes passos que contemplam tipicamente as operações de *pick-and-place* e explicado a razão pela qual as competições de robótica demonstram importância não só para o desenvolvimento da robótica mas também como destino de aplicação da metodologia proposta.

4.1 Arquitetura do Sistema

A operação de *pick-and-place* pode ser definida como uma tarefa complexa que requer vários patamares de controlo. A arquitetura do sistema implementada assenta sobre três níveis operacionais (ver Tabela 4.1).

Tabela 4.1: Níveis Operacionais da Operação de *Pick-and-Place*

Níveis Operacionais	Especificação do Nível	Exemplos de Aplicação
1	Reconhecimento da Cena	Processamento de Imagem; Triangulação Câmara-Laser; Sonar; Entre Outros.
2	Planeamento Estratégico e Movimentação	Planeamento de Trajetória; Planeamento de Tarefas; Heurísticas; Movimentação do Robot; Controlo das Articulações.
3	Atuação e Controlo	*Gripping* e *Grasping*; Correções de Erros; Outros.

De uma maneira geral, pode-se definir as operações de *pick-and-place* como uma metodologia com 3 passos principais. Esta metodologia permite a completa adaptação do robot a diferentes aplicações e diferentes cenários, desde logo pela sua capacidade de corrigir erros de movimento do robot, de reconhecimento do ambiente, da perceção da posição de objetos e obstáculos e pela segurança e certeza na altura do *gripping*. De certa forma, as operações de *pick-and-place* podem ser vistas como um ciclo de controlo, onde os erros são corrigidos continuamente com base na perceção do ambiente e do constante planeamento das trajetórias.

O objetivo desta arquitetura é o desenvolvimento de um método simples e universal para todas as operações de *pick-and-place* capaz de ser aplicado em diversos robots e diversos ambientes. A arquitetura proposta segue os níveis operacionais presentes na Tabela 4.1, o que permite o adicionamento de flexibilidade à solução final. Desta forma, o sistema implementado é capaz de ser autonomamente colocado em qualquer espaço, reconhecê-lo e atuar sobre o mesmo, enquanto se ajusta a erros que ocorram ao longo de todo o processo.

Assim, para o sistema ser capaz de detetar e manipular vários objetos surge a necessidade de desenvolver um algoritmo capaz de planear a sequência correta de ações que o robot deve executar de forma controlada. Este algoritmo deve incluir passos como reconhecimento do ambiente, calibração do robot, planeamento de trajetória e *grasping* (ver figura 4.1).

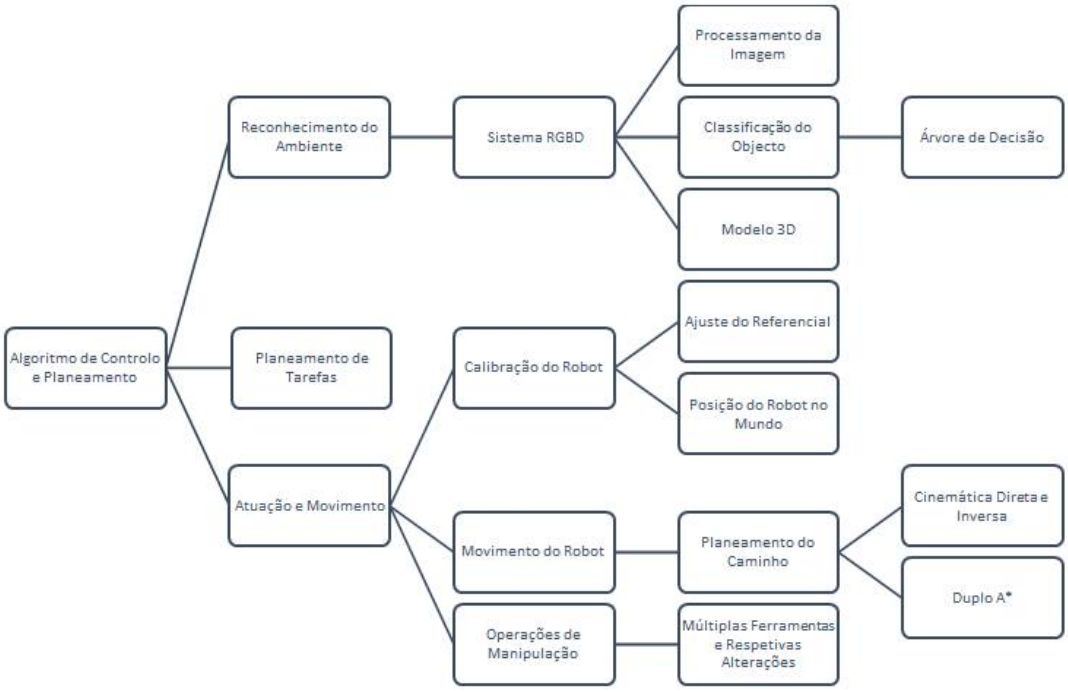

Figura 4.1: Esquema Ilustrativo das Tarefas do Algoritmo

Adicionalmente, a arquitetura do sistema assenta sobre a *framework* ROS (ver secção 4.1.4). A utilização de ROS permite dividir os níveis operacionais já referido de forma simples. Sucintamente, a implementação com a utilização de ROS consiste na criação de nós (programas

independentes) para cada um dos níveis. Cada um desses nós consegue comunicar com os restantes utilizando mensagens ou a política de publicador-subscritor. Outro fator consiste no facto de cada nó poder ser responsável pelo controlo de vários serviços, o que permite dentro de cada nível operacional lançar diversos processos, desde processamento de imagem ou criação do modelo 3D no primeiro nível, a definição de trajeto e movimentação do robot no segundo nível ou o controlo do planeamento no terceiro nível.

Essencialmente, utilizando a *framework* ROS e a abordagem proposta pretende-se implementar um algoritmo que consiga descobrir e reconhecer a cena, definir uma sequência de ações e controlar o processo desta sequência (ver algoritmo 1).

Algoritmo 1: Controlo da Operação de *Pick-and-Place*

```
Inputs: Interface_Robot + Modelo_3D ;
Outputs: end_effector_pose;

Tarefa = DescobrirTarefa();
Cena = Identificacao_cena(Modelo_3D);
Obj = Detecao_Objetos(Modelo_3D);
Sequencia = Descobrir_Sequencia(Tarefa, Cena, Obj);
Ac_Tree = Criar_Arvore_Accao(Sequencia);

while ( Numero_De(Ac_Tree.Ramos) > 0 ) do
    for (NumeroTentativas) do
        ExecutarSequencia(Ac_Tree.Ramo[i]);
        OK = Avaliar_Placing(Ac_Tree.Ramo[i], Modelo_3D);

        if (OK) then
            RemoveRamo(Obj[i]);
            Break;
            else
                Atualizar_Sequencia_Cena(Ac_Tree.Branch[i], Tarefa, Cena, Obj[i]);
            end
        end
    end
end
```

De seguida explicar-se-á detalhadamente cada nível operacional, seguido das aplicações *pick-and-place* que serviram de base de testes ao longo do projeto.

4.1.1 Nível Operacional 1: Reconhecimento do Ambiente

Todas as operações de *pick-and-place* têm a necessidade de compreender o ambiente envolvente ao robot responsável pela tarefa, identificando objetos de interesse e respetivas zonas de interesse. Este nível é crucial em todas as abordagens a problemas de *pick-and-place*.

Consequentemente o primeiro passo da arquitetura proposta é identificar o ambiente, criando um modelo 3D do mesmo. Obtendo este modelo 3D é possível identificar todas as estruturas e zonas de interesse numa dada cena. No sistema implementado, o reconhecimento do ambiente foi alcançado utilizando imagens obtidas por um sistema RGBD. Com estas imagens, é possível reunir todas as informações necessárias por forma a identificar estruturas e objetos de interesse.

Adicionalmente recorrendo a um dicionário de objetos e a uma árvore de decisão é possível identificar todos os objetos a ser manipulados e qual a técnica de manipulação mais correta para cada um. Essencialmente processando o modelo 3D, é possível comparar objetos já anteriormente guardados numa base de dados com os objetos encontrados na cena usando a árvore de decisão. De forma a identificar corretamente os objetos é necessário classificar cada um desses objetos com características essenciais como por exemplo área, volumetria, forma ou cor. Finalmente um passo extra é a inserção de intervalos de tolerância para que novos objetos de características similares (por exemplo, objetos ligeiramente maiores ou com uma cor diferente) aos objetos presentes na base de dados sejam manipulados de forma similar. Aplicando a árvore de decisão correta, é possível validar todos os objetos como objetos conhecidos (isto é, já presentes na base de dados) ou como objetos não conhecidos. No segundo caso procede-se a uma atualização da classe dos objetos com um novo elemento (o novo objeto) e com as suas respetivas características.

Outro fator importante durante a fase de reconhecimento é a capacidade de localização autónoma do robot no ambiente onde está inserido. Este fator é particularmente importante em desafios com alvos estacionários ou zonas de interesse imóveis, mas também em ambientes não estruturados visto possibilitar o futuro planeamento, movimentação e atuação do robot de forma mais segura e eficaz. Sucintamente esta localização é realizada pela deteção do sistema RGBD de um objeto conhecido e com um referencial próprio. Aplicando um algoritmo corretivo de transformação de referenciais é possível calibrar o robot para o meio, calibrando o seu referencial de acordo com o referencial do objeto conhecido. Assim consegue-se expressar as informações num referencial constante facilmente transformável no referencial do robot.

4.1.2 Nível Operacional 2: Movimentação do Robot

O nível operacional 2 da arquitetura proposta é composto pelas ações de movimentação do robot: planeamento de trajetórias, tarefas e controlo das articulações do robot. Uma vez criado e processado o modelo 3D do espaço de trabalho do robot é necessário discretizá-lo segundo o espaço de configurações do robot, visto ser este o espaço no qual é possível identificar e definir os movimentos do robot. Adicionalmente é possível também inserir possíveis obstáculos dentro deste espaço de configurações permitindo a implementação de um algoritmo de planeamento de trajetória e caminho capaz de evitar estes obstáculos e eficazmente determinar a sequência de movimentação adequada à operação de *pick-and-place*.

De forma a conseguir completar a operação é colocado neste nível operacional, para além da definição do espaço de configuração, a cinemática do robot, particularmente a cinemática inversa. Comummente robots atuais possuem vários graus de liberdade de forma que para atingir uma determinada pose cartesiana existe mais que uma solução. Assim, para além da habitual

cinemática inversa, é proposto um algoritmo inteligente de adaptação da configuração do robot conforme a posição cartesiana pretendida. Para tal, o espaço cartesiano é dividido em quadrantes. Cada quadrante tem associada uma configuração do robot (para o caso particular deste projeto, ver as diferentes configurações na figura 3.6). Para conseguir fazer a alteração de configurações entre quadrantes, define-se uma posição de segurança considerada origem que permite as trocas de configuração básicas caso seja necessário trocar de quadrante.

Quando determinada a configuração final do robot e atendendo à configuração inicial, é necessário aplicar um algoritmo de planeamento de trajetória capaz de partir o trajeto entre posição inicial e final em vários pontos. Para este fim é utilizado um duplo A*, capaz de rapidamente aproximar-se do espaço envolvente ao objeto e de seguida aproximar-se com precisão do mesmo. Um último passo quando se define o trajeto do robot é a sua validação. Para se conseguir tal objetivo é utilizada a cinemática direta de forma a garantir que o trajeto possa ser percorrido de forma segura entre a origem e a pose pretendida.

4.1.3 Nível Operacional 3: Controlo e Atuação

O último nível operacional desta arquitetura baseia-se num ciclo de controlo. A operação de *pick-and-place* precisa de correções constantes devidos aos erros inseridos pela perceção 3D do mundo e pelas movimentações do robot. Utilizando este ciclo de controlo consegue-se minimizar as incertezas na operação e consequentemente reduzir-se o risco de falha enquanto se maximiza a eficiência da solução.

Adicionalmente neste nível operacional pode-se incluir o *grasping* das peças e o seu *placement*. Para a maximização de aplicações desta solução aqui é inserido a possibilidade também de troca de ferramenta responsável por estas operações. Esta troca facilita a utilização desta arquitetura de forma universal, atendendo ao facto que adicionalmente ao robot surge, no entanto, a necessidade de idealizar uma estação de troca de ferramenta.

Uma vez definida toda a estratégia (poses a alcançar, objetos a manipular, trajeto a seguir, entre outros), o ciclo de controlo é benéfico já que assegura que todos os passos da estratégia são cumpridos sem comprometer a solução final. Além disso, o ciclo de controlo pode ser gerido dinamicamente para que em cada ciclo seja atualizado o estado atual do ambiente e replaneada a calendarização de tarefas de forma a evitar qualquer tipo de erro no processo.

4.1.4 *Framework* ROS

Para o desenvolvimento de uma arquitetura modular adequada ao desenvolvimento de software complexo procura-se uma ferramenta capaz de sustentar este desenvolvimento. O ROS é um *framework* simples de utilizar e com capacidade de descomplexação de problemas típicos da robótica.

Atualmente já um grupo considerável de investigadores tem adotado esta ferramenta permitindo o melhoramento significativo do trabalho até então realizado [9, 10, 58]. O *framework* ROS

permite o desenvolvimento de nós que conseguem comunicar entre si como se se tratasse de programas a funcionar paralelamente. Essencialmente um nó de ROS corresponde à execução de um programa com a publicação ou a subscrição de serviços. Assim, desenvolver uma solução em ROS permite dividir as partes cruciais do projeto em várias camadas de controlo e diferentes nós de ROS cada um desenvolvido para uma parte modular e mais simples de um problema complexo.

Visto ser este um projeto dinâmico surge a necessidade de incluir ferramentas de desenvolvimento eficientes para uma aplicação específica. Por ser um projeto baseado na implementação de algoritmos em código funcional e computacional, torna-se evidente a pesquisa dessas mesmas ferramentas, sendo que ultimamente as seguintes se têm evidenciado:

- QT Creator – Um IDE simples, capaz de organizar projetos baseados em ROS, que permite também realizar testes sobre trabalho realizado [59].
- MoveIt – Um software inovador que incorpora, entre outros, os últimos avanços na tecnologia de planeamento de movimento, manipulação ou reconhecimento de ambiente, adaptável a grande parte dos manipuladores robóticos [60].
- Gazebo – Ambiente de simulação robótico que permite realizar testes de forma rápida e eficaz em ambiente de simulação realísticos [61].
- OpenCV – Biblioteca que potencializa o processamento de imagem, desenhada para maximização da eficiência computacional e para aplicação em *real-time* [62].

4.2 Competições de Robótica

Competições de Robótica têm demonstrado o seu potencial em termos de inovação e desenvolvimento tecnológico. Ao longo do projeto, foi aperfeiçoada a estratégia apresentada em cima recorrendo a duas competições em particular: *European Robotic Challenge (EuRoC)* (secção 4.2.1) e *Amazon Picking Challenge* (secção 4.2.2).

O continuo desenvolvimento da robótica contribui ainda a melhorias significativos no mundo industrial. As competições referidas distinguem-se por permitirem e promoverem a interação entre centros de investigação, universidades e indústrias, proliferando o conhecimento de todas as partes e acelerando o desenvolvimento da robótica.

Ambas as competições apresentavam sistemas complexos de *pick-and-place* com algumas particularidades que serão explicadas de seguida.

4.2.1 European Robotic Challenge (EuRoC)

A competição EuRoC sobressai por ser uma competição direcionada para o desenvolvimento de robótica industrial. A possibilidade de unir operadores humanos a um operário robótico numa única tarefa potencializa a minimização de erros inseridos por cada um dos intervenientes isolados. EuRoC é a competição europeia mais importante de robótica, tendo em conta os investimentos

técnicos e financeiros por detrás desta iniciativa e subdividia-se em três ramos. Para a aplicação da estratégia proposta, apenas um se demonstrava interessante, o *challenge* 2 - um desafio com o objetivo de desenvolver um sistema robotizado para operações de logística e manipulação.

Este desafio motiva ao desenvolvimento de um algoritmo inteligente e adaptativo direcionado para a operação *pick-and-place*. Para tal é fornecido um série de tarefas com grau de dificuldade crescente num ambiente simulado. Associado a este simulador existia informação retirada pelos sensores inseridos nesse ambiente (dois sistemas RGBD) e informação sobre o cenário de cada tarefa. Todas as tarefas partilhavam o espaço de trabalho: uma mesa onde o robot se inseria com objetos espalhados ao longo da mesma mesa.

A figura 4.2 é um esquemático do ambiente completo usado ao longo do projeto EuRoC. Como já referido, nesta aplicação existem dois sistemas *red-green-blue-depth* (RGBD), um associado a um sistema *pan-tilt* acoplado a um poste (posição 3 da figura 4.2) e outro associado ao *gripper* do robot *Lightweight* da KUKA (posição 2 da figura 4.2). Para além deste elementos existem objetos e obstáculos ao longo das diversas tarefas.

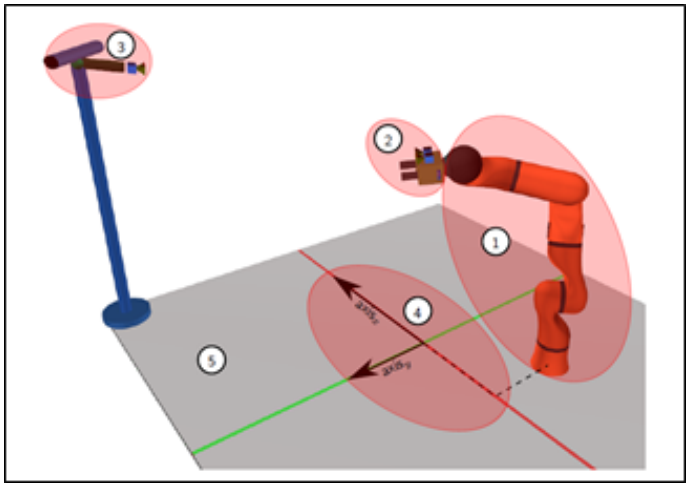

Figura 4.2: Ambiente EuRoC: 1 – *Lightweight* Robot; 2- *Gripper* e sistema RGBD associado; 3 – RGBD associado a poste com *pan-tilt*; 4 – eixos do robot; 5 – Mesa das Operações.

O desafio de logística e manipulação estava sub-dividido em seis tarefas. Como já referido a dificuldade era acrescida com o avançar das tarefas. Tal se deve ao aumento de erros inseridos no ambiente simulado, desde erros de medição, erros de localização ou obstáculos inseridos ao longo da mesa de operações. Todos estes erros e obstáculos têm a dificuldades acrescidas de serem gerados de forma aleatória. Além destes aspetos, a última tarefa pressupõe um tapete rolante que pretende simbolizar uma linha de produção. Nesse sentido, nessa tarefa é inserida a questão de objetos com velocidade constante e velocidade variável. O objetivo de todas as tarefas era a manipulação e o transporte dos objetos de interesse para zonas pré-definidas.

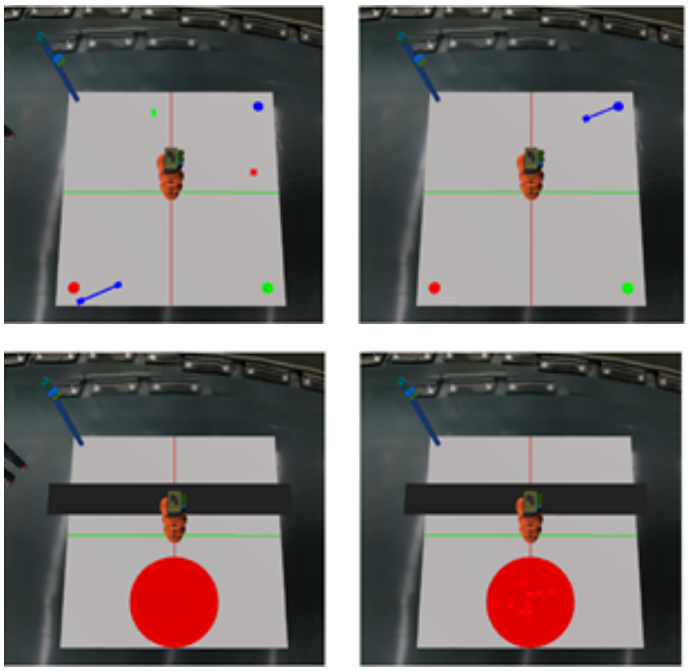

Figura 4.3: Imagens superior - Típica cena do EuRoC, à esquerda o aspeto inicial, à direita o aspeto pretendido; Imagens inferiores - Tarefa com tapete rolante, à esquerda o estado inicial, à direita estado desejado.

4.2.2 Amazon Picking Challenge

Amazon Picking Challenge é uma competição internacional direcionada para o desenvolvimento de algoritmos capazes de solucionar os problemas típicos da robótica tradicional.

Este desafio específico representa a possibilidade de desenvolver uma solução para um dos problemas mais comuns da robótica e da indústria em geral (operação *pick-and-place*), enquanto colocando exigências em termos de diversidades de pedidos, objetos e ambiente envolvente. Estes requisitos obrigam a solução a ser capaz de evitar algumas das complicações associadas com outras metodologias já apresentadas anteriormente, o que conduz à necessidade de utilização de uma estratégia robusta e completa como a sugerida.

O Amazon Picking Challenge é definido por uma prateleira com subdivisões repletas de objetos (ver figura 4.4).

Figura 4.4: Amazon Picking Challenge Workstation

Este problema sugere a utilização de um sistema de visão global capaz de reconhecer todo o ambiente envolvente ao robot. Além deste aspeto surge a necessidade de entender qual a melhor configuração e forma de manipulação para cada objeto presente na prateleira. Finalmente a organização sugeria a utilização de um grupo restrito de robots por forma às soluções desenvolvidas poderem ser eficazmente colocadas nos seus armazéns. Consequentemente, o Amazon Picking Challenge é a oportunidade perfeita de testar soluções eficientes no campo da robótica.

4.3 Resumo do Capítulo

Neste capítulo foi proposta uma abordagem flexível e genérica para aplicações de *pick-and-place*, baseada em três níveis operacionais e com o recurso do *framework* ROS.

Aliar ROS à abordagem proposta permite desenvolver separadamente cada um dos níveis operacionais apresentados, o que conduz à construção de soluções modulares, mais simples de implementar e testar.

Mais ainda, foi descrita dois exemplos de aplicação para esta abordagem, dois desafios de robótica: o EuRoC e o Amazon Picking Challenge.

Capítulo 5

Implementação

Neste capítulo são apresentadas as soluções implementadas ao longo do projeto, nomeadamente o planeador inovador na secção 5.1 e os princípios aplicados nos desafios EuRoC (secção 5.2) e Amazon Picking Challenge (secção 5.3).

5.1 Planeador Duplo A*

Os planeadores de trajetória procuram minimizar o trajeto entre dois pontos. Para este efeito existe a necessidade de definir o espaço de configurações que o mesmo robot pode assumir. Visto que o robot é controlado por posição da articulação é benéfico definir esse espaço de configurações no espaço de articulações.

Os robots manipuladores atuais têm na sua generalidade seis graus de liberdade, o que conduz a problemas na discretização do espaço de configurações. Maioritariamente esses problemas devem-se a falta de memória para armazenar esse espaço. Suponhamos que se pretende uma definição do espaço de um grau. Para esse caso, e considerando o intervalo de [-π, π] para todas as juntas temos:

$$Mem = N\_Config_1 * N\_Config_2 * N\_Config_3 * N\_Config_4 * N\_Config_5 * N\_Config_6 \quad (5.1)$$

$$N\_Config_1 = N\_Config_2 = ... = N\_Config_6 = 360(1^o/celula) \quad (5.2)$$

$$Mem = 360^6 \approx 2.2 * 10^{15} \quad (5.3)$$

Mesmo que se considere um único *byte* para cada uma das posições deste espaço ter-se-ia aproximadamente 2200 TB. Por este motivo, a precisão desta discretização teve de ser reduzida. Optou-se por implementar, então um duplo planeador, com o primeiro a fazer o planeamento da aproximação ao destino pretendido, discretizando-se para este caso o espaço de configurações em divisórias de 20º (ver figura 5.1). O segundo planeador permitia adicionar precisão ao sistema, uma vez que considerava apenas cada uma das divisórias discretizadas e alcançadas pelo planeador

de aproximação e as dividia em frações de 1^o. Com esta modificação as posições de memórias necessárias são 18^6 o que corresponde a $34 * 10^6$ afetado pela estrutura de memória definida. Ao considerar novamente apenas um *byte* por posição ter-se-ia aproximadamente 34MB, valor aceitável em termos de uso de memória e bastante inferior aos 2200 TB referidos.

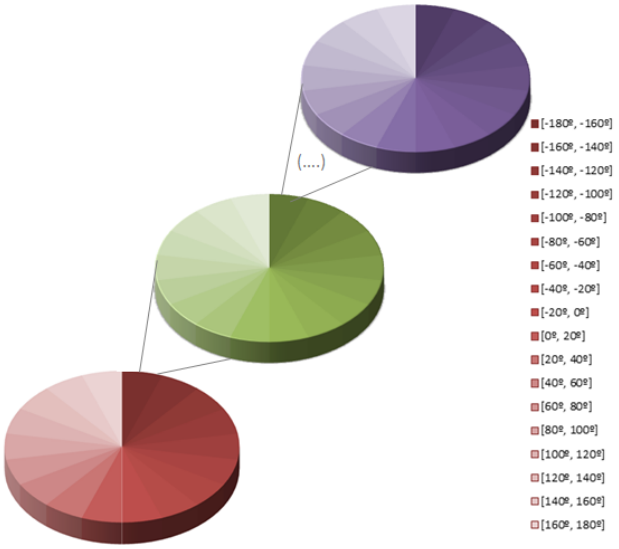

Figura 5.1: Diagrama Ilustrativo do Espaço de Configurações

Aqui a única questão seria a utilização de duas estruturas para guardar os dois espaços de configurações, o de aproximação e o de precisão. Contudo, por questões de reaproveitamento de memória, o espaço de configurações no movimento de precisão utiliza a estrutura do algoritmo de aproximação reescrevendo as propriedades passíveis de serem alteradas (nomeadamente posições das articulações em cada configuração).

Uma vez definida a estratégia de discretização do espaço, surge a necessidade de definir qual a estrutura de dados a utilizar ao longo do planeador. Atendendo ao facto do duplo planeador ser concordante com os algoritmos do tipo A*, faz sentido a estrutura possuir o índice correspondente a cada configuração e a respetiva configuração das juntas. Além destes campos, o algoritmo A* funciona como a iteração de vizinhos em direção ao ponto de destino. Para diferenciar todos os vizinhos são definidas regras de orientação do trajeto (heurísticas) e a distância entre estes vizinhos e o vizinho que lhes dá origem. Ambos os valores devem aparecer na estrutura, bem como a relação entre os vizinhos. Desta forma a sequência entre os mesmos deve ser recordada, por intermédio de uma variável *pai* que armazena a origem imediatamente anterior de cada vizinho. Optou-se por usar um vetor com esta estrutura, tendo-se optado ainda por usar uma estrutura *heap* para criar uma lista aberta (correspondente aos vizinhos ainda não processados e ordenados por relevância para a determinação do projeto, isto é, com menor custo). Finalmente existe uma propriedade para cada uma das configurações que identifica a mesma como passível de atingir (livre) ou como obstáculo. Isto permitirá à frente isolar certas configurações e desviar corretamente de obstáculos.

Para determinar os custos associados a cada vizinho, criaram-se duas funções: uma de heurística e outra de distância. A função distância devolvia como o somatório do erro de cada configuração das articulações quando comparada com a anterior (isto é, pelo *pai*). Quanto à função de heurística, o retorno era o mesmo erro mas afetado por um fator de importância consoante a articulação em questão. De referir que para as configurações que são definidas como obstáculos este custo é muito elevado (idealmente é infinito). Assim, evita-se procurar estas células de configurações sendo consequentemente descartada a hipótese de colidir com algum obstáculo.

Com estes dados, é possível a cada momento definir um total de $3^6 - 1$ vizinhos, e para cada um deles calcular o custo total associado (isto é, distância mais heurística). Com base nesse custo, isola-se para uma lista fechada (*heap*) os melhores vizinhos. Mais ainda, inicialmente são definido os índices das configurações inicial e final, pelo que encontrado o vizinho com correspondência à configuração final o processo é terminado e na lista são colocados consequentemente os índices das configurações que compõem o trajeto.

Finalmente, concluído o preenchimento da lista, basta retirar da lista de índices, os valores que após transformação conseguem criar uma lista de configurações que o robot deve adotar para atingir o ponto desejado partindo do ponto inicial.

5.2 EuRoC

Durante o desafio do EuRoC, a arquitetura implementada necessitava de considerar diversos objetos que seriam manipulados ao longo de cada tarefa. Assim, apareceu a necessidade de criar um algoritmo capaz de planear e controlar a sequência de ações correta para a operação de *pick-and-place* (o programa de controlo). Paralelamente, a necessidade de reconhecer o ambiente levou-nos a desenvolver e implementar dois algoritmos (um para cada câmara) onde era realizado processamento de imagem para identificar a cena, objetos e zonas de destino. Finalmente para controlar o movimento do robot existe um quarto programa de movimentação e controlo do robot (ver figura 5.2).

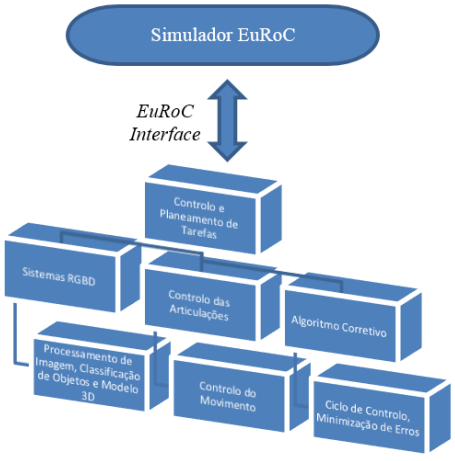

Figura 5.2: Diagrama de Interação do EuRoC

Desenvolver uma solução em ROS oferece a possibilidade de dividir as partes cruciais do projeto em diferente nós. Para esta aplicação específica, foram então criados quatro nós: um nó para a imagem fornecida pela câmara do poste (direcionada para o reconhecimento da cena), um para o controlo do movimento do robot e das operações de *grabbing/placement*, outro para a imagem fornecida pelo sistema RGBD presente no *gripper* e, finalmente, um último nó que controlasse toda a operação, ou seja, o nó de controlo do sistema. Uma vez criados é necessário associar as mensagens de cada sensor ao nó correspondente. Isto permite, por exemplo, fundir a informação da câmara de Depth com a informação da câmara RGB de cada um dos sistemas RGBD (ver figura 5.3).

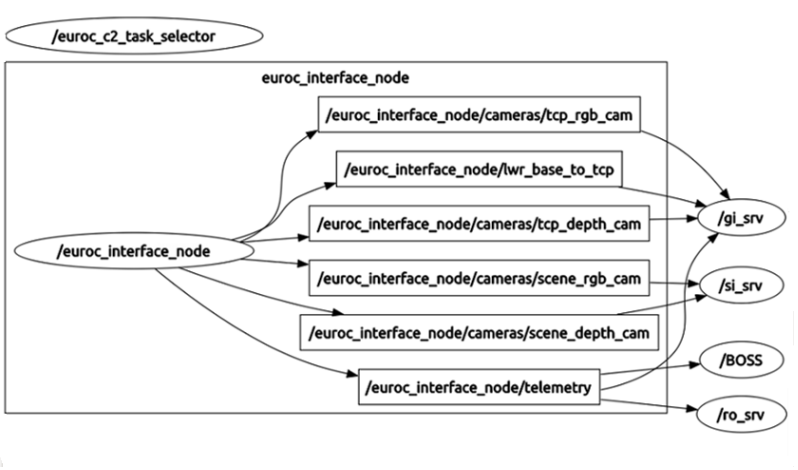

Figura 5.3: Nós de ROS implementados para EuRoC

Adicionalmente o uso do *framework* ROS permite eficientemente dividir todas os níveis operacionais da nossa metodologia. Mais ainda, o *framework* ROS permite fazer a comunicação entre vários programas de uma forma *standard* e simples ao criar e subscrever serviços, requisitando informação e reunindo as respostas desses serviços. No exemplo do EuRoC, foi definido o nível operacional de controlo como o nó principal e o que controlava todos os processos. Este nó requisitava a imagem da cena no início do processo para criar o modelo 3D e para com base nesse modelo poder enviar ordens de movimento ao robot. Essencialmente o modelo 3D advinha de um processamento de imagem que permitia identificar objetos de interesse e as suas coordenadas. Baseado nessas informações o robot é comandado até à posição do objeto detetado, posicionado o *gripper* acima do mesmo objeto 50 cm. Uma vez sobre o objeto, a informação captada pelo sistema RGBD presente no *gripper* permite corrigir a trajetória entre esse ponto e o objeto. Essencialmente o nó de imagem do sistema presente no *gripper* publica as poses dos objetos de forma mais precisa e exata e com esta informação o planeamento de trajetória até entrar em contacto com os objetos de interesse é ajustado e controlado. O nó de movimentação do robot e de controlo do *grasping* das peças, recorre a esta informação visual continuamente enquanto desce até ao objeto, conseguindo depois segurá-lo, manipulá-lo e transportá-lo até ao destino previamente definido.

A seguir serão explicados detalhadamente todos os aspetos da arquitetura implementada.

5.2.1 Processamento de Imagem e Reconhecimento do Ambiente

Nesta aplicação, o reconhecimento do ambiente (nível operacional 1 da nossa metodologia) é inicialmente conseguido usando as imagem das câmaras fixas ao poste. As imagens obtidas pela câmara conseguem obter e transmitir a informação geral do ambiente (neste caso, das mesa de operações). Das imagens adquiridas, é possível reunir e usar a informação da câmara *Depth* para isolar os objetos e estruturas cruciais na mesma mesa de operações. Sabendo que o robot e os objetos de interesse estão localizados em cima da mesa, é necessário definir o plano da mesa na imagem adquirida. Uma vez determinado esse plano, esta informação é subtraída à imagem de *depth* original. Posteriormente é gerada uma imagem do fundo da cena, de forma a que esse mesmo fundo seja removido, isolando-se, assim, mesa da restante cena, e dentro da mesa consegue-se visualizar e identificar os objetos e estruturas presentes (ver figura 5.4).

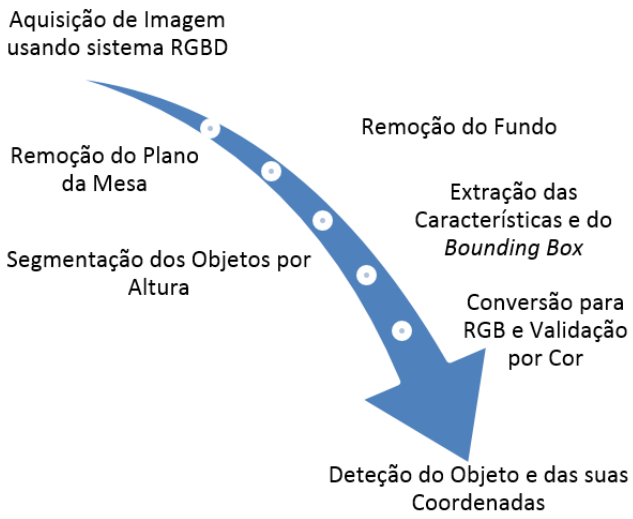

Figura 5.4: Diagrama de Processamento de Imagem EuRoC

Uma vez detetados os objetos, e com base na informação que se retira dos mesma, pode-se completar o reconhecimento deste usando uma árvore de decisão. Esta árvore de decisão considera características cruciais como a excentricidade, área, volume ou número de cantos e compara-os com a informação presente no dicionário da classe de objetos. Adicionalmente, com a informação da câmara RGB é possível aplicar filtros de cor aos objetos, o que conduz à eliminação de estruturas de cor bem definidas que não se tratam de objetos de interesse (por exemplo, neste caso, o robot). Isto permite localizar e classificar todos os objetos encontrados (ver figura 5.5). Acabado este processo, apenas é necessário realizar uma troca de referenciais entre o referencial das câmaras que captaram as imagens e o referencial do robot no mundo.

A inserção de tolerância acrescenta flexibilidade ao sistema em geral. Existem casos nos quais um novo objeto consegue enquadrar-se em várias característica de outro já anteriormente guardado na classe de objetos. Tal se pode dever a pequenas mudanças como, por exemplo, a alteração da cor ou ligeiro aumento do tamanho. Assim durante a aplicação da árvore de decisão, a aplicação de

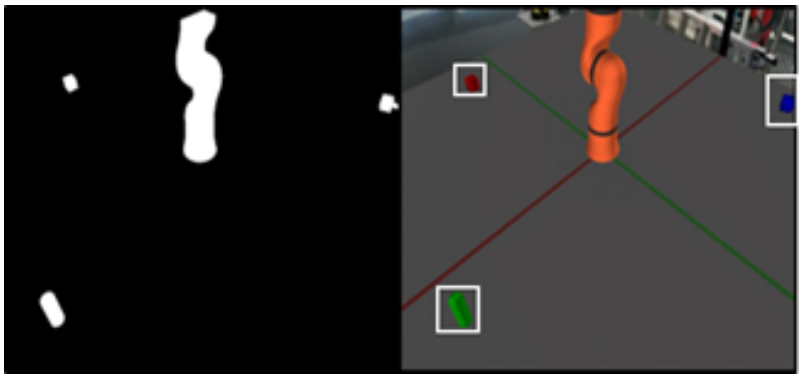

Figura 5.5: Exemplo de Imagens de Depth e RGB de EuRoC

níveis de tolerância permitiria identificar esses objetos como objetos similares aos anteriormente guardados. Assim, o novo objeto pode ser manipulado de forma idêntica, informando o restante sistema da correta sequência de ações já definidas para a sua manipulação.

Outro aspeto importante a considerar nesta aplicação, é o facto do poste ser imóvel e por vezes o robot bloquear parte da cena. Para corrigir este problema consideram-se duas abordagens possíveis:

1. movimentar o robot e repetir o processo de identificação da cena a cada dez segundos;
2. utilizar o sistema de câmaras presente no *gripper* para investigar a zona bloqueada.

Desta forma, garante-se um processamento de identificação do ambiente dinâmica que consegue avaliar corretamente toda a cena.

Essencialmente, aplicando estas técnicas consegue-se responder à necessidade de detetar e classificar objetos. Com este algoritmo é também possível identificar novos objetos e classificá-los como similar a algum previamente guardado na base de dados ou como inteiramente novo, atualizando a mesma base de dados com as características deste. É ainda possível isolar claramente objetos de estruturas ou obstáculos indesejados, guardando-se apenas informação sobre os objetos que se pretende manipular.

5.2.2 Movimentação e Atuação do robot

Uma vez que a cena é conhecida e descrita, recorre-se à cinemática inversa para determinar a configuração correta de cada articulação do robot e usam-se os atuadores do mesmo robot de modo a colocar estas articulações no seu estado desejado. De forma a alcançar essas configurações definiu-se um algoritmo de controlo de articulação baseado no espaço de configurações disponível para o robot de forma a evitar obstáculos ou objetos indesejados.

Existe ainda a necessidade de definir a sequência de alteração da articulação. Para tal utiliza-se um algoritmo de planeamento de trajetória. Como forma de validação do novo algoritmo de planeamento, foi utilizado o planeador duplo A*. Mais ainda, neste trabalho a movimentação do robot foi dividida de forma a se manter controlo total sobre a trajetória definida e de forma a

se corrigir possíveis erros que aparecessem devido ao algoritmo de controlo de articulação. Esta divisão vai de encontro com o duplo A* visto que assim era possível a aplicação do A* macro para a primeira parte do movimento e do A* preciso para a segunda parte.

Finalmente concluída a movimentação, a questão da atuação neste caso é realizada com dois dedos paralelos que funcionavam como pinça elétrica (ou seja, podia ser controlada como uma articulação elétrica). Outra ferramenta nesta fase de atuação é a inserção de um sensor de força localizado no *gripper* do robot que reporta se o objeto está seguro ou não pelo mesmo *gripper* e que informa se a força exercida pela mesmo é adequada para cada objeto.

5.2.3 Controlo Corretivo

O último nível operacional da metodologia proposta é o nível operacional de controlo. O controlo corretivo foi implementado nesta aplicação usando o nó de interface do EuRoC. Esta interface permite compreender e saber como as articulações se encontram a cada momento da trajetória. Outro aspeto centra-se no sistema de imagem presente no *gripper*. Durante o *picking* e o *placing* é gerado um ciclo de controlo com *feedback* com base na imagem processada do sistema RGBD do *gripper*.

Uma vez que a câmara responsável por captar a cena e os atuadores que controlam as articulações inserem frequentemente alguns erros ao processo de *pick-and-place*, a nossa abordagem implementou um ponto intermédio para o *gripper* do robot a meio metro de altura acima dos objetos ou zonas de colocação desejados. Uma vez nesse ponto, as imagens adquiridas pelo sistema RGBD presente no *gripper* eram usadas para redefinir a posição do objeto garantido uma manipulação mais precisa e exata do objeto. Esta ação era repetida em ciclo enquanto o objeto era lentamente colhido ou colocado.

5.2.4 Planeamento de Tarefa

Os nós de processamento de imagem e de movimentação do robot permitem identificar e mover o sistema a cada momento. No entanto, para tornar o sistema mais eficiente existe a necessidade de desenvolver um algoritmo de planeamento de tarefas que compreende a sincronização com o simulador, a interpretação da tarefa e a construção da lista de procedimentos a realizar. Para esta aplicação particular existia um ficheiro YAML que descrevia a tarefa que se pretendia cumprir. Associando a este ficheiro a informação dos sistemas RGBD é possível identificar todos os objetos e desenvolver uma árvore de sequência de tarefas para cada objeto. Posteriormente em ciclo, é processada essa sequência associada a cada um dos objetos até o mesmo objeto ser colocado no local desejado. Após concluída a tarefa, pode-se remover o ramo onde esse sequência estava presente, repetindo o processo para o ramo seguinte até não existirem mais ramos (ver algoritmo 2).

Algoritmo 2: *Pick-and-place* do exemplo EuRoC

```
Inputs: EuRoC_Interface + Sistemas_RGBD + YAML ;
Outputs: end_effector_pose;

Tarefa = DescobrirTarefa(YAML);
Cena = Identificacao_cena(RGBD_Cena);

'usar RGBD_Gripper se necessário'
Obj = Detecao_Objetos(RGBD_Cena(+RGBD_Gripper));

Sequencia = Descobrir_Sequencia(Tarefa, Cena, Obj);
Ac_Tree = Criar_Arvore_Accao(Sequencia);

while ( NumeroDe(Ac_Tree.Ramos) > 0 ) do
    for (NumeroTentativas) do
        ExecuteSequencia(Ac_Tree.Ramo[i]);
        OK = Avaliar_Placing(Ac_Tree.Ramo[i], RGBD_Cena(+RGBD_Gripper));

        if (OK) then
            RemoveRamo(Obj[i]);
            Break;
            else
                Atualizar_Sequencia_Cena(Ac_Tree.Ramo[i], Tarefa, Cena, Obj[i]);
            end
        end
    end
end
```

5.3 Amazon Picking Challenge

O desafio *Amazon Picking Challenge* foi um problema de *pick-and-place* com vários objetos que necessitavam de ser detetados e manipulados. Para esse fim teria de se desenvolver um algoritmo multifacetado que capaz de planear a correta sequência de ações a tomar de forma a controlar toda a operação (recorde-se a figura 4.1).

Novamente desenvolver a solução em ROS permitiu dividir as partes cruciais do projeto em diferentes nós, dividindo também os níveis operacionais da nossa metodologia de forma eficiente. O nível operacional de controlo assumiu, também aqui, o papel preponderante de coordenação entre os restantes nós. No caso do desafio *Amazon Picking Challenge*, este nó de controlo controlava todo o processo, começando por exigir a um nó de imagem a criação do modelo 3D do mundo envolvente, ordenando de seguida o nó de movimento do robot a seguir as indicações baseadas nesse modelo, tudo isto enquanto se aplicava um algoritmo de controlo de correção à operação.

De seguida, estes aspetos serão revistos de forma detalhada.

5.3.1 Processamento de Imagem e Reconhecimento de Ambiente

Para a aplicação do *Amazon Picking Challenge*, o reconhecimento do ambiente (nível operacional 1 da nossa metodologia) é conseguido usando imagens de um sistema RGBD colocado acima da última articulação do robot (imagem 4.4).

Usando as imagens adquiridas, é possível detetar os objetos, a prateleira, obstáculos e outras estruturas cruciais para o desafio. Uma vez detetados a prateleira e os objetos, devido à informação retirada pelo sistema RGBD, é possível auto-calibrar a posição do robot usando um algoritmo autónomo e corretivo. Este algoritmo consegue interligar todos os referenciais (robot, camera, divisões da prateleira...) de forma a que o processo de conversão de poses entre referenciais seja rapidamente e facilmente conseguido (ver figura 5.6).

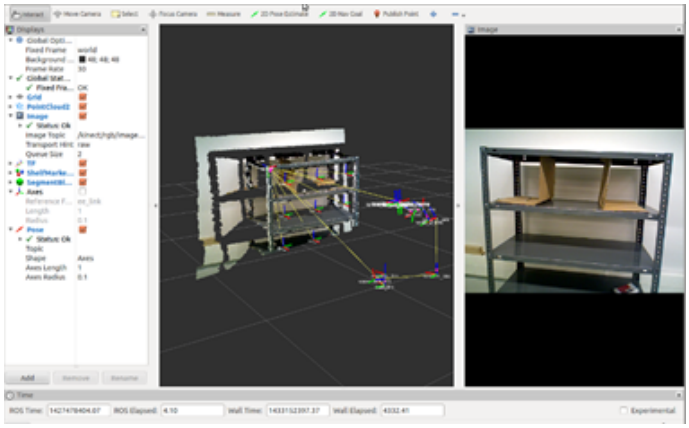

Figura 5.6: Visualização em RVIZ da cena de Amazon Picking Challenge

Simultaneamente, o reconhecimento dos objetos é conseguido recorrendo novamente a uma árvore de decisão. Esta árvore de decisão concentra-se nos mesmos fatores de excentricidade, área, volume, cantos, mas possui a vantagem de serem conhecidos os objetos em cada ponto da

prateleira. Assim o processamento da árvore de decisão é mais reduzido, uma vez que se pode completar a classe de objetos com a divisória de prateleira onde o objeto se insere. A inserção de tolerâncias é vantajosa neste caso devida à similaridade de diversos dos objetos. Assim objetos com comprimentos ligeiramente diferentes podem ser agrupados com outros que possuam a mesma forma mas menores, minimizando a classe de objetos.

Aplicando estas técnicas consegue-se identificar os objetos presentes na prateleira, isolando-os da mesma, eliminando obstáculos, identificando a sua pose no referencial do mundo e conseguindo-se preparar a sequência de ações a realizar de seguida.

5.3.2 Movimentação do robot e Actuação

A movimentação do robot advém de um processo de cinemática inversa baseada na informação geométrica que é retirada do processamento de imagem. Com a determinação da pose dos objetos de interesse é determinado o correspondente dessa pose em espaço de articulações. Para alcançar esse objetivo foi definido um algoritmo que inicialmente identifica o quadrante no qual se insere o objeto de interesse no referencial do robot. Com base nesse informação, é definida a única configuração do robot que alcança determinada pose com determinada orientação enquanto respeitando a estrutura de configurações definida para o quadrante em questão (ver na figura 3.6 as diferentes configurações possíveis para os quadrantes). Com o ponto inicial e final bem definido em espaço de articulações é aplicado um algoritmo de planeamento de trajetória que define o trajeto entre os dois pontos. Este trajeto constitui um conjunto de configurações do robot (que origina um conjunto de pontos em espaço geométrico) que permite dividir a trajetória mantendo-se assim mais controlável o movimento do robot entre a pose inicial e final. Nos passos finais do cumprimento deste algoritmo surge a necessidade de aumentar a precisão dos pontos perto do destino final. Assim, aplicando este método consegue-se definir um planeamento macro do robot que aproxima o robot do objeto, e um planeamento preciso que evita todo o tipo de estruturas e conduz o robot até ao objeto a ser manipulado.

Finalmente, chegado ao objeto, é necessário uma ferramenta para o manipular. Assim, atendendo à diversidade de objetos desta competição, foi desenhada e definida uma estação de ferramentas onde se alocou vários tipos de *end-effectors* desde pinças pneumáticas, *grippers* de sucção ou mãos elétricas (por exemplo, mão ROBOTIQ). Cada objeto anteriormente identificado possui também uma correspondência a uma ferramenta específica. Portanto, no início do processo procedia-se à alteração da ferramenta conforme o objeto que se pretendia tratar. Assim, conseguia-se abranger um maior leque de objetos e consequentemente obter uma solução mais flexível e genérica.

5.3.3 Controlo Corretivo e Planeamento da Tarefa

No desafio *Amazon Picking Challenge* optou-se por identificar claramente a cena no início do processo e sempre que o robot tomava consciência da prateleira, planeando ou replaneando todo o

restante processo a partir desses pontos. O controlo era então realizado com base nesses momentos e também na confiança existente sobre ferramentas e robot usados.

O planeamento de tarefa era definido no início de forma a se encontrar a sequência de ações totais com base no aspeto total da cena. Existia ainda um ficheiro que descrevia a tarefa e a disposição dos objetos segundo divisórias da prateleira.

Outro fator inserido para este caso específico foi a definição de poses para cada uma das divisórias da prateleira, conseguindo-se assim definir configurações de aproximação de forma padronizada e configurações intermédias de troca entre divisórias, acelerando o processo em geral.

Essencialmente, o algoritmo de controlo de gestão do sistema assenta nos princípios do algoritmo 1, com a inserção de alguns passos específicos desta aplicação (ver algoritmo 3).

Algoritmo 3: *Pick-and-place* do exemplo Amazon Picking Challenge

```
Inputs: Robot_Interface + Sistema_RGBD + Ficheiro_Disposição_Prateleira ;
Outputs: end_effector_pose;

Tarefa = DescobrirTarefa(Sistema_RGBD);
Cena = Identificacao_cena(Sistema_RGBD);
Obj = Detecao_Objetos(Sistema_RGBD);
Confirmacao_Objetos(Ficheiro_Disposição_Prateleira, &Obj);

Sequencia = Descobrir_Sequencia(Tarefa, Cena, Obj);
Ac_Tree = Criar_Arvore_Accao(Sequencia);

while ( NumeroDe(Ac_Tree.Ramos) > 0 ) do
    Ferramenta = Descobrir_Ferramenta(Ac_Tree.Ramo[i]);
    if (Necessidade_Troca(Ac_Tree.Ramo[i], Ferramenta)) then
        Trocar_Ferramenta();
    end

    for (NumeroTentativas) do
        ExecuteSequencia(Ac_Tree.Ramo[i]);
        OK = Avaliar_Placing(Ac_Tree.Ramo[i], Sistema_RGBD);

        if (OK) then
            RemoveRamo(Obj[i]);
            Break;
            else
                Atualizar_Sequencia_Cena(Ac_Tree.Ramo[i], Tarefa, Cena, Obj[i]);
            end
        end
    end
end
```

5.4 Resumo do Capítulo

Ao longo deste capítulo foram explicadas os passos principais desenvolvidos ao longo deste trabalho, ou seja:

1. O planeador inovador de trajetória, Duplo A*, criado, desenvolvido e testado ao longo desta tese;

2. As soluções encontradas para os desafios de robótica: EuRoC e Amazon Picking Challenge.

O planeador baseou-se na separação do movimento do robot em dois passos, um de aproximação e um de precisão. O primeiro pretende uma aproximação do robot ao objeto de interesse de forma rápida, enquanto que o segundo conduz o robot ao objeto de forma mais lenta mas com um erro mínimo.

Quanto às soluções desenvolvidas baseiam-se na arquitetura descrita ao longo do capítulo 4. Neste capítulo são descriminadas e particularizadas as soluções para cada um dos níveis operacionais.

Capítulo 6

Validação Experimental

Neste capítulo serão apresentados os resultados do planeador desenvolvidos durante todo o projeto (secção 6.1) e a sua comparação com o planeador atualmente adotado em soluções robóticas modernas por meio da interface *MoveIt* (secção 6.2). Finalmente as duas últimas secções pretendem mostrar os resultados da aplicação de toda a abordagem descrita anteriormente nos casos do EuRoC (secção 6.3) e do Amazon Picking Challenge (secção 6.4).

6.1 Validação do Planeador

A inserção do algoritmo A* no duplo planeador já explicado anteriormente (ver secção 5.1) permite teoricamente minimizar o trajeto. Para testar este facto foram definidas seis poses cartesianas em diferentes quadrantes espaciais por forma a testar o comportamento do algoritmo.

Tendo-se definido que o espaço de configurações sobre o qual este algoritmo iria atuar era o espaço de articulações de seguida apresentar-se-ão os resultados da evolução da posição das articulações em cada iteração. De referir que o robot utilizado para estes testes foi o robot UR5. Todas os trajetos que serão apresentados partiam de um ponto similar ((x,y,z) = (-0.275, -0.110, 0.425)) e com orientação similar ((r,p,y) = (-1.45,-0.70,1.45)). Foram testadas múltiplas posições e todas apresentaram resultados idênticos. Contudo esta pose foi escolhida baseada nas aplicações futuras de *pick-and-place* nomeadamente o desafio Amazon Picking Challenge (secção 5.3).

Para cada uma das poses testadas surgirão duas tabelas, uma do planeador na fase de aproximação e outra do planeador na fase de precisão. Cada uma terá o número de iterações da respetiva configuração a cada iteração e apresentará essas configurações num vetor de θ_1 a θ_6 (de acordo com a articulação e com o valor apresentado em graus). Será ainda descrita a configuração final para cada uma das poses. As poses escolhidas foram então:

1. Pose 1: ((x,y,z) = (0.100, 0.075, 0.725))
2. Pose 2: ((x,y,z) = (-0.125, -0.250, 0.750))
3. Pose 3: ((x,y,z) = (-0.450, -0.350, 0.650))

4. Pose 4: ((x,y,z) = (0.205, -0.205, 0.600))

5. Pose 5: ((x,y,z) = (-0.200, 0.220, 0.450))

6. Pose 6: ((x,y,z) = (0.330, 0.100, 0.690))

Para a pose 1, a configuração final resultante da cinemática inversa é (167.5 ; -53.0 ; -49.5 ; 107.8 ; -100.0 ; 140.6). Ver tabelas 6.1 e 6.2.

Tabela 6.1: Fase de Aproximação: Evolução das articulações - Pose 1

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-80	-100	-60
2	20	-20	-120	-60	-120	-40
3	40	-40	-100	-40	-120	-20
4	60	-60	-80	-20	-120	0
5	80	-60	-60	0	-120	20
6	100	-60	-60	20	-120	40
7	120	-60	-60	40	-120	60
8	140	-60	-60	60	-120	80
9	160	-60	-60	80	-120	100
10	160	-60	-60	100	-120	120
11	160	-60	-60	100	-120	140
12	160	-60	-60	100	-100	140

Tabela 6.2: Fase de Precisão: Evolução das articulações - Pose 1

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	161.1	-58.9	-58.9	100	-101.1	140
2	162.2	-57.8	-57.8	101.1	-101.1	140
3	163.3	-56.7	-56.7	102.2	-101.1	140
4	164.4	-55.6	-55.6	103.3	-101.1	140
5	165.6	-54.4	-54.4	104.4	-101.1	140
6	166.7	-53.3	-53.3	105.6	-101.1	140
7	166.7	-53.3	-52.2	106.7	-101.1	140
8	166.7	-53.3	-51.1	107.8	-101.1	140
9	166.7	-53.3	-50	107.8	-101.1	140

Comparando as posições das articulações finais obtidas com as posições das articulações pretendidas verifica-se um erro inferior a 1º por cada articulação, resultando num erro total aglomerado entre todas as articulações de 3.3º. Recordando que a pose desejada (x,y,z) era (0.100,0.075,0.725), alcançou-se a posição (0.095, 0.073, 0.724) que corresponde a um erro de 5mm.

Para a pose 2, a configuração final resultante da cinemática inversa é (55.2 ; -26.6 ; -82.9 ; -80.3 ; -147.3 ; -48.6). Ver tabelas 6.3 e 6.4.

Novamente analisando o erro total final tem-se um somatório de 2.9º. Sendo a posição final obtida (-0.125, -0.243, 0.750), o que corresponde a um desvio de 2mm.

Tabela 6.3: Fase de Aproximação: Evolução das articulações - Pose 2

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-80	-100	-60
2	20	-20	-120	-100	-120	-60
3	40	-40	-100	-100	-140	-60
4	40	-40	-100	-100	-160	-60
5	40	-20	-80	-80	-140	-40

Tabela 6.4: Fase de Precisão: Evolução das articulações - Pose 2

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	40	-22.2	-82.2	-81.1	-142.2	-41.1
2	41.1	-23.3	-83.3	-81.1	-143.3	-42.2
3	42.2	-24.4	-83.3	-81.1	-144.4	-43.3
4	43.3	-25.6	-83.3	-81.1	-145.6	-44.4
5	44.4	-26.7	-83.3	-81.1	-146.7	-45.6
6	45.6	-26.7	-83.3	-81.1	-147.8	-46.7
7	46.7	-26.7	-83.3	-81.1	-147.8	-47.8
8	47.8	-26.7	-83.3	-81.1	-147.8	-48.9
9	48.9	-26.7	-83.3	-81.1	-147.8	-48.9
10	50	-26.7	-83.3	-81.1	-147.8	-48.9
11	51.1	-26.7	-83.3	-81.1	-147.8	-48.9
12	52.2	-26.7	-83.3	-81.1	-147.8	-48.9
13	53.3	-26.7	-83.3	-81.1	-147.8	-48.9
14	54.4	-26.7	-83.3	-81.1	-147.8	-48.9

Para a pose 3, a configuração final resultante da cinemática inversa é (31.5 ; -10.5 ; -66.0 ; -109.8 ; -123.8 ; -43.9). Ver tabelas 6.5 e 6.6.

Tabela 6.5: Fase de Aproximação: Evolução das articulações - Pose 3

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-80	-100	-60
2	20	-20	-120	-100	-120	-60
3	20	-20	-100	-120	-140	-60
4	20	-20	-80	-120	-140	-60
5	20	-20	-60	-100	-120	-40

No terceiro caso, o duplo planeador obteve um erro total de 3^o, tendo-se obtido a posição (-0.442, -0.340, 0.657) que corresponde a um desvio de 14mm.

A pose 4 corresponde a uma configuração final de (126.1; -28.9; -64.2; 101.5; -141.2; 146.3). Ver tabelas 6.7 e 6.8.

Tabela 6.6: Fase de Precisão: Evolução das articulações - Pose 3

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	21.1	-20	-62.2	-102.2	-122.2	-41.1
2	22.2	-18.9	-63.3	-103.33	-123.3	-42.2
3	23.3	-17.8	-64.4	-104.4	-124.4	-43.3
4	24.4	-16.7	-65.6	-105.6	-124.4	-44.4
5	25.6	-15.6	-66.7	-106.7	-124.4	-44.4
6	26.7	-14.4	-66.7	-107.8	-124.4	-44.4
7	27.8	-13.3	-66.7	-108.9	-124.4	-44.4
8	28.9	-12.2	-66.7	-110	-124.4	-44.4
9	30	-11.1	-66.7	-110	-124.4	-44.4
10	31.1	-11.1	-66.7	-110	-124.4	-44.4

Tabela 6.7: Fase de Aproximação: Evolução das articulações - Pose 4

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-80	-100	-60
2	20	-20	-120	-60	-120	-40
3	40	-40	-100	-40	-140	-20
4	60	-40	-80	-20	-160	0
5	80	-40	-80	0	-160	20
6	100	-40	-80	20	-160	40
7	120	-40	-80	40	-160	60
8	120	-40	-80	60	-160	80
9	120	-40	-80	80	-160	100
10	120	-40	-80	100	-160	120
11	120	-40	-80	100	-160	140
12	120	-20	-60	100	-140	140

Tabela 6.8: Fase de Precisão: Evolução das articulações - Pose 4

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	120	-22.2	-61.1	100	-141.1	141.1
2	121.1	-23.3	-62.2	101.1	-142.2	142.2
3	122.2	-24.4	-63.3	101.1	-142.2	143.3
4	123.3	-25.6	-64.4	101.1	-142.2	144.4
5	124.4	-26.7	-64.4	101.1	-142.2	145.6
6	125.6	-27.8	-64.4	101.1	-142.2	145.6
7	125.6	-28.9	-64.4	101.1	-142.2	145.6

Comparando os resultados do 4º teste com a configuração inicial desejada encontra-se um erro total de 2.8º. Chegou-se à posição (x,y,z) de (0.203, -0.207, 0.599), que corresponde a um desvio cartesiano de 3mm.

A 5ª pose escolhida, aplicando o mesmo princípio de cinemática inversa exigiria uma configuração final de (-87.7; 11.3; -119.1; -120.4; -7.1; 7.7). A evolução do cálculo da trajetória está presente nas tabelas 6.9 e 6.10.

Tabela 6.9: Fase de Aproximação: Evolução das articulações - Pose 5

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-60	-100	-60
2	-20	0	-120	-80	-80	-40
3	-40	0	-120	-100	-60	-20
4	-60	0	-120	-120	-40	0
5	-80	0	-120	-140	-20	0
6	-100	0	-120	-140	-20	0
7	-100	20	-120	-120	0	0

Tabela 6.10: Fase de Precisão: Evolução das articulações - Pose 5

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	-100	18.9	-120	-121.1	-1.1	1.1
2	-98.9	17.8	-120	-121.1	-2.2	2.2
3	-97.8	16.7	-120	-121.1	-3.3	3.3
4	-96.7	15.6	-120	-121.1	-4.4	4.4
5	-95.6	14.4	-120	-121.1	-5.6	5.6
6	-94.4	13.3	-120	-121.1	-6.7	6.7
7	-93.3	12.2	-120	-121.1	-7.8	6.7
8	-92.2	11.1	-120	-121.1	-7.8	6.7
9	-91.1	11.11	-120	-121.1	-7.8	6.7
10	-90	11.1	-120	-121.1	-7.8	6.7
11	-88.9	11.1	-120	-121.1	-7.8	6.7
12	-87.8	11.1	-120	-121.1	-7.8	6.7

O algoritmo implementado atinge uma configuração final para a pose 5 com um erro total de 2.7º. Mais ainda a posição atingida pelo algoritmo é a posição (-0.199, 0.217, 0.448), que se traduz num erro de menos de 4mm.

Finalmente para a última pose escolhida, pose 6, aplicando a cinemática inversa obtém-se a configuração (178.2, -48.9, -20.4, 74.7, -89.3, 139.6). Para seguir a evolução ver tabelas 6.11 e 6.12.

O duplo planeador A* implementado atinge uma configuração final para a pose 6 com um erro total de 3.9º. Mais ainda a posição atingida pelo algoritmo é a posição (0.317, 0.097, 0.696), que se traduz num erro de menos de 14mm.

A questão dos obstáculos também foi testada. Para isso foi reutilizada a pose 3. Como já referido o planeamento é baseado numa estrutura com um índice associado a cada configuração possível. Por índice entende-se o valor identificativo de uma determinada configuração. Assim, para simular um objeto basta colocar esse índice como fazendo parte de um obstáculo. Outro aspeto é que o duplo planeamento é independente, ou seja, o planeamento de aproximação não vai afetar o de precisão no caso de inserção de um obstáculo. Nas tabelas 6.13 e 6.14 tem-se o planeamento da pose 3 traduzido em índices.

Tabela 6.11: Fase de Aproximação: Evolução das articulações - Pose 6

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	-140	-60	-100	-60
2	20	-20	-120	-40	-100	-40
3	40	-40	-100	-20	-100	-20
4	60	-60	-80	0	-100	0
5	80	-60	-60	20	-100	20
6	100	-60	-40	40	-100	40
7	120	-60	-40	60	-100	60
8	140	-60	-40	60	-100	80
9	160	-60	-40	60	-100	100
10	160	-60	-40	60	-100	120
11	160	-40	-20	80	-80	140

Tabela 6.12: Fase de Precisão: Evolução das articulações - Pose 6

Iteração	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	160	-42.2	-21.1	77.8	-82.2	138.9
2	161.1	-43.3	-21.1	76.7	-83.3	138.9
3	162.2	-44.4	-21.1	75.6	-84.4	138.9
4	163.3	-45.6	-21.1	74.4	-85.6	138.9
5	164.4	-46.7	-21.1	74.4	-86.7	138.9
6	165.6	-47.8	-21.1	74.4	-87.8	138.9
7	166.7	-48.9	-21.1	74.4	-88.9	138.9
8	167.8	-50	-21.1	74.4	-90	138.9
9	168.9	-50	-21.1	74.4	-90	138.9
10	170	-50	-21.1	74.4	-90	138.9
11	171.1	-50	-21.1	74.4	-90	138.9
12	172.2	-50	-21.1	74.4	-90	138.9
13	173.3	-50	-21.1	74.4	-90	138.9
14	174.4	-50	-21.1	74.4	-90	138.9
15	175.6	-50	-21.1	74.4	-90	138.9
16	176.7	-50	-21.1	74.4	-90	138.9
17	177.8	-50	-21.1	74.4	-90	138.9

Tabela 6.13: Fase de Aproximação: Evolução dos índices (sem obstáculos) - Pose 3

Iteração	Índice	Configuração
1	17964258	[0 ; 0 ; -140 ; -80 ; -100 ; -60]
2	19754340	[20 ; -20 ; -120 ; -100 ; -120 ; -60]
3	19759830	[20 ; -20 ; -100 ; -120 ; -140 ; -60]
4	19765662	[20 ; -20 ; -80 ; -120 ; -140 ; -60]

Tabela 6.14: Fase de Precisão: Evolução dos índices (sem obstáculos) - Pose 3

Iteração	Índice	Configuração
1	1988369	[21.1 ; -20 ; -62.2 ; -102.2 ; -122.2 ; -41.1]
2	3976738	[22.2 ; -18.9 ; -63.3 ; -103.3 ; -123.3 ; -42.2]
3	5965107	[23.3 ; -17.8 ; -64.4 ; -104.4 ; -124.4 ; -43.3]
4	7953494	[24.4 ; -16.7 ; -65.6 ; -105.6 ; -124.4 ; -44.4]
5	9941882	[25.6 ; -15.6 ; -66.7 ; -106.7 ; -124.4 ; -44.4]
6	11936102	[26.7 ; -14.4 ; -66.7 ; -107.8 ; -124.4 ; -44.4]
7	13930322	[27.8 ; -13.3 ; -66.7 ; -108.9 ; -124.4 ; -44.4]
8	15924542	[28.9 ; -12.2 ; -66.7 ; -110 ; -124.4 ; -44.4]
9	17919086	[30 ; -11.1 ; -66.7 ; -110 ; -124.4 ; -44.4]
10	19808654	[31.1 ; -11.1 ; -66.7 ; -110 ; -124.4 ; -44.4]

Para testar unicamente a capacidade de desviar de obstáculos definiu-se que o índice 19754340 no algoritmo de aproximação é considerado um obstáculo e recalculou-se a trajetória tendo-se obtido os resultados de aproximação presentes na tabela 6.15.

Tabela 6.15: Fase de Aproximação: Evolução dos índices (com um obstáculo) - Pose 3

Iteração	Índice	Configuração
1	17964258	[0 ; 0 ; -140 ; -60 ; -100 ; -60]
2	19754664	[20 ; -20 ; -120 ; -80 ; -120 ; -60]
3	19760154	[20 ; -20 ; -100 ; -100 ; -140 ; -60]
4	19765662	[20 ; -20 ; -80 ; -120 ; -140 ; -60]

Na tabela 6.15 pode-se reparar que o algoritmo encontrou uma trajetória que conecta o par de pontos origem-destino assim como reestruturou o restante trajeto de acordo com a alteração. A fase de precisão ficou inalterada neste teste, como seria expectável. Tendo testado esta hipótese colocou-se a hipótese de inserir erros em ambos as fases, tendo-se inserido dois erros em cada, os índices 19754340 e 19760154 na fase de aproximação (este último advinha da primeira tentativa de correção) e os índices 3976738 e 5965125 na fase de precisão tendo-se obtido os resultados presentes nas tabelas 6.16 e 6.17.

Tabela 6.16: Fase de Aproximação: Evolução dos índices (com vários obstáculos) - Pose 3

Iteração	Índice	Configuração
1	17964582	[0 ; 0 ; -140 ; -60 ; -100 ; -60]
2	19754664	[20 ; -20 ; -120 ; -80 ; -120 ; -60]
3	17870586	[0 ; -20 ; -100 ; -100 ; -140 ; -60]
4	19765662	[20 ; -20 ; -80 ; -120 ; -140 ; -60]

Tabela 6.17: Fase de Precisão: Evolução dos índices (com vários obstáculos) - Pose 3

Iteração	Índice	Configuração
1	1988369	[21.1 ; -20 ; -62.2 ; -102.2 ; -122.2 ; -41.1]
2	3976738	[22.2 ; -18.9 ; -63.3 ; -103.3 ; -122.2 ; -42.2]
3	5965107	[23.3 ; -17.8 ; -64.4 ; -104.4 ; -123.3 ; -42.2]
4	7953494	[24.4 ; -16.7 ; -65.6 ; -105.6 ; -124.4 ; -43.3]
5	9941882	[25.6 ; -15.6 ; -66.7 ; -106.7 ; -124.4 ; -44.4]
6	11936102	[26.7 ; -14.4 ; -66.7 ; -107.8 ; -124.4 ; -44.4]
7	13930322	[27.8 ; -13.3 ; -66.7 ; -108.9 ; -124.4 ; -44.4]
8	15924542	[28.9 ; -12.2 ; -66.7 ; -110 ; -124.4 ; -44.4]
9	17919086	[30 ; -11.1 ; -66.7 ; -110 ; -124.4 ; -44.4]
10	19808654	[31.1 ; -11.1 ; -66.7 ; -110 ; -124.4 ; -44.4]

Analisando as tabelas 6.16 e 6.17 é possível comprovar que o algoritmo é robusto com a inserção de obstáculos tendo-se desviado dos mesmos e adaptando o restante percurso. Foram realizados mais testes semelhantes tendo todos eles mostrado resultados semelhantes. Um dos quais que se torna fácil de compreender e visualizar num ambiente tridimensional é o teste de rotação do robot sobre um eixo. Para tal foi definido um conjunto de pontos de forma a que o robot passando por eles dê-se uma volta sobre si próprio. Seguidamente, colocou-se como obstáculo algum desses pontos e analisou-se a resposta do planeador (ver figura 6.1).

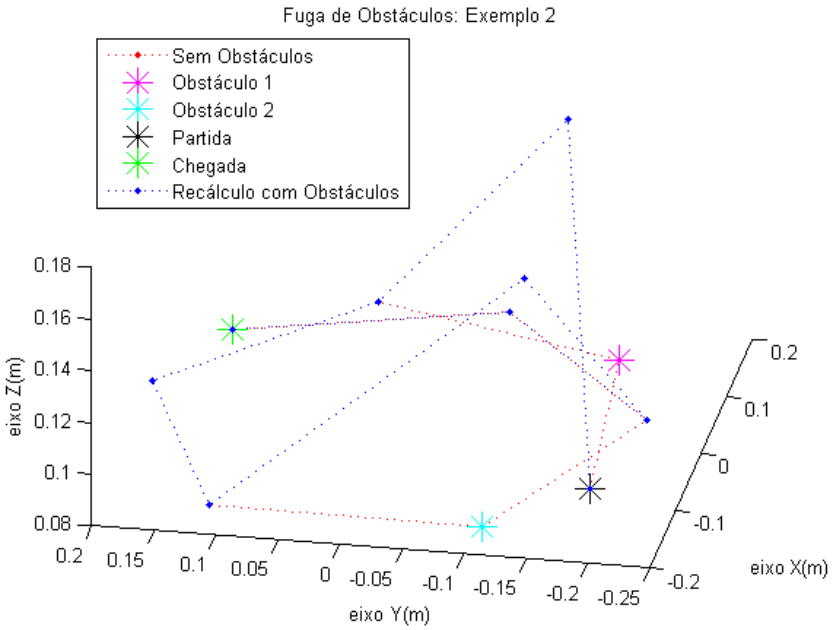

Figura 6.1: Visualização do Desvio de Obstáculos

6.2 Comparação do Planeador com *MoveIt!*

A solução *MoveIt* recorre a planeadores OMPL para definir trajetórias para cada robot já inserido na estrutura *MoveIt*. OMPL (Open Motion Planning Library) é uma biblioteca com vários algoritmos atuais para a determinação de trajetórias. Para o caso do robot Universal 5 já existe um planeador pré-definido, o algoritmo *Kinodynamic Planning by Interior-Exterior Cell Exploration* (KPIECE). Este algoritmo é um planeador baseado em árvore que usa a discretização para conduzir à exploração do espaço contínuo. Esta discretização é simplificada pela aplicação de uma única grelha espacial. Isto é conseguido pela projeção do espaço segundo essa grelha. Esta implementação é direcionada para sistema de restrições referenciais. Para uniformizar os resultados com as experiências anteriores, escolheu-se poses já anteriormente testadas. No caso, selecionaram-se 4 poses com posições (x,y) de sinais variáveis, ou seja, ambos positivos, ambos negativos ou de sinais contrários. Para esse efeito optou-se pelas poses 1, 2, 4 e 5. De seguida apresentar-se-à os gráficos 3D comparativos das posições alcançadas por cada um dos planeadores (duplo A* e planeador OMPL do *MoveIt*) e ainda o número de passos, a distância percorrida e o tempo de execução até à posição final de cada um bem como a posição exata alcançada.

Para a pose 1 constatar na figura 6.2 e na tabela 6.18, para a pose 2 verificar na figura 6.3 e na tabela 6.19, para a pose 4 visualizar na figura 6.4 e na tabela 6.20 e finalmente para a pose 5 reparar na figura 6.5 e na tabela 6.21.

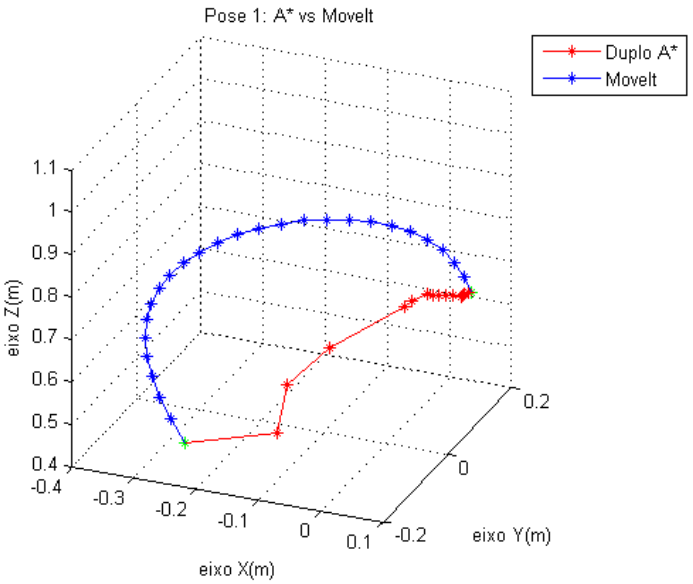

Figura 6.2: Comparativo MoveIt vs Duplo A* - Pose 1

Tabela 6.18: Comparativo *MoveIt* vs Duplo A*): Número de Iterações - Pose 1

Algoritmo	Passos	Distância (m)	Tempo de Execução (s)
MoveIt	28	1.1328	5.894
Duplo A*	11+9 = 20	0.8095	4.2775

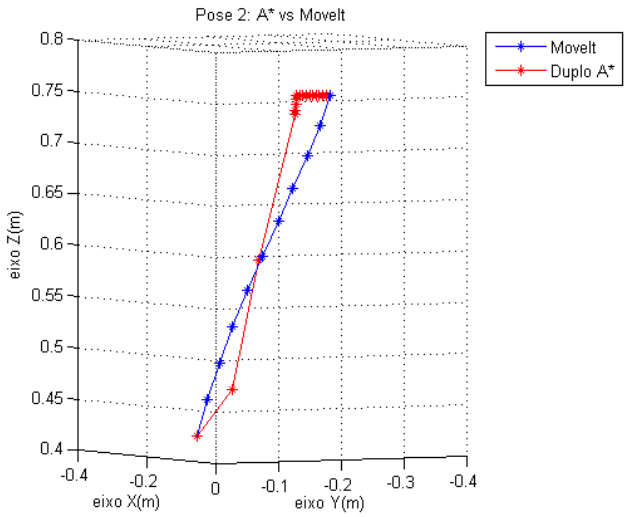

Figura 6.3: Comparativo MoveIt vs Duplo A* - Pose 2

Tabela 6.19: Comparativo *MoveIt* vs Duplo A*): Número de Iterações - Pose 2

Algoritmo	Passos	Distância (m)	Tempo de Execução (s)
MoveIt	11	0.3996	2.228
Duplo A*	4+14 = 18	0.7205	3.8325

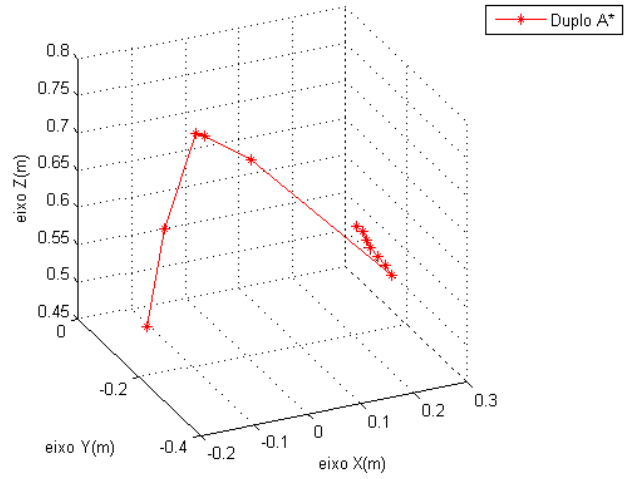

Figura 6.4: Comparativo MoveIt vs Duplo A* - Pose 4

Tabela 6.20: Comparativo *MoveIt* vs Duplo A*): Número de Iterações - Pose 4

Algoritmo	Passos	Distância (m)	Tempo de Execução (s)
MoveIt	—	—	—
Duplo A*	11+7 = 18	0.8651	4.5555

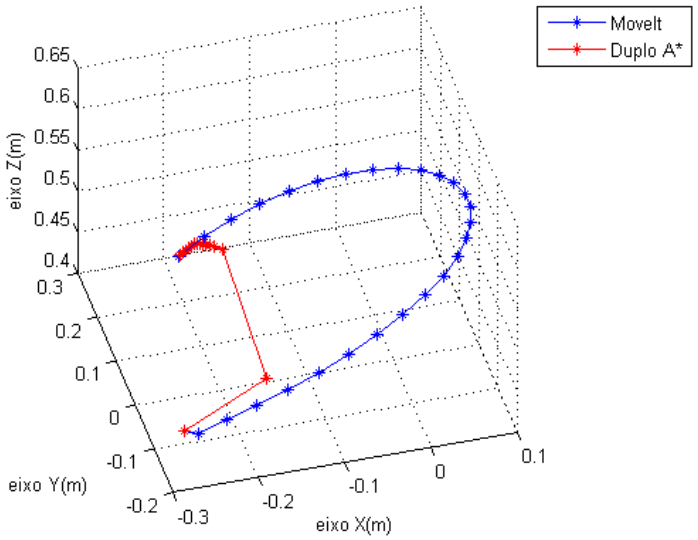

Figura 6.5: Comparativo MoveIt vs Duplo A* - Pose 5

Tabela 6.21: Comparativo *MoveIt* vs Duplo A*): Número de Iterações - Pose 5

Algoritmo	Passos	Distância (m)	Tempo de Execução (s)
MoveIt	30	0.9438	4.949
Duplo A*	6+12 = 18	0.6178	3.319

Analisando rapidamente os resultados produzidos pela duplo A* e pela *MoveIt* tem-se que na grande generalidade dos casos o planeador duplo A* produz melhores resultados de aproximação com distâncias percorridas menores. Outro aspeto prende-se com o facto do duplo A* ter encontrado sempre trajetória para todos os casos, ao par que o *MoveIt* não o conseguiu para a pose 4. Finalmente, e com igual importância está o facto de o número de passos que o duplo A* necessita para alcançar o destino final é também substancialmente menor em grande parte dos casos.

6.3 EuRoC

Ao longo do projeto EuRoC conseguiu-se alguns resultados importantes. Para além da completação do projeto, conseguiu-se desenvolver algoritmos genéricos e soluções suficientemente flexíveis para serem aplicadas numa grande parte de problemas robóticas. As soluções implementadas e testadas foram então:

- Reconhecimento de ambientes estáticos e objetos estáticos e dinâmicos;
  - Remoção de fundo;
  - Extração de características de objetos usando sistemas RGBD;
  - Criação de Classe de Objetos;

  - Identificação de objetos lineares (cubos, cilindros, pegas e aglomerados destes) com base na classe de objetos e na aplicação da árvore de decisão.

- Movimentação controlada do robot e actuação sobre os objetos detetados;
  - Cinemática Inversa de um robot ABB *lightweight*;
  - Aplicação do planeador Duplo A*;
  - Comando do robot por intermédio de controlo de articulações.

- Algoritmo de Controlo e Planeamento de Tarefas;
  - Ciclo de controlo com Feedback, baseado em processamento de imagem.
  - Finalização de operações de *pick-and-place* com objetos elementares, composição de puzzles e desvio de obstáculos, nomeadamente paredes.
  - Correção de erros inseridos por movimentação do robot ABB e dos sistemas RGBD.
  - Adaptação a diferentes condições de trabalho (mesa com objetos estáticos e tapete rolante com objetos dinâmicos).

De referir ainda que as soluções desenvolvidas foram também validadas pela comunidade robótica e científica no sentido que com este projeto, conseguiu-se a aprovação à fase seguinte do EuRoC, onde serão desenvolvidos projetos direcionados a vários ramos da indústria.

6.4 Amazon Picking Challenge

Ao longo do projeto Amazon Picking Challenge conseguiu-se outros resultados importantes. O desafio foi terminado, mas muitos dos algoritmos desenvolvidos foram já inseridos na comunidade robótica (nomeadamente por intermédio do *software* ROS) devido à sua relevância para as aplicações robóticas atuais, nomeadamente aplicações de *pick-and-place* em geral e de visão, manipulação e cálculo de trajetória em particular.

- Reconhecimento de ambientes estáticos e objetos estáticos e dinâmicos;
  - Deteção da prateleira e dos diversos objetos utilizando técnicas de processamento de imagem em ficheiros *PointCloud*;
  - Com base na deteção da prateleira, processamento da calibração de referenciais;
  - Extração de características de objetos usando sistemas RGBD;
  - Criação de Classe de Objetos;
  - Identificação de objetos não-estruturados com base na classe de objetos e na aplicação da árvore de decisão.

- Movimentação controlada do robot e actuação sobre os objetos detetados;

  - Cinemática Inversa do robot UR5;
  - Aplicação do planeador Duplo A*;
  - Comando do robot por intermédio de controlo de articulações;
  - Implementação de uma estação de ferramentas (com mão ROBOTIQ, ventosas, pinças pneumáticas e VERSABALL) com possibilidade de troca entre as ferramentas existentes.

- Algoritmo de Controlo e Planeamento de Tarefas;
  - Pré-planeamento das tarefas com atualização e auto-ajuste sempre que se visualiza a estante para a qual o trabalho é direcionada.

6.5 Resumo do Capítulo

O capítulo 6 apresenta os resultados encontrados ao longo do trabalho realizado.

Assim, neste capítulo foram apresentadas as respostas do algoritmo de planeamento desenvolvido e os erros por ele produzidos. Consegue-se comprovar no entanto que este algoritmo de planeamento é eficaz no sentido que com um erro pequeno atinge a pose desejada.

Mais ainda, este algoritmo é comparado com um planeador já existente (o planeador pré-definido pelo *MoveIt*), obtendo-se resultados auspiciosos.

Finalmente nas duas últimas secções do capítulo são descritas as conclusões apresentadas nos dois desafios já anteriormente referidos.

Capítulo 7

Conclusões

Ao longo do projeto apresentado, foram desenvolvidas várias soluções que podem ser valorizados pela comunidade científica e tecnológica, mais particularmente pela comunidade robótica.

A questão da cinemática é um problema comum em vários desafios atuais e futuros que evolvam compreender o movimento de um robot genérico. Ao longo desta tese, o robot de eleição para testes de algoritmos foi o robot Universal 5. Para este robot foi determinada a interferência de cada articulação no movimento do robot e conseguiram-se definir equações para compreender o estado do robot a nível de articulação e cartesiano a cada momento.

Mais ainda, foi proposta uma arquitetura genérica e flexível baseada em três níveis operacionais:

1. Reconhecimento / Percepção / Sensorização;
2. Movimento / Manipulação / Uso da Ferramenta;
3. Controlo / Questões Adaptativas.

Esta abordagem provou ser flexível, assim como as estratégias associadas a cada nível provaram ser essenciais para a aplicação futura. Nomeadamente, é vantajoso a utilização de uma árvore de decisão sobre as características de objetos encontrados numa cena de forma a estes serem comparados com objetos já presentes numa base de dados. Esta base de dados sugere que a sua criação e manutenção funcione como um processo de *data mining*. Todo este processo deve ser acompanhado por tolerâncias, de forma a inserir alguma flexibilidade ao projeto.

Outro aspeto a considerar quando se analisa estes níveis operacionais e as estratégias por eles escondidas é o planeador de trajetória. O objetivo primordial deste projeto era o desenvolvimento de um planeador capaz de competir com os algoritmo de estado-da-arte atualmente usados. Para este fim foi desenvolvido um planeador Duplo A*.

Este algoritmo provou ter resultados muito interessantes, na medida que rivaliza, conseguindo na grande generalidade dos casos superiorizar-se aos algoritmos comummente utilizados tanto na indústria como na área de investigação. Além de ter tempos de execução reduzidos, o número de passos para encontrar a solução final é também inferior assim como a distância do trajeto calculado

quando comparada com outros algoritmos presentes no mercado. Além disso a possibilidade de dividir o percurso em duas fases permite acelerar o processo de aproximação, refinando o mesmo movimento nos instantes finais de modo a se atingir a pose (posição + orientação) final de forma rápida e precisa.

Adicionalmente foram também apresentados dois exemplos de aplicação: o EuRoC e o Amazon Picking Challenge. Para cada um destes casos foram construídas soluções com base na arquitetura proposta e com o planeador desenvolvido. Todos os resultados foram positivos e validados pela comunidade, sendo que grande parte dos algoritmos desenvolvidos é atualmente partilhado em algumas aplicações para o mundo industrial.

O trabalho apresentado já foi alvo de produção de artigos de interesse científico. Atualmente existem já dois artigos publicados baseados na estratégia apresentada sendo que existem ainda mais três à espera de aceitação.

Em conclusão, o trabalho apresentado formaliza uma abordagem às operações de *pick-and-place*. A abordagem proposta pode ser valorizada em vários campos da robótica em virtude da sua flexibilidade e pode ser aplicável a um grande leque de aplicações. Mais ainda, o algoritmo de planeamento de trajetória demonstra vários resultados interessante que sugerem a sua utilização frequentemente para o mesmo leque de aplicações.

7.1 Trabalho Futuro

O trabalho apresentado tem alguns campos de melhoria. Primeiramente, o algoritmo desenvolvido mostra bons resultados a nível de encontrar trajetória para várias poses com um erro mínimo. No entanto a fase de aproximação produz trajetórias um pouco repentinas e demasiado diretas. Isto deve-se ao facto do espaço de configurações ser demasiado amplo e uma iteração corretiva de uma única célula produzir variações cartesianas demasiado acentuadas.

Assim, futuras melhorias passam por cruzar os resultados obtidos com um algoritmo de suavização a nível do espaço cartesiano.

Mais ainda, toda a arquitetura e o planeador em específico foram testados em aplicações específicas (EuRoC e Amazon), no entanto seria vantajoso aplicar a estratégia a tarefas repetitivas e de grande exigência.

Bibliografia

[1] M. Hvilshoj and S. Bogh, ""little helper- an autonomous industrial mobile manipulator concept," *International Journal of Advanced Robotic Systems*, vol. 8, no. 2, pp. 80–90, 2011.

[2] J.-h. Kim, "Automated medicine storage and medicine introduction/discharge management system," Oct. 9 2012, uS Patent 8,281,553.

[3] H. P. Chen, J. J. Wang, G. Zhang, T. Fuhlbrigge, and S. Kock, "High-precision assembly automation based on robot compliance," *International Journal of Advanced Manufacturing Technology*, vol. 45, no. 9-10, pp. 999–1006, 2009.

[4] J. Khurshid, H. Bing-Rong, and Ieee, *Military robots - A glimpse from today and tomorrow*, ser. 2004 8th International Conference on Control, Automation, Robotics and Vision, Vols 1-3. New York: Ieee, 2004.

[5] I. F. o. Robotics, "World robotics 2014 industrial robots," Report, 2014.

[6] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, Conference Proceedings, pp. 1–8.

[7] P. Westhead, M. Wright, and D. Ucbasaran, "The internationalization of new and small firms: A resource-based view," *Journal of Business Venturing*, vol. 16, no. 4, pp. 333–358, 2001.

[8] H. Mikael, H. Erik, and J. Mats, "Robotics for smÉs - investigating a mobile, flexible, and reconfigurable robot solution," in *39th International Symposium on Robotics, ISR 2008*, Conference Proceedings, pp. 56–61.

[9] K. DeMarco, M. E. West, T. R. Collins, and Ieee, "An implementation of ros on the yellowfin autonomous underwater vehicle (auv)," *Oceans 2011*, p. 7, 2011.

[10] S. Cousins, "Ros on the pr2," *Ieee Robotics & Automation Magazine*, vol. 17, no. 3, pp. 23–25, 2010.

[11] K. Tsuchiya, S. Kagami, W. Yoshizaki, H. Mizoguchi, and Ieee, "Grasp planning precomputation by considering center of gravity of objects and its evaluation using openrave," *Proceedings 2012 Ieee International Conference on Systems, Man, and Cybernetics (Smc)*, pp. 2091–2096, 2012.

[12] O. S. R. Foundation, "About ros," 2015. [Online]. Disponível: http://www.ros.org/ [Acedido: 16-06-2015]

[13] OpenRAVE, "Welcome to open robotics automation virtual environment," 2015. [Online]. Disponível: http://openrave.org/ [Acedido: 16-06-2015]

[14] M. Magazine, "An automatic block-setting crane," 1938.

[15] I. F. o. Robotics, "History of industrial robots," 2012.

[16] P. Y. Chua, T. Ilschner, and D. G. Caldwell, "Robotic manipulation of food products - a review," *Industrial Robot-an International Journal*, vol. 30, no. 4, pp. 345–354, 2003.

[17] X. Jiang, K. M. Koo, K. Kikuchi, A. Konno, M. Uchiyama, and Ieee, "Robotized assembly of a wire harness in car production line," *Ieee/Rsj 2010 International Conference on Intelligent Robots and Systems (Iros 2010)*, pp. 490–495, 2010.

[18] R. Mattone, G. Campagiorni, and F. Galati, "Sorting of items on a moving conveyor belt. part i. a technique for detecting and classifying objects," *Robotics and Computer-Integrated Manufacturing*, vol. 16, no. 2-3, pp. 73–80, 2000.

[19] R. Mattone, M. Divona, and A. Wolf, "Sorting of items on a moving conveyor belt. part 2: performance evaluation and optimization of pick-and-place operations," *Robotics and Computer-Integrated Manufacturing*, vol. 16, no. 2-3, pp. 81–90, 2000.

[20] T. Gecks, D. Henrich, and Ieee, *Human-robot cooperation: Safe pick-and-place operations*, ser. 2005 IEEE International Workshop on Robot and Human Interactive Communication. New York: Ieee, 2005.

[21] S. Daoud, H. Chehade, F. Yalaoui, and L. Amodeo, "Efficient metaheuristics for pick and place robotic systems optimization," *Journal of Intelligent Manufacturing*, vol. 25, no. 1, pp. 27–41, 2014.

[22] L. D. Wang, L. Ren, J. K. Mills, and W. L. Cleghorn, "Automated 3-d micrograsping tasks performed by vision-based control," *Ieee Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 417–426, 2010.

[23] C. M. Son, "Optimal control planning strategies with fuzzy entropy and sensor fusion for robotic part assembly tasks," *International Journal of Machine Tools & Manufacture*, vol. 42, no. 12, pp. 1335–1344, 2002.

[24] C. Son, "Intelligent robotic path finding methodologies with fuzzy/crisp entropies and learning," *International Journal of Robotics & Automation*, vol. 26, no. 3, pp. 323–336, 2011.

[25] O. Ghita and P. F. Whelan, "A bin picking system based on depth from defocus," *Machine Vision and Applications*, vol. 13, no. 4, pp. 234–244, 2003.

[26] L. Zhibin, L. Yunjiang, L. Zexiang, Y. Guilin, and G. Jian, "T2: A novel two degree-of-freedom translational parallel robot for pick-and-place operation," in *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, Conference Proceedings, pp. 725–730.

[27] I. Bonec, "Delta parallel robot - the story of success," 2015.

[28] V. Nabat, M. de la O Rodriguez, O. Company, S. Krut, and F. Pierrot, "Par4: very high speed parallel robot for pick-and-place," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Conference Proceedings, pp. 553–558.

[29] Mecademic, "Dextar," 2014.

[30] D. Bristol, "Sketchy," 2011.

[31] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Hoboken, NJ: John Wiley & Sons, 2006.

[32] EPSON, "Epson g20," 2015. [Online]. Disponível: http://robots.epson.com/product-detail/5 [Acedido: 16-06-2015]

[33] TAPAS, "Welcome to tapas," 2015.

[34] N. Boubekri and P. Chakraborty, "Robotic grasping: gripper designs, control methods and grasp configurations – a review of research," *Integrated Manufacturing Systems*, vol. 13, no. 7, pp. 520–531, 2002.

[35] ROBOTIQ, "Robotiq products," 2015.

[36] E. Robotics, "Versaball," 2015. [Online]. Disponível: http://empirerobotics.com/#about [Acedido: 16-06-2015]

[37] L. F. Rocha, G. Veiga, M. Ferreira, A. P. Moreira, and V. Santos, "Increasing flexibility in footwear industrial cells," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, Conference Proceedings, pp. 291–296.

[38] o. Barreto, "Wide area multiple camera calibration and estimation of radial distortion," 2004.

[39] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, Jo, #227, and o. Barreto, "Camera models and fundamental concepts used in geometric computer vision," *Found. Trends. Comput. Graph. Vis.*, vol. 6, no. 1–2, pp. 1–183, 2011.

[40] Intel, "Rgb-d: Techniques and usages for kinect style depth cameras," 2015. [Online]. Disponível: http://ils.intel-research.net/projects/rgbd [Acedido: 16-06-2015]

[41] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from rgbd images," in *Proceedings - IEEE International Conference on Robotics and Automation*, Conference Proceedings, pp. 842–849.

[42] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," pp. 746–760, 2012.

[43] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an rgbd camera," *ACM Transactions on Graphics*, vol. 31, no. 6, 2012.

[44] E. Ackerman, "Top 10 robotic kinect hacks," 2011. [Online]. Disponível: http://spectrum.ieee.org/automaton/robotics/diy/top-10-robotic-kinect-hacks [Acedido: 16-06-2016]

[45] F. Belkhouche, "Reactive path planning in a dynamic environment," *Robotics, IEEE Transactions on*, vol. 25, no. 4, pp. 902–911, 2009.

[46] Y. K. Hwang and N. Ahuja, "Gross motion planning—a survey," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 219–291, 1992.

[47] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[48] J. van den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Conference Proceedings, pp. 4253–4258.

[49] Y. Yang and O. Brock, "Elastic roadmaps–motion generation for autonomous mobile manipulation," *Auton. Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[50] J. Hing, "Breadth-first and depth-first search for path planning," 2015. [Online]. Disponível: http://dasl.mem.drexel.edu/Hing/BFSDFSTutorial.htm [Acedido: 16-06-2015]

[51] W. Huijuan, Y. Yuan, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on*, Conference Proceedings, pp. 1067–1069.

[52] L. Lamport, "A new solution of dijkstra's concurrent programming problem," *Commun. ACM*, vol. 17, no. 8, pp. 453–455, 1974.

[53] N. Mouly and J. P. Merlet, "Singular configurations and direct kinematics of a new parallel manipulator," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, Conference Proceedings, pp. 338–343 vol.1.

[54] D. Oetomo, L. Hwee Choo, G. Alici, and B. Shirinzadeh, "Direct kinematics and analytical solution to 3rrr parallel planar mechanisms," in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, Conference Proceedings, pp. 1–6.

[55] J. P. Merlet, "Direct kinematics of parallel manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 6, pp. 842–846, 1993.

[56] J. Denavit and R. S. Hartenberg, *A kinematic notation for lower pair mechanisms based on matrices*.

[57] R. Keating, "Ur5 inverse kinematics," Report.

[58] T. Pinho, A. P. Moreira, and J. Boaventura-Cunha, "Framework using ros and simtwo simulator for realistic test of mobile robot controllers," *Controlo'2014 - Proceedings of the 11th Portuguese Conference on Automatic Control*, vol. 321, pp. 751–759, 2015.

[59] Q. Creator, "Qt creator," 2015. [Online]. Disponível: http://qt-project.org/wiki/Category:Tools::QtCreator [Acedido: 16-06-2015]

[60] "Moveit!" 2015. [Online]. Disponível: http://moveit.ros.org/ [Acedido: 16-06-2015]

[61] O. S. R. Foundation, "Why gazebo?" 2014. [Online]. Disponível: http://gazebosim.org/ [Acedido: 16-06-2015]

[62] Itseez, "Opencv," 2015. [Online]. Disponível: http://opencv.org/ [Acedido: 16-06-2015]