ESTRATÉGIAS PARA FACILITAR A OPERAÇÃO DE BRAÇOS ROBÓTICOS EM AMBIENTES NÃO-ESTRUTURADOS

Edwin Francis Cárdenas Correa

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Mecânica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Mecânica.

Orientador: Max Suell Dutra

Rio de Janeiro
Abril de 2015

ESTRATÉGIAS PARA FACILITAR A OPERAÇÃO DE BRAÇOS ROBÓTICOS EM AMBIENTES NÃO-ESTRUTURADOS

Edwin Francis Cárdenas Correa

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA MECÂNICA.

Examinada por:

Prof. Max Suell Dutra, Dr.-Ing.

Prof. Fernando Augusto de Noronha Castro Pinto, Dr.-Ing.

Prof. Marco Antonio Meggiolaro, Ph.D

Prof. Luciano Santos Constantin Raptopoulos, D.Sc.

Prof. Gerson Gomes Cunha, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2015

Correa, Edwin Francis Cárdenas

Estratégias para facilitar a operação de braços robóticos em ambientes não-estruturados /Edwin Francis Cárdenas Correa. – Rio de Janeiro: UFRJ/COPPE, 2015.

XXI, 203 p.: il.; 29, 7cm.

Orientador: Max Suell Dutra

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Mecânica, 2015.

Referências Bibliográficas: p. 171 – 192.

1. Robótica. 2. Realidade Aumentada. 3. Planejamento de Movimentos. 4. Geração de Rotas Livre de Colisões. I. Dutra, Max Suell. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Mecânica. III. Título.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ESTRATÉGIAS PARA FACILITAR A OPERAÇÃO DE BRAÇOS ROBÓTICOS EM AMBIENTES NÃO-ESTRUTURADOS

Edwin Francis Cárdenas Correa

Abril/2015

Orientador: Max Suell Dutra

Programa: Engenharia Mecânica

O presente trabalho concentra-se no desenvolvimento de estratégias que auxiliem aos pilotos no controle de manipuladores robóticos teleoperados em ambientes não-estruturados. Para exemplificar as estratégias optou-se pelo estudo de caso do manipulador TITAN-4, o qual é geralmente encontrado nos veículos submarinos operados remotamente, conhecidos como ROVs. Estes sistemas robóticos contam com câmeras que transmitem imagens em tempo real, fornecendo ao piloto informações visuais para decidir como controlar o manipulador.

As dificuldades deste tipo de teleoperação são apresentadas nesta tese. Em consequência disso, duas tecnologias recentes foram estudadas e analisadas: a Realidade Aumentada (AR) e geração de rotas livres de colisões. Com AR é possível acrescentar informações visuais e simultaneamente dotar de informações numéricas as estratégias de geração de rotas para antecipar os movimentos dos manipuladores.

Como resultado da pesquisa foram apresentadas metodologias para criação de software AR e suas possibilidades na engenharia. Também foram propostos novos algoritmos baseados no método denominado Árvores Aleatórias para Exploração Rápida (RRTs) e assim possibilitar movimentos autônomos dos braços robóticos que operem em ambientes não estruturados.

Além disto, foi desenvolvido um software que mostra como combinar as duas tecnologias usando conceitos de *Human-In-The-Loop* (HITL). Um aspecto pouco estudado na literatura é a maneira como os humanos podem acompanhar rotas geradas automaticamente. Portanto, foi especificado um contexto deste tema para o estudo de caso e é proposto um novo algoritmo capaz de gerar rotas que são facilmente replicadas pelos pilotos. Diversas simulações ilustram o desempenho e viabilidade das técnicas desenvolvidas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

STRATEGIES TO FACILITATE THE OPERATION OF ROBOTIC ARMS IN UNSTRUCTURED ENVIRONMENTS

Edwin Francis Cárdenas Correa

April/2015

Advisor: Max Suell Dutra

Department: Mechanical Engineering

This work is concerned on developing strategies that help the human to control robotic manipulators inside of unstructured environments. To illustrate these strategies was chosen the case study of the TITAN-4 robot, which is commonly mounted on remotely operated underwater vehicles (ROVs). These robotic systems have cameras that transmit images in real time, providing visual information to the pilot for deciding how to operate remotely the manipulator.

The thesis shows the difficulties of this type of teleoperation. To help overcome these difficulties, two recent technologies have been studied and analyzed: Augmented Reality (AR) and the free-collision paths generation. With AR, it is possible to insert visual information on the screen and simultaneously gives numerical information to the automatic path planner to anticipate the movements of the manipulator.

The methodology to constitute an AR software and its potential applications in the engineering were presented as research results. It was also proposed new strategies based on a Rapidly-exploring Random Tree algorithm (RRT) to solve main drawbacks for the autonomous movements of the robotic arms.

Another advancement of the present study was the development of a software that shows how to combine the two technologies using the concepts of Human-In-The-Loop (HITL). How to follow programmed robot movements by humans is a topic often not considered in the literature. Therefore, an adequate context was provided, and it was also proposed a new algorithm capable of generating paths that are easily replicated by the pilots. Several simulations illustrate the performance and feasibility of the techniques developed.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathbf{q}$	vetor de configuração que representa ao robô, p. 28
$\mathcal{A}$	Definição de um robô no espaço de trabalho, p. 26
$\mathcal{C}$	Representação topológica do Espaço de Configuração, p. 28
$\mathcal{O}$	Obstáculo no espaço de trabalho, p. 26
$\mathcal{Q}$	Espaço de Configuração, p. 28
$\mathcal{W}$	Espaço de trabalho (*workspace*), p. 26
$\mathcal{Q}_{free}$	Espaço de configuração livre de colisões, p. 28
$\mathcal{Q}_{obs}$	Representação dos obstáculos no espaço de configuração, p. 28
q_i	uma variável de configuração, θ_i para junta rotacional e d_i para prismática, p. 28
SO(3)	Grupo especial ortonormal da ordem 3 (*special orthonormal group* 3), p. 19

Lista de Abreviaturas

AR	Realidade Aumentada (*Augmented Reality*), p. 18
AUV	Veículo submarino autónomo (*Autonomous Underwater Vehicle*), p. 1
DRRT-Con	Algoritmo de geração de rotas dinâmicas usando árvores aleatórias para exploração rápida (*Dynamic RRT-Connect*), p. 108
EF	Efetuador Final, p. 9
GDL	Graus De Liberdade, p. 9
GPM	Método de Projeção do Gradiente (*Gradient Projection Method*), p. 9
HID	Dispositivo de interface humana (*Human Interface Device*), p. 64
HITL	Inserção do homem no circuito de controle, (*Human-In-The-Loop*), p. 131
IBVS	Servovisão baseada na imagem (*Image-Based Visual Servoing*), p. 16
IKURT	Algoritmo de Cinemática Inversa Usando Árvores Aleatórias Para Exploração Rápida (*Inverse Kinematics Using Random Trees Exploration algorithm*), p. 92
PBVS	Servovisão baseada na posição (*Position-Based Visual Servoing*), p. 16
PRM	Mapa de Rotas Probabilístico, (*Probabilistic Roadmap Method*), p. 31
ROV	Veículo submarino operado remotamente (*Remotely Operated Vehicle*), p. 1

RRT-Con — Árvores Aleatórias para Exploração Rápida com Conexão bi-direcional, (*Rapidly-exploring Random Trees - Connect*), p. 35

RRT — Árvores Aleatórias para Exploração Rápida, (*Rapidly-exploring Random Trees*), p. 32

SISO — Uma entrada e uma saída, monovariável (*Single-Input-Single-Output*), p. 11

SQP — Programação Quadrática Sequencial (*Sequential Quadratic Programming*), p. 10

VRLM — *Virtual Reality Modeling Language*, p. 59

VR — Realidade virtual (*Virtual Reality*), p. 59

Capítulo 1

Introdução

Dos últimos 20 anos para o presente os pesquisadores conseguiram com sucesso controlar os robôs manipuladores industriais de base fixa, de maneira que as empresas de transformação têm se beneficiado da velocidade, precisão e confiabilidade nas tarefas realizadas por estas máquinas. Enquanto isso, na última década novos robôs surgiram em resposta às necessidades de outras áreas de atuação, tais como: medicina, entretenimento, inspeção de tubulações e aplicações militares.

Uma das promissoras áreas de pesquisa na robótica são os robôs que interagem em ambientes não-estruturados, onde os objetos não estão bem posicionados ou o seu ambiente é parcialmente conhecido [1]. Um robô é considerado como um sistema não linear acoplado de difícil controle devido às complexas dinâmicas envolvidas. Esta situação é exacerbada quando o ambiente é não-estruturado, neste caso são requeridas técnicas sofisticadas de percepção, estimação e controle inteligente [2].

Como no caso dos manipuladores de base fixa, espera-se no futuro próximo desenvolver estrategias para controlar eficientemente os robôs que trabalham em ambientes não-estruturados.

A Fig. 1.1 apresenta dois exemplos destes tipos de robôs da empresa iRobot. O robô móvel Roomba da Fig. 1.1(a) que limpa um ambiente usando algoritmos de exploração de maneira autônoma e na Fig. 1.1(b) o PackBot que é operado remotamente, seu braço robótico é utilizado em várias aplicações onde a segurança humana está envolvida. Na Fig. 1.1(b) pode ser observado a exploração feita na usina nuclear de Fukushima Daiichi apos do terremoto que atingiu o Japão em 2011.

Entre os robôs submetidos a ambientes não-estruturados tem-se os robôs submarinos AUVs (por suas siglas *Autonomous Underwater Vehicles*) e ROVs (das siglas *Remotely Operated Vehicles*), estes atualmente estão presentes em muitas atividades de exploração, operação, manutenção e pesquisa nos oceanos [3]. Este é o caso da indústria do petróleo onde os ROVs já fazem parte das operações offshore [4].

Em geral os ROVs de intervenção estão constituídos por uma base móvel e dois

(a) (b)

Figura 1.1: Exemplos de robôs da empresa iRobot que operam em ambientes não-estuturados. (a) Robô móvel de uso doméstico Roomba em tarefa de limpeza. (b) Robô PackBot de aplicações especias, operando na usina nuclear de Fukushima Daiichi - Japão. (Imagens disponíveis na internet).

braços robóticos (manipuladores). Devido às duras condições físicas das profundezas e dinâmicas envolvidas dos corpos flutuantes são requeridas pelo menos duas pessoas para controlar o sistema robótico; uma para manobrar a base e outra para operar os manipuladores. Em terra firme ou dentro de uma embarcação os operadores comandam e acompanham o robô através das imagens recebidas do sistema.

A experiência do piloto é um fator importante nas operações com ROVs, particularmente em áreas de forte corrente, o conhecimento das capacidades do veículo e suas limitações é essencial. Um bom operador terá desenvolvido um senso de percepção espacial e controle dos manipuladores após meses de prática [5]. O tempo e qualidade do trabalho realizado pelos operadores também é influenciado pela situação de confinamento além dos períodos que eles passam embarcados nos navios [6].

Entre os aspetos técnicos mais relevantes que explicam porque é dificil o controle dos manipuladores dos ROVs, tem-se [7]:

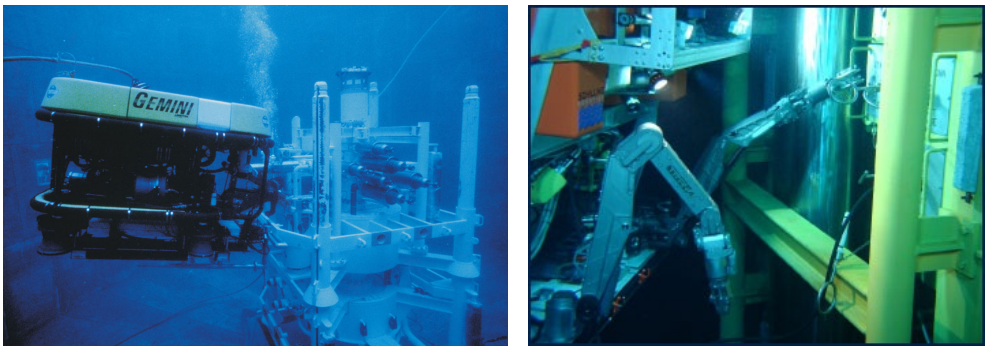
- Tentar controlar o sistema do veículo-manipulador, como um todo, é difícil face a geração de perturbações não lineares sobre a base do manipulador mesmo com variações lenta do veículo;
- Há uma interação entre as forças do movimento do manipulador que afeta a "atitude"do ROV, ou capacidade de localizar-se em relação ao seu ambiente;
- Os coeficientes hidrodinâmicos frequentemente não são conhecidos, a dinâmica do conjunto manipulador-veículo pode variar consideravelmente de acordo com a velocidade e direção de movimento do manipulador, bem como do veiculo;
- O acoplamento dinâmico, que surge a partir da transmissão de forças e mo-

mentos entre o manipulador e o veículo, varia de acordo com a gama de especificações das trajetória do manipulador;

- Outro fator existente são os sistemas de sensoriamento do manipulador, eles são pobres ou simplesmente não há um *feedback* sensorial ao operador do manipulador além das imagens obtidas das câmeras;

- Dos múltiplos vídeos obtidos das diferentes intervenções offshore usando ROV, pode se afirmar que em geral sempre tem-se objetos flutuantes em constante movimento dentro do espaço de trabalho do manipulador, mesmo que se estiver usando um braço como sistema de retenção em uma estrutura sólida para ajudar na estabilização da base do ROV ou na manipulação de estruturas (Fig. 1.2(b));

- Os manipuladores possuem atuadores elétricos e hidráulicos, devido às condições nas profundidades, os movimentos gerados são muitas vezes irregulares com um restrito controle de velocidade.



(a) (b)

Figura 1.2: Imagens obtidas na internet de ROVs. (a) ROV em operaçao offshore, robô usado para identificar o vazamento de óleo no Campo de Marlim, na Bacia de Campos - Brasil, no ano 2012. (b) Manipuladores de um ROV, observe-se que o braço esquerdo é utilizado para melhorar o posicionamento do ROV na estrutura.

Em virtude dos aspectos mencionados, planejar e seguir uma rota definida para o efetuador final, especificando orientação e posição no espaço de trabalho é um desafio no campo do controle e da robótica. Isso explica porque ainda não temos um AUV com a capacidade de movimentar braços robóticos de maneira totalmente autônoma. A execução de tão variadas tarefas submarinas precisam das habilidades de planejamento de rotas, destrezas mecânicas e a experiencia dos humanos.

A longo prazo, espera-se que os atuais sistemas teleoperados ROV sejam mais inteligentes. O trabalho apresentado neste texto, vai nessa direção, propondo estrategias de ajuda nas manobras de um manipulador submarino para mover seu

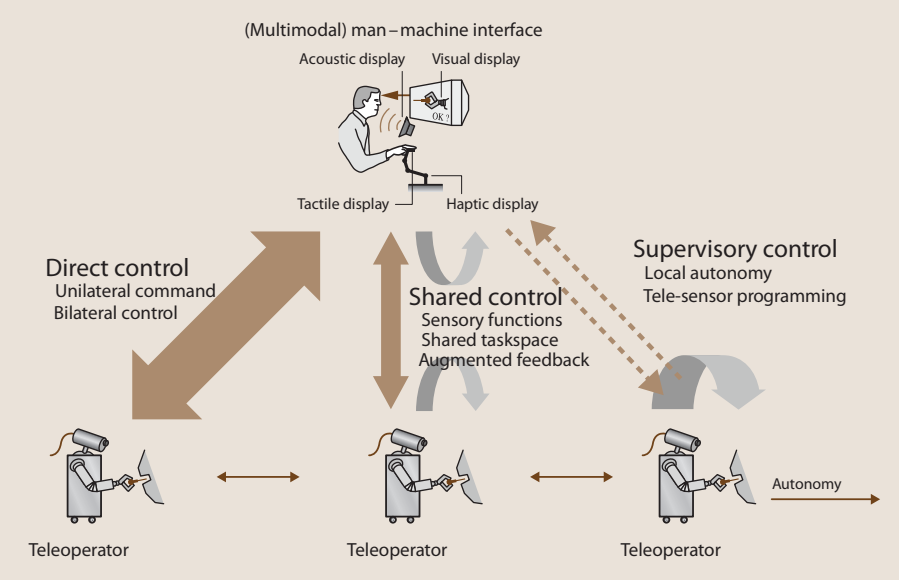


Figura 1.3: Diferentes conceitos de arquiteturas telerobóticas de controle. Figura extraída de [2].

efetuador final de uma posição atual até uma posição desejada de agarramento; diminuindo os tempos de operação, os níveis de estresse dos operadores e consequentemente pode-se lograr minimizar riscos de acidente e redução dos custos de operação para as empresas.

Para fornecer inteligencia aos sistemas robóticos ROV é preciso estabelecer uma arquitetura telerobótica de controle compartilhada (descrita como *Shared Control* na Fig. 1.3). Esta arquitetura permite ao robô realizar tarefas semi-autônomas de acordo com algumas decisões simples do operador, também precisa-se de um sistema de *feedback* sensorial aumentado, como realidade virtual ou outro sistema automático de ajuda [2].

O controle dos movimentos dos robôs seriais é um tópico recorrente na literatura robótica, mas os estudos desenvolvidos são para manipuladores terrestres de base fixa de dinâmica conhecida, em entornos estruturados e instrumentação validada na industria por décadas [8–10]. Com o fim de lidar com ambientes dinâmicos, onde é requerido obter uma rota livre de colisões com o ambiente (deslocamento de robôs moveis ou serie de movimentos dos elos do braço robótico) diversos algoritmos foram propostos na última década [11, 12]. Não obstante que estes algoritmos estão validados de maneira teórica e com alguns testes no laboratório, estas estratégias não tem sido acolhidas pelo setor produtivo. Uma causa provável é a programação off-line, pois os robôs industriais trabalham de maneira repetitiva em uma só rota e em ambientes isolados. Alguns estudos sobre manipuladores em ambientes dinâmicos são feitos em relação com a interação homem-máquina.

Os algoritmos baseados em amostragem para evitar colisões de maneira online é

um tópico inovador para robôs em ambientes não-estruturados. Na última década estes algoritmos foram estudados e apresentados para futuras aplicações de carros controlados autonomamente e neste trabalho são estudados para o caso de manipuladores seriais. Nos próximos capítulos se pode observar as vantagens dos métodos escolhidos para geração de rotas baseados na teoria mais recente, assim também de um sistema de sensorial aumentado que pode ser usado além no caso de manipulação em ROVs.

1.1 Propósito deste Trabalho

O objetivo deste trabalho é indicar como as atuais tecnologias em realidade aumentada junto com os recentes algoritmos de geração de rotas livres de colisão, baseados em amostragem, podem ser usados nos robôs manipuladores seriais submarinos para facilitar as tarefas dos operadores humanos. Em especial, tem-se uma abordagem do planejamento do movimento autônomo dos braços robóticos para obter uma posição desejada de agarramento do efetuador final num ambiente não-estruturado.

Vale ressaltar também que há ainda muito trabalho a ser continuado, tendo em vista que uma quantidade significativa de questões precisam ser resolvidas antes de realizar testes em condições reais de operação. Neste trabalho não se tem a intenção de formular teorias avançadas no campo da engenharia de controle, mas permite no futuro a implementação destes tipos de estruturas baseadas nos diversos conceitos apresentados aqui.

1.2 Organização

A apresentação desta tese está organizada nos seguintes capítulos:

Capítulo 1. Apresenta o contexto atual da tese, abordando suas motivações, alguns conceitos gerais, delimitando os objetivos e indicando a organização do texto.

Capítulo 2. Em primeiro lugar exibe uma revisão da literatura disponível referente aos problemas surgidos na manipulação submarina e as soluções particulares sugeridas por pesquisadores. Também é apresentado os principais conceitos e representações formais da teoria robótica, da servovisão e da geração de rotas livres de colisões para o entendimento adequado dos seguintes capítulos.

Capítulo 3. Indica as características do braço robótico *TITAN-IV* da *Schilling Robotics - FMC*, que é o estudo de caso da tese, também faz um análise matemática das cinemáticas e singularidades do robô.

Capítulo 4. Nesta seção se aprecia como realizar implementações da Realidade Aumentada na engenharia e especificamente para o estudo de caso, tornando-se um sistema sensorial aumentado para auxilio do pilotos de ROVs.

Capítulo 5. Descreve como solucionar com os algoritmos de geração de rotas livres de colisões as dificuldades típicas para conseguir movimentos autônomos dos manipuladores: a cinemática inversa estendida, os ambientes dinâmicos e obtenção de rotas curtas.

Capítulo 6. Mostra como produzir um software para auxiliar a teleoperação de braços robóticos usando a Realidade Aumentada e o planejamento de rotas automáticas. O capítulo está dividido em três partes: a primeira indica como realizar a interação entre homem e máquina. A segunda descreve o funcionamento e resultados obtidos com um software desenvolvido. A terceira parte estabelece como os pilotos humanos conseguem acompanhar as rotas obtidas automaticamente e introduz uma nova estrategia que gera rotas de fácil seguimento, seus resultados são validados com simulações e ilustra-se as possibilidades futuras desta estrategia.

Capítulo 7. Apresenta as contribuições e principais conclusões dos temas abordados nesta tese assim como oferecer as indicações para continuar em pesquisas futuras.

Capítulo 2

Contexto Bibliográfico

Este capítulo expõe em primeiro lugar as diversas facetas técnicas que influenciam na operação dos ROVs e as soluções propostas por diferentes pesquisadores sobre diversos pontos de vista. Estas técnicas são apresentadas de maneira rápida, já que existem muitas estratégias propostas tanto quanto problemas a serem resolvidos. Posteriormente, apresentam-se os termos específicos e definições teóricas requeridos para a compreensão dos capítulos seguintes. Finalmente o capítulo introduz o problema geral da geração de rotas e destaca a importância de seu estudo.

Um objetivo deste capítulo é citar as teorias atuais que visam tornar os ROVs sistemas autônomos. Entretanto, todas essas teorias não são aplicadas no desenvolvimento da tese, estas são apresentadas para visualizar a complexidade de manobrar o conjunto base-manipulador. Até este momento evidências de solução geral para o caso da manipulação autônoma, ou uma proposta formal onde sejam reunidas as áreas de realidade aumentada e geração rotas dinâmicas não foram encontradas na literatura.

2.1 Sistema Base-Manipulador

Os veículos operados remotamente ROVs podem ser divididos em dois grupos: ROV de observação e ROV de trabalho [13]. Os primeiros são de dimensões menores comparados com o segundo grupo cujas dimensões são em torno de dois metros de altura por três de comprimento; em geral os ROVs de trabalho possuem dos braços robóticos chamados de manipuladores. De agora em diante são mencionados os ROVs de trabalho por simplicidade como ROVs.

A base do ROV é uma estrutura normalmente metálica tipicamente de forma prismática na qual estão montados os sistemas de flutuação, propulsão (*thrusters*), iluminação, comunicação e sensoriamento (câmeras, sonar, etc.). Cada manipulador é considerado como um robô de cadeia cinemática serial com seu próprio sistema atuador.

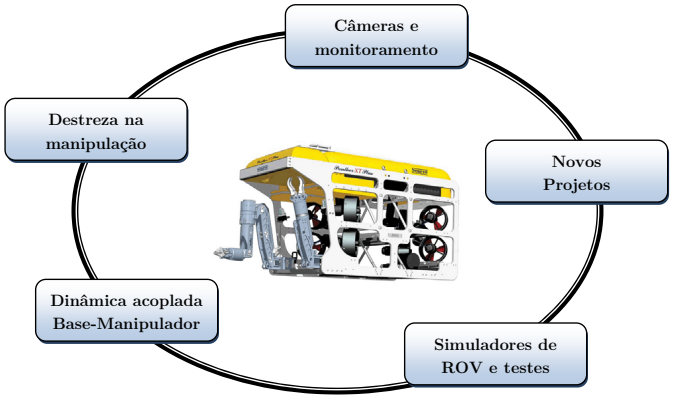


Figura 2.1: Tópicos gerais para a compreensão da interação base-manipulador em ROVs.

A Fig. 2.1 indica cinco temas ao redor do sistema base-manipulador, estes temas foram definidos como resultado do agrupamento de textos especializados que são referenciados nas próximas secções e que ajudam na compreensão da complexidade da teleoperação.

2.1.1 Câmeras e Monitoramento

Um dos principais sistemas para a operação remota de um ROV são as câmeras, as imagens transmitidas por elas fornecem muita informação aos operadores do ambiente onde o robô atua. Em diversos campos da ciências, as atuais câmeras digitais pode ser usadas para melhorar as atividades feitas por máquinas, no entanto nas condições submarinas as câmeras apresentam alguns desafios para ser usadas como pontos de melhora, além de precisar de ambientes livres de sujeira que atrapalham a visão. Por exemplo, algumas vezes usar a função zoom é essencial nas operações do efetuador final, mas os pilotos devem usar as imagens de outras câmeras para conferir a configuração do manipulador e posição da base.

Na década de 1990 com a popularização dos ROVs começaram várias pesquisas para usar as câmeras como parte do controle destes veículos. Em [14] foi proposta a análise das imagens para inferir a atitude do ROV, esta estratégia usa informação bidimensional dos contornos retos dos objetos captados pela câmera para determinar um estado 3D do observador. Os autores mostram que esta técnica funciona bem para ambientes simples e com geometrias adequadas.

No ano 2001 apareceram os primeiros estudos de medições submarinas baseadas em imagens estereoscópicas [15], no mesmo ano foram descritas técnicas de servo-visão (técnicas que permitem posicionar automaticamente um robô com relação ao seu ambiente, utilizando dados visuais), baseadas na aritmética das matrizes de transformação homogênea [16], correspondentes aos movimentos da câmera pan/tilt

do ROV [17].

Uma tendência nos últimos anos é a visão estereoscópica, uma descrição do possível uso de duas câmeras sincronizadas para o seguimento de objetos no ambiente dos ROVs é descrito em [18]. Outro estudo que descreve a complexidade do uso da visão estereoscópica no âmbito da inspeção visual, navegação e mapeamento submarino encontra-se em [19]. A utilização destas técnicas para calcular a posição do manipulador pelas câmeras é descrito em [20].

Uma preocupação constante no uso real de qualquer técnica que envolva câmeras são as condições submarinas. Neste caso a instrumentação é um fator predominante, pois tem-se dificuldade de posicionamento preciso dos componentes mecânicos, são requeridos sistemas especiais de sensoriamento para controle das juntas, por exemplo os manipuladores são propensos a vibrações do oceano que afetam as medições [21].

2.1.2 Destreza na Manipulação

A manipulação de objetos usando ROV pode ser entendida pelo número dos graus de liberdade (GDL) extra que possuem o conjunto base-manipulador. Em geral o sistema completo tem mais de 10-GDL, nestas condições o sistema é redundante propriamente dito. Mas o problema da manipulação é como os GDL são utilizados. Por exemplo, tem-se o caso particular de usar a base para movimentar objetos que estão sujeitos ao efetuador final (EF) e às vezes os operadores preferem usar a base para pequenas operações que poderiam ser feitas com o manipulador. Isto pode ser explicado felo fato que os GDL nos eixos x, y, z são controláveis diretamente com os propulsores, para caso do EF precisa-se das destrezas do operador para inferir a cinemática inversa e movimentar um objeto agarrado ao EF da maneira desejada. A Fig. 2.2 apresenta em resumo uma operação de manipulação onde é requeridos os GDL combinados do sistema base-manipulador. Esta figura mostra três fases, aproximação, operação de agarramento e remoção do alvo. A maneira de realizar a teleoperação depende de conhecer a posição do alvo em relação com o robô, da viabilidade do agarramento e a detecção de colisões com o ambiente [22].

A flutuação e análise cinemática do sistema completo base-manipulador não é escopo deste trabalho, entretanto é mencionado nesta seção para futuros avanços e pesquisas, tais como as feitas nos últimos anos em [23–25]. Na maioria dos casos os pilotos tentam eliminar qualquer movimento da base do ROV, enquanto isso, o operador do manipulador faz as manobras. Quando o piloto da base ajuda nas manobras do manipulador usando os propulsores as vezes simplifica a tarefa do operador, mas elimina a redundância inerente ao sistema ROV. Um estudo para lidar com a falta de redundância usando o Método de Projeção do Gradiente, GPM (*Gradient Projection Method*) é apresentado em [26]

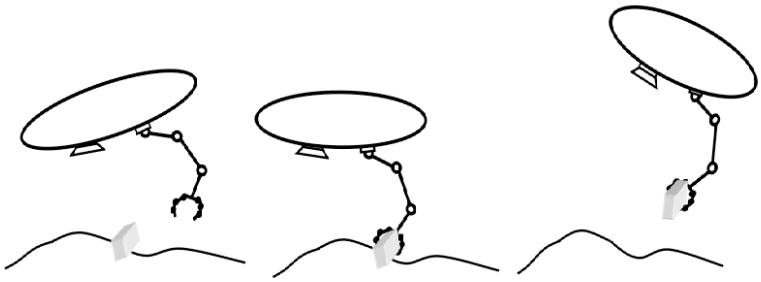


Figura 2.2: Intervenção submarina, representado por um sistema flutuante de manipulador único, o qual tem que se aproximar, logo agarrar e transportar um objeto à um local diferente. Figura obtida de [22].

Mesmo que o manipulador possua 6-GDL ou menos, o manipulador pode ter redundância de acordo com o tipo ou a prioridade da tarefa a ser realizada. Se uma tarefa requer uma posição precisa, mas não a orientação da ponta do EF, o manipulador com 6-GDL tem redundância de 3-GDL. O problema utilizando estes tipos de redundâncias é chamado de *resolução de redundância por prioridade-tarefa* [27]. Um estudo que abriga análise destas redundâncias dando solução com uma função escalar para Programação Quadrática Sequencial (SQP) offline é apresentado em [28].

Em robótica a capacidade de um braço robótico para atingir uma posição em seu espaço de trabalho e de ser capaz de orientar a ferramenta é chamada *destreza* (*dexterity* na literatura inglesa). O lugar do espaço de trabalho que permite a circulação em todos os graus de liberdade euclidianos é chamado de *espaço de trabalho completo-destro* (*full-dexterous workspace*). Em [29] apresenta-se uma representação visual destes espaços para ajudar aos operadores a reconhecer as restrições dos manipuladores usados em ROVs.

2.1.3 Dinâmica Acoplada

A manipulação é fortemente influenciada pela dinâmica acoplada. Hildebrandt em [30] estabeleceu que o problema pode ser abordado de maneiras diferentes: mantendo a posição do ROV tão estável quanto possível, utilizando algoritmos de estabilização, ou reconhecer e compensar o movimento ROV pela correção da posição do efetuador do manipulador. O primeiro tem sido tema de intensa pesquisa e existe um bom número de métodos com desempenho aceitável, por exemplo em [7, 31, 32]. A compensação de movimento ROV usando o manipulador é mal vista em sistemas submarinos, pois os sistemas manipuladores são geralmente diretamente teleoperados com pouco ou nenhum controle do computador, fazendo com que os algoritmos de compensação automática sejam de difícil adoção.

Uma das primeiras descrições da dinâmica base-manipulador foi realizada por McMillan e outros [33] no ano 1995, os autores usaram as equações recursivas Newton-Euler além incorporarem forças de flutuação, aceleração do fluido e viscosidade. Devido à complexidade do problema dinâmico, em 1998 e 2001 foram apresentados outros conceitos de controle baseados no sensoriamento na junta base-manipulador [34, 35].

A segunda abordagem é o estudo do controle do manipulador usando estratégias de apoio. Por exemplo, em [36] Hildebrandt, Albiez e Kirchner propõem perfis rampa de posição das juntas para eliminar a necessidade de controle puramente manual do braço escravo. Os autores mostram resultados experimentais e apresentam as dificuldades na execução, em parte por confiabilidade do funcionamento dos sensores e nos rápidos movimentos do manipulador no ambiente real. Em [37] Soylu e outros, apresentam o controle PID aplicado no manipulador 4-GDL, no entanto, a dinâmica dos propulsores foi negligenciada e assumiu-se que não há corrente oceânica.

Por outro lado [38] propôs o controle linear do EF do manipulador usando três etapas: primeiro estima-se a posição do ROV usando o Filtro de Kalman Estendido. Segundo usa-se um método de compensação da velocidade baseado na cinemática inversa diferencial. E terceiro, o que os autores chamam de Controle Preciso do Espaço de Trabalho, este utiliza diferenciais de terceira ordem da cinemática inversa usando a expansão Taylor. Também se apresenta nos últimos anos estudos que contemplam o cabo umbilical usando modelos SISO de controle por modo deslizante [39]. Finalmente, embora muitos estudos estão sendo conduzidos, nenhum é conclusivo e a maioria ficam no campo da simulação.

2.1.4 Simuladores e Testes

Em 1994 foram apresentados os primeiros conceitos para a simulação de ROV [40], mas, com o aumento da capacidade do processamento dos computadores, as simulações se tornaram em uma importante ferramenta para a industria e pesquisadores no mundo inteiro.

Na última década todas as grandes empresas de ROV, incluindo Oceaneering, Subsea 7, Fugro, Technip, Acergy, e Schilling Robotics, tem investido fortemente, tanto internamente como externamente empregando empresas de software (como GRI ou GRL), para criar simuladores que estejam mais perto da realidade. Similar à indústria da aviação, muitos dos novos pilotos de ROV usam programas de pilotagem em treinamento intensivo, para melhorar suas habilidades na operação real. Além disso, aulas de pilotagem avançadas foram desenvolvidas para ajudar a acrescentar as competências de pilotagem dos atuais operadores de ROV.

Os novos recursos visuais das placas de aceleração gráfica sugerem que no futuro

próximo, a fusão dos campos da realidade e da simulação permita aumentar as capacidades operativas dos ROVs [41]. Outra tecnologia que pode influenciar o futuro das teleoperações são os monitores ou telas sensíveis ao toque. Um dos estudos neste campo para o controle de um robô móvel é apresentado em [42].

Outro ponto de partida para a melhoria e autonomia dos ROV são os testes reais. Em [43] apresenta-se a interação entre simulação e testes reais para avaliação de novas técnicas de manipulação. Os testes foram realizados com um manipulador real montado sobre um robô cartesiano, tipo Gantry, este último é utilizado para simular os movimentos da base do ROV. Todo o conjunto está submergido em um reservatório com água que possui duas janelas. Com o reservatório é possível controlar as condições de iluminação e de clareza da água.

Recentemente [44] mostra o estudo conduzido para testes hidrodinâmicos dos ROV. Neste caso foi utilizado um modelo em escala de um novo ROV projetado para operar em profundidades de 4500 metros. Com testes como estes, os autores pretendem decifrar os coeficientes hidrodinâmicos próprios do projeto e validar os modelos matemáticos. Outros estudos para a determinação dos coeficientes hidrodinâmicos podem ser encontrados em [45, 46].

2.1.5 Novos Projetos

É inegável que as empresas fornecedoras de ROV liderem os desenvolvimentos destes sistemas robóticos. No entanto, outras maneiras para aumentar as capacidades dos ROVs podem vir dos novos projetos. Ainda que estes projetos não sejam comerciais, eles estabelecem abordagens originais ao problema de manipulação. Por exemplo, em [47, 48] é mostrada a concepção de um ROV com seus manipuladores com morfologia semelhantes aos braços humanos, de maneira que seja mais natural a tarefa de manipulação; além disto, propõem um novo sistema de flutuação e controle de atitude do ROV.

Em [49] é apresento um conceito radical, um sistema híbrido entre a estrutura típica da navegação autônoma do AUV e a operabilidade dos manipuladores do ROV. O projeto está definido como um sistema mecânico onde a base pode mudar sua configuração desde a forma quadrada do ROV para uma forma hidrodinâmica do AUV, similar à carcaça da tartaruga, na qual seu manipulador podem ser coletado em seu interior.

Para desenvolver novos ROV precisam-se além de vontade investigativa os recursos econômicos. Em geral são requeridos conhecimentos interdisciplinares e experiência no tema. Por conseqüência, estas iniciativas estão enfocadas em melhorar as capacidades dos atuais veículos submarinos, por exemplo, em [50, 51] propuseram, independentemente, um ROV híbrido que atinge 11.000 metros de profundidade para

as operações científicas. Por outro lado, há poucos estudos voltados a melhorar os manipuladores submarinos existentes, estes ainda são de quatro ou seis graus de liberdade, de similar configuração, com atuadores elétricos e hidráulicos; assim como é descrito nos capítulos subseqüentes deste texto.

2.2 Servovisão

A servovisão refere-se à utilização de dados de visão por computador para controlar o movimento de um robô. Os dados de visão podem ser adquiridos a partir de uma câmara que está montada sobre o robô manipulador ou robô móvel; em caso que o movimento do robô induza movimento da câmara, esta pode ser fixada na área de trabalho de modo que possa observar o movimento do robô a partir de uma configuração estacionária. Um dos primeiros trabalhos descrevendo os sistemas servovisão foi feito em 1996 por Hutchinson em [52]. O desenvolvimento matemático de todos esses casos é semelhante e apresentado recentemente em [2, 53, 54], estes trabalhos são de referencia mundial e mostram uma descrição geral da servovisão se concentrando no caso chamado de configuração olho-em-mão.

Uma característica fundamental da servovisão, em comparação com o controle de movimento e força, é o fato de que as variáveis controladas não estão diretamente medidas por sensores, mas são obtidas a partir das quantidades medidas através de complexos cálculos baseados em algoritmos de processamento de imagem e visão computacional [8].

2.2.1 Processamento da Imagem

A informação visual, ao contrário da informação fornecida por outros tipos de sensores, é variada e muitas vezes com perturbações, portanto, requerem-se diversas e complexas transformações computacionais antes que possa ser utilizada para controlar um sistema robótico. O objetivo dessas transformações é a extração de informações numérica a partir da imagem, que fornece uma descrição sintética e robusta dos objetos de interesse na cena, através dos chamados parâmetros da imagem. Para este fim, duas operações são necessárias. A primeira é chamada de *segmentação*, seu alvo é obter uma representação adequada para identificar as características mensuráveis da imagem. A operação subseqüente é denominada *interpretação* e refere-se à medição dos parâmetros das características da imagem.

Segmentação da Imagem. Consiste em um processo de agrupamento, através da qual a imagem é dividida num certo número de grupos, de modo que os componentes de cada grupo tem características semelhantes. Normalmente, os grupos distintos da imagem correspondem a objetos do ambiente, ou partes de objetos ho-

mogêneos. Tem-se duas abordagens complementares ao problema da segmentação de imagens: *Segmentação baseada em regiões*, nesta as regiões são obtidas pelos os pixels similares adjacentes. Em geral são usadas imagens binárias, pois na prática no histograma em escala de cinza os grupos são ruidosos e difíceis de identificar. A segunda abordagem é a *segmentação baseada em limites*, esta técnica usa a escala de cinza para agrupar descontinuidades devidas à mudança abrupta da intensidade de luz, logo são construídos segmentos curtos de curvas, e, finalmente, são obtidos os limites juntando estes segmentos de curva através de primitivas geométricas, muitas vezes conhecidas antecipadamente.

Interpretação da Imagem. É o processo de cálculo dos parâmetros característicos da imagem. Estes parâmetros individualizam matematicamente uma região e podem entregar as características de posição, orientação e forma de um objeto bidimensional extraído da imagem. Existem várias técnicas e muitos artigos que apresentam algoritmos para *Segmentação da Imagem* e seus resultados dependem das aplicações específicas para as quais foram concebidos.

2.2.2 Componentes Básicos

O objetivo de todos os esquemas de controle baseado na visão é minimizar um erro $e(t)$ que geralmente é definido como:

$$e(t) = s[m(t), a] - s^* \tag{2.1}$$

Esta é uma formulação geral que é acompanhada por uma variedade de abordagens [53]. O vetor $m(t)$ é um conjunto de medições tomadas da imagem, por exemplo coordenadas de imagem dos pontos de interesse ou centróide de um objeto. Estas medições de imagem são utilizadas para calcular um vetor de características, $s[m(t), a]$, no qual a é um conjunto de parâmetros que representam conhecimentos adicionais do sistema, por exemplo, parâmetros da câmera ou modelos tridimensionais dos objetos da cena. O vetor s^* contém os valores desejados.

Depois da seleção de s, o projeto de controle pode ser simples, porém uma escolha logica é um controle de velocidade. Para fazer isso é necessário a relação entre a variação no tempo de s e a velocidade da câmera. Em geral a velocidade da câmera pode ser denotada como $\mathbf{v}_c = (v_c, w_c)$, sendo v_c a velocidade linear instantânea da origem do sistema de coordenadas da câmera e w_c é sua velocidade angular instantânea. A relação entre $\dot{s}$ é $\mathbf{v}_s$ dada por:

$$\dot{s} = \mathbf{L}_s \mathbf{v}_c \tag{2.2}$$

Na qual, $\mathbf{L}_s \in \mathbb{R}^{k \times 6}$ é chamado de *matriz de interação* relacionada a s [55].

Usando a Eq. 2.1 e 2.2 obtemos a relação entre velocidade da câmera e a variação no tempo do erro:

$$\dot{e} = \mathbf{L}_e \mathbf{v}_c \tag{2.3}$$

Sendo $\mathbf{L}_e = \mathbf{L}_s$. Para projetar um controlador do robô usando servovisão, pode-se utilizar a $\mathbf{v}_c$ como o sinal de entrada, agora para tentar garantir um decremento em desacoplo do erro (por exemplo, $\dot{e} = -\lambda e$), a Eq. 2.3 pode ser escrita como:

$$\mathbf{v}_c = -\lambda \mathbf{L}_e^+ e \tag{2.4}$$

A expressão $\mathbf{L}_e^+ \in \mathbb{R}^{k \times 6}$ é a pseudo-inversa Moore-Penrose de $\mathbf{L}_e$. Isto é $\mathbf{L}_e^+ = (\mathbf{L}_e^T \mathbf{L}_e)^{-1} \mathbf{L}_e^T$, quando $\mathbf{L}_e$ tem posto completo 6. Esta escolha permite que $\|\dot{e} - \lambda \mathbf{L}_e \mathbf{L}_e^+ e\|$ e $\|\mathbf{v_c}\|$ sejam mínimas[1]. Em sistemas reais de servovisão é impossível saber perfeitamente os valores de $\mathbf{L_e}$ ou $\mathbf{L_e^+}$, por isso, é necessário aproximar ou fazer uma estimativa destas duas matrizes. Portanto, a aproximação da pseudo-inversa da matriz de interação pode ser representada por $\widehat{\mathbf{L_e^+}}$. Usando esta notação, a equação que caracteriza o sistema é da seguinte forma:

$$\mathbf{v_c} = -\lambda \widehat{\mathbf{L_e^+}} \mathbf{e} \tag{2.5}$$

Fechando o loop, e assumindo que o controlador do robô é realizável, a lei de controle fica como:

$$\dot{e} = -\lambda \mathbf{L_e} \widehat{\mathbf{L_e^+}} \mathbf{e} \tag{2.6}$$

Este é o projeto básico aplicado ao sistemas de controle de servovisão de maneira geral, mas deve ser escolhidos apropriadamente s, $\mathbf{L_s}$ ou as matrizes de interação.

2.2.3 Tipos de Servovisão

Os sistemas de controle baseados em visão podem ser divididos em duas categorias, aqueles que fazem a servovisão no espaço operacional, também denominados *servovisão baseada na posição* (PBVS), e aqueles que fazem a servovisão no espaço de imagem, também conhecidos como *servovisão baseada na imagem* (IBVS). A diferença principal reside no fato de que os sistemas da primeira categoria utilizam medições visuais para reconstruir a posição (ou configuração) relativa de um objeto com respeito ao robô, ou vice-versa, enquanto que os esquemas da segunda categoria são baseados na comparação de parâmetros característicos, da imagem de um objeto, entre a posição atual e posição desejada [8]. Há também sistemas que combi-

[1] Mais informação desta generalização matemática pode ser encontrada em [53, 54].

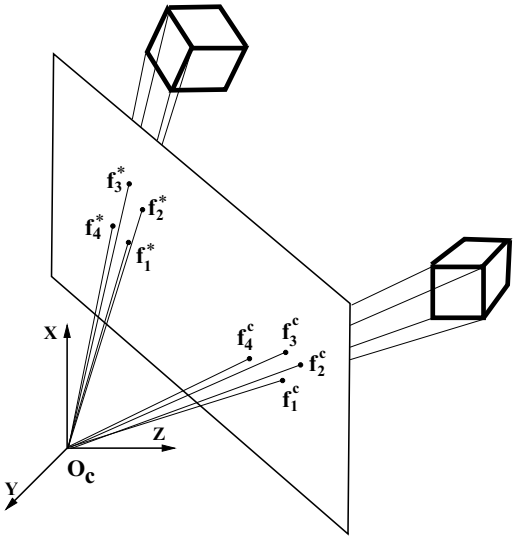


Figura 2.3: Un exemplo de Servovisão Baseada na Imagem (IBVS). Imagem obtida de [56].

nam características comuns a ambas categorias, estes podem ser classificados como *sistemas híbridos de servovisão.*

Servovisão Baseada na Imagem (IBVS). Para este tipo de servovisão as medições da imagem 2D são utilizadas diretamente para estimar o movimento desejado do robô. Por isso, algumas vezes é chamado de servovisão 2D. Em geral o controle é feito através da redução do erro de distância da imagem entre um conjunto de características da imagem no plano da mesma.

A Fig. 2.3 apresenta um exemplo de IBVS no caso de uma câmera estática e um robô segurando um objeto. Um apropriado número de pontos característicos do objeto são rastreados e usados para gerar um vetor de medições atuais, $\mathbf{f^c}$. O vector de medições de referência é denotado como $\mathbf{f}^*$. A função de erro é definida como uma função da distância entre as medições, $\mathbf{e} = \mathbf{f^c} - \mathbf{f}^*$. Esta função de erro é então atualizada em cada quadro e usada junto com a matriz de interação para estimar o sinal de entrada de controle do robô [56].

A vantagem desta solução conceptual é fato de que não é necessária uma estimativa em tempo real da postura do objeto em relação à câmara. Além disso, uma vez que o controle atua diretamente nos parâmetros característicos da imagem, é possível manter o objeto no interior do campo de visão da câmara durante o movimento. No entanto, devido à não-linearidade do mapeamento entre os parâmetros da imagem e as variáveis operacionais espaciais, as configurações singulares podem ocorrer, o que causa a instabilidade ou a saturação da ação de controle. Em relação às trajetórias do EF, estas não podem ser facilmente previstas com antecedência e pode produzir colisões com obstáculos ou violação dos limites das juntas [8, Cap.10].

O controle baseado na imagem é menos sensível a erros de calibração com respeito a PBVS, no entanto, é necessário o cálculo da matriz de interação da imagem, e seu valor depende da distância entre o alvo e a câmara, o que é difícil de avaliar [57]. Em geral o IBVS é conhecido por ser mais robusto com respeito à câmara e aos erros de calibração do robô, mas, a convergência é garantida apenas em uma região em torno da posição desejada, a qual é difícil determinar analiticamente. Exceto em casos muito simples, a análise de estabilidade dos erros de calibração parece impossível de determinar já que o sistema é acoplado e não-linear.

Usando a notação matemática da Eq. 2.1, os sistemas tradicionais de controle baseados em imagem usam as coordenadas no plano da imagem de um conjunto de pontos para definir o conjunto s. As medições da imagem m são geralmente as coordenadas de pixel do conjunto de pontos de imagem (embora esta não é a única escolha possível), e os parâmetros a na definição de $s = s[m, a]$ são os parâmetros intrínsecos da câmara obtidos de medições de imagem expressa em pixeis [2, Cap.24].

Servovisão Baseada na Posição (PBVS). Estes sistemas recuperam a informação tridimensional da cena sendo conhecido o modelo da câmera, usualmente com o modelo geométrico do alvo, para estimar a posição e a orientação do alvo em relação à câmera no sistema coordenado. A tarefa de localização e seguimento é definida no espaço 3D. Por isso, algumas vezes é chamado de servovisão 3D.

A desvantagem desta abordagem é que, devido à ausência de um controle direto das características da imagem, o objeto pode sair do campo de visão da câmara durante o transiente e como uma consequência têm-se erros de planejamento de movimentos, logo, o loop de feedback fica aberto devido a falta de medidas visuais e pode acontecer instabilidades [8, Cap.10].

A Fig. 2.4 apresenta um esquema clássico de PBVS para uma câmera montada no EF do manipulador. P_d e φ_d representam a posição e orientação desejadas. A com-

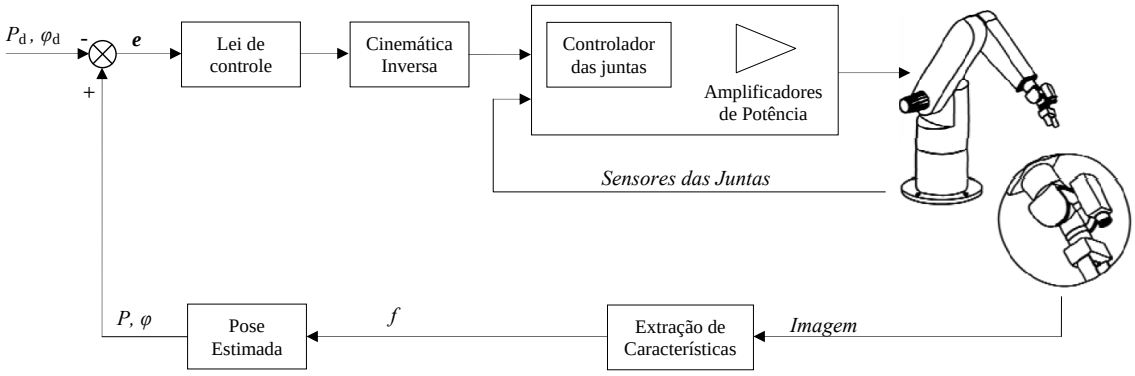


Figura 2.4: Esquema de servovisão baseada na posição no caso da câmera montada no robô. Imagem modificada de [58].

paração entre os valores estimados de posição P e orientação φ devidos a aquisição da imagem, extração das características (usando os parâmetros da câmera e modelo dos objetos da cena) e posterior conversão com una função f, são comparados com os valores desejados para obter o erro de controle e.

É típico definir s em termos da parametrização usada para representar a pose da câmera. Percebe-se que os parâmetros a, na formulação Eq. 2.1, são os parâmetros intrínsecos da câmera e do modelo 3D do objeto. O fato de calcular a pose de um objeto usando um conjunto de medições é um problema clássico da visão por computador que é chamado *problema de localização 3D* e muitas soluções têm sido apresentadas na literatura, por exemplo em [59, 60].

Outra tecnologia que lida com o problema da localização 3D é a Realidade Aumentada ou AR, por suas siglas *Augmented Reality*, foi concebida adequadamente há uma década no campo da visão computacional e recentemente permitiu o desenvolvimento de aplicativos para celulares e outros dispositivos móveis, a AR também tem diversos usos no campo do entretenimento e na pesquisa médica. O foco da presente tese é a utilização desta tecnologia, por esta razão a AR é explicada com melhor detalhe nos seguintes capítulos.

2.3 Teoria Robótica

A teoria robótica atual é extensa, ainda que seu desenvolvimento começou em meados dos anos 1950. O objetivo desta seção é apresentar as definições básicas para a compreensão da tese, fazendo referência aos conceitos teóricos encontrados na literatura clássica, como em [8, 10, 16, 61, 62]. É enfatizada a descrição dos termos matemáticos dos robôs seriais antropomórficos de cadeia aberta, chamados aqui simplesmente como manipuladores.

2.3.1 Nomenclatura Básica

Em robótica tem-se interesse pela interação entre um ou mais corpos rígidos, por exemplo, os elos de um manipulador. Um corpo sólido a pode ser representado por um sistema de coordenadas $\{a\}$, da mesma maneira $\{b\}$ representa um corpo b. Um sistema $\{b\}$ é descrito em relação a outro sistema $\{a\}$ por uma translação P_{ab} e uma rotação R_{ab}. $P_{ab} \in \mathbb{R}^{3\times1}$ é um vetor composto pelas coordenadas do origem de $\{b\}$ visto desde $\{a\}$.

Uma interpretação da matriz de rotação $R_{ab} \in \mathbb{R}^{3\times3}$ é uma transformação que usa giros seqüenciais para orientar o sistema $\{b\}$ até $\{a\}$. Tradicionalmente é adotada uma notação compacta de três vetores unitários chamados de *cossenos diretores*. Uma descrição mais completa é encontrada em [8, 16]. Outra interpretação da

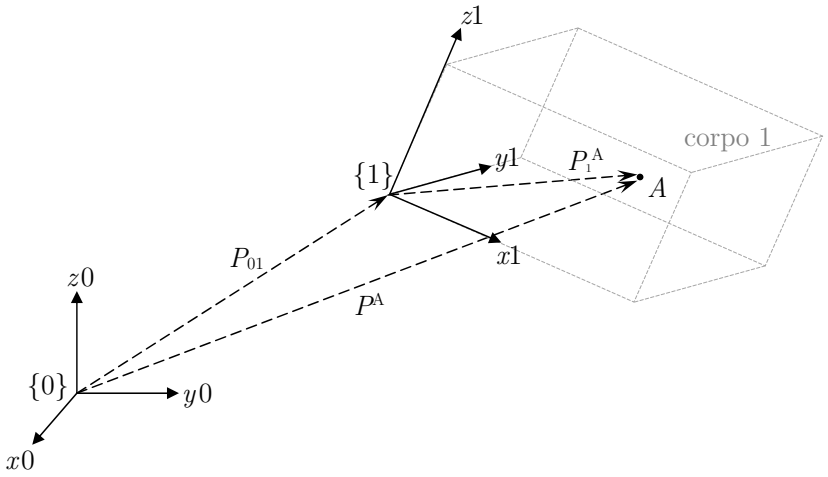


Figura 2.5: Notação básica de sistemas de coordenadas e pontos.

matemática das rotações R é feita usando uma representação exponencial, um dos livros que utiliza essa abordagem é [63].

Convém saber que R é uma matriz ortonormal, o que significa que: $R^T R = I_3$, sendo I_3 a matriz identidade 3×3. Pode-se verificar que $R^T = R^{-1}$. Estas características fazem que a matriz de rotação seja incluída no chamado *grupo especial ortonormal* $SO(3)$, onde $\det(R) = 1$ (Mais informação em [64]).

Por outro lado, tem-se uma representação reduzida para os pontos que são representados em um só sistema de coordenadas, por exemplo, a Fig. 2.5 apresenta o sistema inercial $\{0\}$ e um sistema $\{1\}$ que representa o corpo prismático 1. O ponto A é representado no sistema $\{1\}$ do corpo por o vetor P_1^A, entretanto, o ponto A é representado no sistema inercial $\{0\}$ por P_0^A o simplesmente em notação reduzida como P^A. Conseqüentemente, a origem de $\{1\}$ visto desde o sistema inercial $\{0\}$ é denotado como P_{01}.

Se temos a descrição de traslação P_{01} e orientação R_{01}, um ponto A descrito no $\{1\}$ pode ser referenciado no sistema inercial $\{0\}$ como:

$$P^A = P_{01} + R_{01} P_1^A \tag{2.7}$$

Uma forma mais compacta de representar a configuração do corpo rígido 1 é mediante a equação:

$$\begin{bmatrix} P^A \\ 1 \end{bmatrix} = \begin{bmatrix} R_{01} & P_{01} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_1^A \\ 1 \end{bmatrix} \tag{2.8}$$

Então, definindo os vetores estendidos

$$\bar{P}^A = \begin{bmatrix} P^A \\ 1 \end{bmatrix}, \quad \bar{P}_1^A = \begin{bmatrix} P_1^A \\ 1 \end{bmatrix}$$

E a matriz de transformação homogênea

$$T_{01} = \begin{bmatrix} R_{01} & P_{01} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.9}$$

Temos que:

$$\bar{P}^A = T_{01}\bar{P}_1^A \tag{2.10}$$

Assumindo que os corpos são designados com letras minúsculas e pontos com letras maiúsculas, para um ponto P, descrito no corpo a, é definido no corpo b de maneira general como:

$$\bar{P}_a = T_{ab}\bar{P}_b \tag{2.11}$$

A diferença entre a notação 2.10 e 2.11, é que $\bar{P}^A$ indica o ponto A em relação ao sistema de coordenas inercial $\{0\}$. E $\bar{P}_a$ indica o ponto P em relação ao sistema $\{a\}$.

2.3.2 Análise Cinemática

A cinemática é uma parte da mecânica que estuda o movimento, restrito a uma descrição geométrica. Isto é: posições, orientações, velocidades e acelerações. Em robótica, as descrições cinemáticas de manipuladores e suas tarefas atribuídas são utilizadas para configurar as equações fundamentais da dinâmica e controle [62].

Um manipulador pode ser esquematicamente representado desde o ponto de vista mecânico como uma cadeia cinemática aberta de corpos rígidos (elos) ligados por meio de juntas de revolução ou prismáticas. Uma ponta da cadeia é limitada por a base, enquanto que um efetuador final (EF) é montado na outra ponta. O movimento resultante da estrutura é obtido pela composição dos movimentos elementares de cada elo com respeito ao anterior. O manipulador é caracterizado por um *braço* que garante a mobilidade, um *punho* que confere a destreza e o EF que executa a tarefa exigida do robô [8].

Em uma cadeia cinemática aberta, cada junta prismática ou de revolução fornece a estrutura com um único grau de liberdade (GDL). Uma junta prismática cria um movimento de translação relativo entre dois elos, enquanto que uma junta de revolução cria um movimento de rotação relativo entre dois elos. Juntas de revolução são geralmente preferidas, tendo em vista seu tamanho compacto e confiabilidade. Por outro lado, em uma cadeia cinemática fechada, o número de GDL é menor do que o número de juntas, tendo em conta as limitações impostas pela malha.

O estudo da cinemática pode ser abordado a partir de dois pontos de vista: a

análise cinemática e a *síntese cinemática*, embora que os dois enfoques são interligados. A *análise cinemática* é dedicada a estudar os efeitos derivados dos movimentos relativos entre vários elos de um manipulador. Existem dois tipos de análise cinemáticos, a *cinemática direta* e *cinemática inversa*. Por outro lado, a *síntese cinemática* é o processo inverso, neste caso, o designer tem o desafio de inventar um novo manipulador ou máquina, que deve possuir certas propriedades cinemáticas desejadas, tais como espaço alcançável e destreza do EF [61].

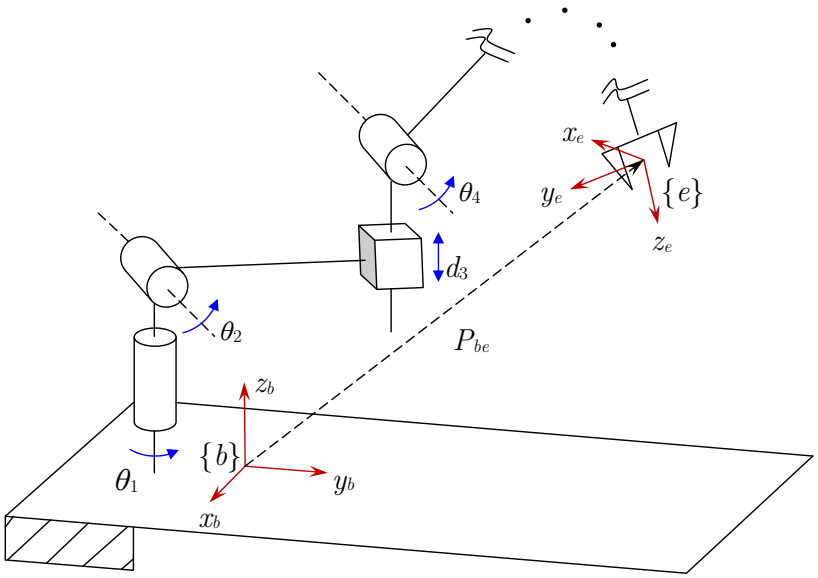


Figura 2.6: Descrição da posição e orientação do sistema de coordenadas do efetuador final $\{e\}$ respeito ao sistema da base $\{b\}$ (adaptado de [8]).

A Fig. 2.6 exibe a representação de um manipulador, na qual, o sistema do EF $\{e\}$ tem como posição relativa ao sistema da base $\{b\}$ o vetor P_{be}. para cada elo i um GDL de uma junta rotacional é denotado por θ_i e o GDL de uma prismática como d_i. De maneira geral uma variável articular, seja rotacional θ_i ou prismática d_i, e designada como q_i, portanto o vetor que representa uma configuração do robô é: $\mathbf{q} = [q_1, q_2, \dots q_n]^T$ sendo n o numero de juntas. Uma maneira de denominar um manipulador é pelos graus de liberdade individuais assim: manipulador de n-GDL.

Cinemática Direta. Determina a posição e orientação do EF, dados os valores das juntas $\mathbf{q}$ de um manipulador n-GDL. A representação do estado do EF é dada por uma matriz de transformação T_{be}, em notação reduzida a cinemática direta está expressada como em [8, Cap.3]:

$$T_e(\mathbf{q}) = \begin{bmatrix} R_e(\mathbf{q}) & P_e(\mathbf{q}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.12}$$

Matematicamente a obtenção da matriz T_{be} (em sua notação reduzida T_e) é dada por a álgebra de matizes de transformação homogênea, usando a pos-multiplicação

das contribuições individuais de posição de um elo respeito ao seu elo imediatamente anterior, assim:

$$T_e = T_{b1}T_{12}\cdots T_{(i-1)(i)}\cdots T_{(n-1)(n)}T_{ne} \tag{2.13}$$

Para aplicações de controle e análise matemática, a cinemática direta pode-se simplificar na seguinte equação:

$$x_e = k(\mathbf{q}) \tag{2.14}$$

Sendo k a função vetorial de cinemática direta. x_e é chamada como pose do EF, composta por representação da posição P_e e da orientação ϕ_e assim:

$$x_e = \begin{bmatrix} P_e \\ \phi_e \end{bmatrix} \tag{2.15}$$

Diversos métodos para obter a cinemática direta foram desenvolvidos, o mais comum na literatura é o uso das matrizes de parâmetros Denavith-Hartenberg, em suas duas notações, standard ou modificada. As duas convenções são explicadas de maneira completa no texto [62]. Outros métodos para obter a cinemática direta são baseados em uma ou outra representação da orientação; entre estes métodos temos: representação em forma exponencial apresentada em [63], usando álgebra de quaternions, ou utilizando o conceito de *screws* apresentado em [61].

Ainda que não haja consenso para adotar um ou outro método, convencionalmente são utilizados por praticidade computacional os parâmetros Denavith-Hartenberg, assim também, são usados no desenvolvimento de esta tese.

Cinemática Inversa. O problema da cinemática inversa consiste na determinação das variáveis das juntas correspondentes a uma determinada posição e orientação do EF (pose do EF). A solução para este problema é de importância fundamental para transformar as especificações de movimento, que são atribuídos no espaço operacional do EF, nos movimentos das juntas que permitem a execução do movimento desejado.

Para o caso de robôs seriais, a cinemática direta é calculada de maneira única, como na Eq. 2.13, uma vez são conhecidos os valores das variáveis articulares q_i. Por outro lado, o problema da cinemática inversa é mais complexo pelas seguintes razões: As equações para resolver são geralmente não-lineares e, por conseguinte, nem sempre é possível encontrar uma solução de forma fechada. Podem existir soluções múltiplas ou soluções infinitas, por exemplo, no caso de um manipulador cinematicamente redundante. As soluções podem ser inaceitáveis, tendo em conta a estrutura cinemática real do manipulador.

Um exemplo é o manipulador de 6-GDL que tem em geral 16 soluções admissíveis,

as quais são eventualmente menores por os limites mecânicos das juntas reais. Na Fig. 2.7 apresenta o caso típico de um robô PUMA com quatro soluções da cinemática inversa para uma desejada posição do EF.

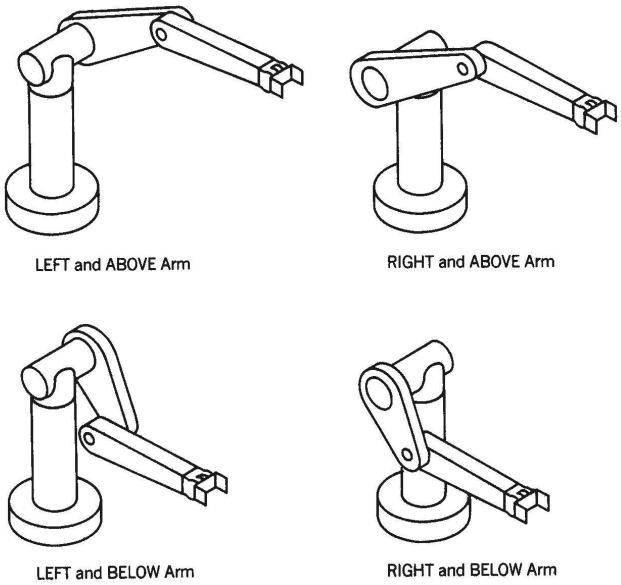


Figura 2.7: Quatro soluções da cinemática inversa de posição para o manipulador PUMA. Imagem obtida de [10].

Para que uma solução da cinemática inversa exista, a posição e orientação desejada do EF devem estar no espaço de trabalho e no espaço de destreza do manipulador. Nos casos em que existem soluções, muitas vezes estas não são alcançadas com metodologias de forma fechada, de modo que são necessários os métodos numéricos. A Fig. 2.8 mostra uma classificação das estratégias desenvolvidas para solucionar o problema da cinemática inversa acordo com [2]. Embora que muitos estudos foram desenvolvidos nos últimos anos, de uma ou outra forma, estas novas estratégias podem ser incorporados na estrutura da Fig. 2.8.

Soluções de forma fechada são desejáveis porque são mais rápidas que as soluções numéricas e facilmente podem identificar todas as soluções possíveis. A desvantagem das soluções de forma fechada é que elas não são gerais, elas dependem da estrutura particular de cada robô. Os métodos mais eficazes para encontrar soluções de forma fechada são técnicas que aproveitam de determinadas características geométricas dos mecanismos específicos do robô. Por exemplo, no caso de manipuladores 6-GDL tradicionais, eles possuem os três últimos GDL em desacoplo e isso permite uma redução do analise [61].

Os *métodos de eliminação simbólica* envolvem manipulações analíticas para eliminar algumas variáveis do sistema de equações não-lineares para obter um conjunto menor de equações. Estas estratégias foram estudadas na década de 1990, como são apresentados em [65–67].

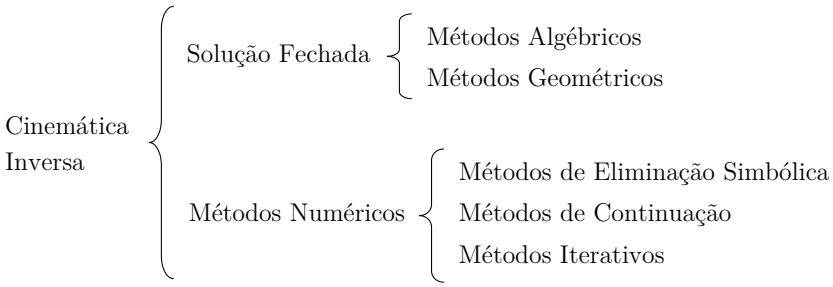


Figura 2.8: Agrupação dos métodos para resolver a cinemática inversa.

Os chamados *métodos de continuação* envolvem um acompanhamento de soluções a partir de outras soluções conhecidas, estas técnicas têm sido aplicadas ao problema da cinemática inversa em [68], e com as propriedades especiais dos sistemas polinomiais, podem ser encontradas todas as soluções possíveis [69].

Um número de diferentes *métodos iterativos* são usados para solucionar o problema da cinemática inversa. A maioria destes métodos converge para uma única solução, baseado em uma configuração inicial, de modo que a qualidade do passo utilizado em cada iteração influencia no tempo e qualidade da solução [2]. Um dos principais enfoques dos métodos iterativos é o uso do inverso do jacobiano ou de seu transposto, uma boa descrição destes métodos é apresentada em [8, Cap.3].

2.4 Planejamento do Movimento

O planejamento é um termo que tem significados diferentes para diversos grupos de pessoas. Na Robótica o planejamento trata da automação de sistemas mecânicos que têm sensoriamento, atuação e capacidades de computação.

O termo *Planejamento do Movimento* (*Motion Planning* em inglês) é referido à capacidade de estabelecer as condições cinemáticas para que um robô realize uma tarefa. O termo *Planejamento de Rotas* (*Path Planning*) às vezes é confundido com *Planejamento de trajetórias* (*Trajectory planning*), em geral este último refere-se na literatura robótica à geração de perfis de posição, velocidade e aceleração das juntas em robôs manipuladores. Enquanto isso, o *Planejamento de Rotas* é uma concepção mais geral fora do campo da robótica; o *Planejamento de Rotas* estabelece as diferentes posições de um "ente" em seu espaço de trabalho em presença de obstáculos conformando um percurso em um lugar geométrico que pode ser multidimensional. No caso da robótica móvel, o *Planejamento de Rotas* estabelece as posições e orientações do robô; para robôs de base fixa são todos os movimentos dos elos para realizar uma tarefa; e para multi-robôs são a soma dos movimentos individuais por robô, em geral são usados a combinação de todos os graus de liberdade controláveis.

Para acentuar a diferença, o *Planejamento de trajetórias* pode ser estudado para dois casos: o primeiro é quando é definido o ponto inicial e final do percurso (movimento ponto-a-ponto). Por exemplo, nos manipuladores temos as operações de *apanhar e colocar* ou PPO (do inglês *Pick-and-Place Operations*), onde o importante é definir a pose inicial e final do EF. O segundo caso é a definição de uma seqüência finita dos pontos que conformam o percurso (movimento multiponto); no caso dos manipuladores têm-se as *operações contínuas* ou CP (do inglês *Continuous-Paths*), um exemplo disto são as tarefas de soldagem e corte onde é requerido definir as posições e velocidades do EF. Geralmente os métodos para geração de perfis das juntas de um manipulador são obtidos usando polinômios ou curvas contínuas diferenciáveis. Uma descrição das estratégias de *Planejamento de trajetórias* são encontradas em [9, 70] e os esquemas de controle em [8].

Por outra parte, é definida como uma tarefa fundamental da robótica planejar movimentos livres de colisões para corpos complexos desde uma posição inicial até a posição meta entre um conjunto de obstáculos estáticos (que é uma definição formal de *Motion Planning*). Apesar da concepção relativamente simples, este problema geométrico de planejamento de rotas (*Path Planning*) é computacionalmente difícil [2, Cap.5]. Extensões desta formulação levam em conta os problemas adicionais dos sensores e limitações mecânicas reais dos robôs, o que complicam ainda mais o desenvolvimento de planejadores automáticos. Uma descrição moderna do que é o *Planejamento dos Movimentos* é encontrada em [71].

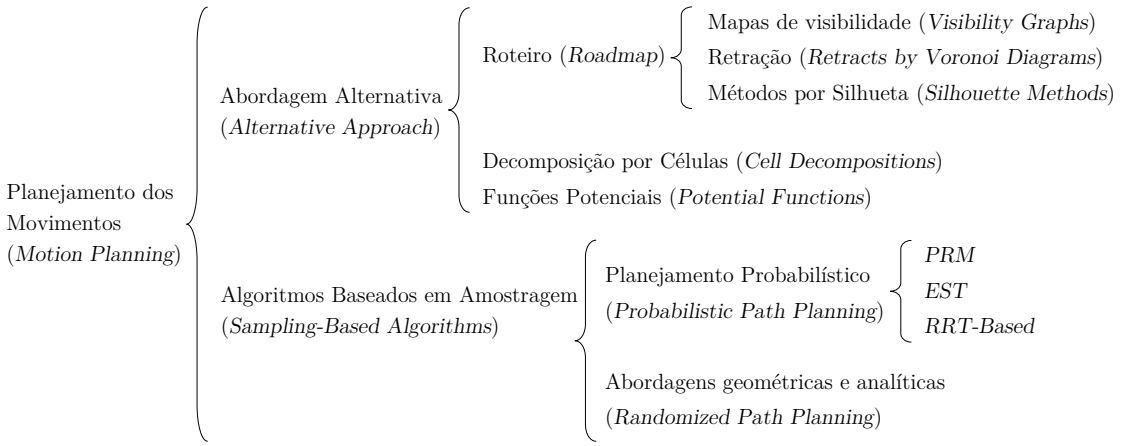


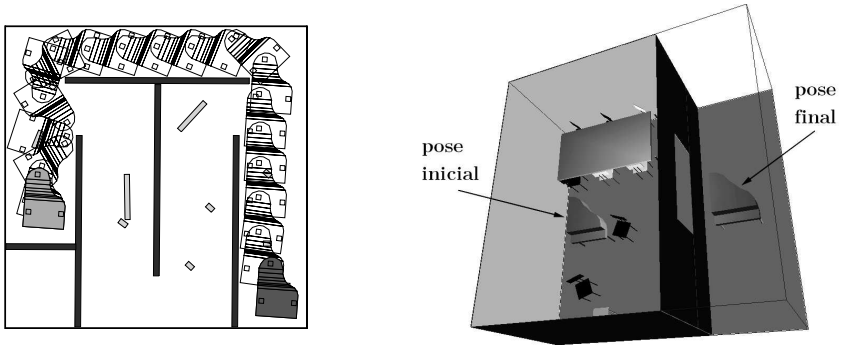
Figura 2.9: Agrupamento dos métodos desenvolvidos para o Planejamento do Movimento.

Com o fim de esclarecer a terminologia circundante deste tópico central da tese, na Fig. 2.9 é ilustrada uma hierarquia dos diversos métodos dentro do *Planejamento do Movimento*. Abaixo e nas seções seguintes é apresentado um resumo da ideologia por trás dessas estratégias e enfatizam-se aquelas que diretamente são utilizadas no desenvolvimento deste trabalho.

2.4.1 Problema Canônico

Um dos conceitos da robótica ideal é aquele robô móvel capaz de transportar bagagem no aeroporto navegando entre obstáculos que podem ser fixos como, correias transportadoras e elementos de construção, ou moveis, como pessoas e trabalhadores. O planejamento do movimento decide qual é o caminho que o robô deve seguir a fim de executar uma tarefa de transferência de uma postura inicial até uma final sem colidir com os obstáculos. Obviamente, gostaríamos de dotar o robô com a capacidade de planejar de forma autônoma seu movimento a partir de uma descrição de alto nível da tarefa fornecida pelo usuário e uma caracterização geométrica do espaço de trabalho. Esse espaço é disponibilizado inteiramente com antecedência (planejamento off-line) ou reunido pelo próprio robô durante seu movimento usando os sensores a bordo (planejamento on-line). No entanto, o desenvolvimento dos métodos automáticos para o planejamento de movimento é até agora uma tarefa difícil de realizar.

O problema canônico pode ser descrito formalmente da seguinte maneira: considere um robô $\mathcal{A}$ que pode consistir em um único corpo rígido (robô móvel), de uma cadeia cinemática de base fixa (manipulador padrão) ou de uma combinação dos dois primeiros (manipulador sobre trilhos ou manipulador móvel). O robô se movimenta em um espaço Euclidiano $\mathcal{W} = \mathbb{R}^N$ chamado *espaço de trabalho* (*workspace*), com $N = 2$ ou 3. Sejam os obstáculos $\mathcal{O}_1, \ldots, \mathcal{O}_p$ definidos no $\mathcal{W}$, por exemplo corpos rígidos fixos. Assumindo que são conhecidas as geometrias de $\mathcal{O}_1, \ldots, \mathcal{O}_p$, $\mathcal{A}$ e as posições de $\mathcal{O}_1, \ldots, \mathcal{O}_p$ em $\mathcal{W}$, além de supor que $\mathcal{A}$ se movimenta livre sem restrições cinemáticas, o problema de *Planejamento do Movimento* é como se segue: dada a pose inicial e final de $\mathcal{A}$ em $\mathcal{W}$, descobrir se existe um caminho, ou seja, uma sequência de poses do robô entre as poses inicial e final, evitando colisões; e informar se tal caminho não existe [8, Cap.12].



(a) (b)

Figura 2.10: Exemplos do problema clássico de trasladar um piano em dois ambientes. (a) Em 2D (imagem extraída de [72]). (b) Em 3D (imagem modificada de [73]).

A descrição clássica do problema da planejamento do movimentos em $\mathcal{W} = \mathbb{R}^2$ é apresentado na Fig. 2.10(a) e para o caso do $\mathcal{W} = \mathbb{R}^3$ na Fig. 2.10(b), onde é preciso trasladar um piano de um quarto a outro.

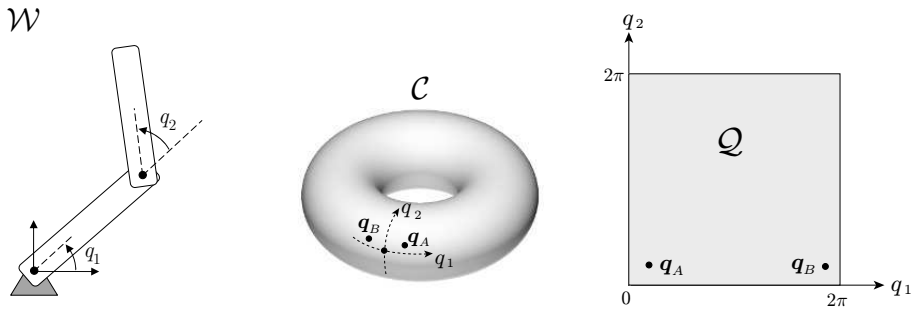
2.4.2 Espaço de Configuração

Uma das ferramentas poderosas para o desenvolvimento dos algoritmos de geração de rota é o conceito de espaço de configuração. Este conceito partiu dos estudos feitos na década de 1980 por Lozano-Pérez [74]. Porém, o conceito unificado no Planejamento de Movimentos foi apresentado pela primeira vez no livro de Latombe [75].

Para resolver o problema do Planejamento de Movimentos é necessário estabelecer o posicionamento de cada ponto da geometria do robô $\mathcal{A}$ no $\mathcal{W}$, onde essa condição é chamada de configuração do robô ou $\mathbf{q}$. Por exemplo, para um robô móvel planar dentro de $\mathcal{W} \in \mathbb{R}^2$, sua configuração está determinada pelas duas coordenadas e pelo parâmetro de orientação, portanto, seu vetor de configuração é $\mathbf{q}_{(3\times1)}$. Em geral, para o caso de manipuladores, o posicionamento dos elos e do EF são determinados pela cinemática direta usando os GDL do robô (Eq. 2.13), então uma configuração do manipulador é o mesmo vetor das variáveis das juntas $\mathbf{q}_{(n\times1)}$, onde n é o numero de GDL.

O *Espaço de Configuração* ou $\mathcal{C}$ é o conjunto de todas as configurações que o robô consegue atingir. A dimensão do espaço de configuração é o número de GDL do robô, ou o número mínimo de parâmetros necessários para especificar uma configuração. Matematicamente um espaço de configuração é o produto cartesiano dos espaços singulares que constituem uma configuração $\mathbf{q}$.

Para ilustrar esta definição, considere o manipulador planar 2-GDL da Fig. 2.11(a), cada ângulo de junta q_i corresponde a um ponto no circulo unitário S^1,



(a) (b) (c)

Figura 2.11: Exemplo de espaço de configuração. (a) Manipulador planar 2-GDL. (b) Representação topológica do espaço de configuração. (c) Representação em 2D do espaço de configuração. (Imagens extraídas de [8]).

por conseguinte, o espaço de configuração é $S^1 \times S^1 = S^2$, um toroide bidimensional. É comum imaginar um toroide como a superfície de uma rosquinha, pois o toroide está embebido em $\mathbb{R}^3$, assim como S^1 é incorporado naturalmente em $\mathbb{R}^2$. Ao cortar o toroide ao longo das curvas $q_1 = 0$ e $q_2 = 0$ na Fig. 2.11(b), podemos alterar o toroide ao plano, como é mostrado na Fig. 2.11(c). Esse tipo de representação planar e chamada também de *Espaço de Configuração* e é designada como $\mathcal{Q}$ para ser diferenciada da adequada representação topológica $\mathcal{C}$. Observe-se que nas Fig. 2.11(b) e 2.11(c) o robô é representado como um ponto duas vezes, na configuração $\mathbf{q}_A$ e na configuração $\mathbf{q}_B$; sendo cada configuração um vetor $\mathbf{q} = [q_1, q_2]^T$.

A importância do conceito abstrato de espaço de configuração é a generalização de um robô como um ponto em um espaço n-dimensional. Deste modo, os algoritmos de geração de rotas livres de colisões podem ser aplicados para qualquer robô, pois trabalham projetando rotas para um único ponto que se movimenta dentro de esse espaço. Por exemplo, um robô móvel espacial de 6-GDL tem um espaço de configuração 6-dimensional (três parâmetros de translação e três de rotação), ainda bem que seu respectivo $\mathcal{Q}$ não possa ser representado, o robô como um ponto pode ser levado de uma configuração $\mathbf{q}_A$ até $\mathbf{q}_B$ nesse espaço. Uma explicação extensa do conceito de espaço de configuração e suas topologias são apresentadas em [11, P.II] e [12, Cap.3].

A Fig. 2.12 mostra a generalização do Planejamento de Movimentos efetuado no espaço de configuração. $\mathcal{Q}_{free}$ indica a porção de espaço de configuração que é livre de colisões, $\mathcal{Q}_{obs}$ é a representação no $\mathcal{Q}$ de um obstáculo existente no ambiente do robô. Percebe-se que a solução do problema de Planejamento de Movimentos e uma rota contínua sem colisões de $\mathbf{q}_{start}$ até $\mathbf{q}_{goal}$, esta rota no $\mathcal{W}$ são os distintos movimentos contínuos do corpo ou corpos do robô.

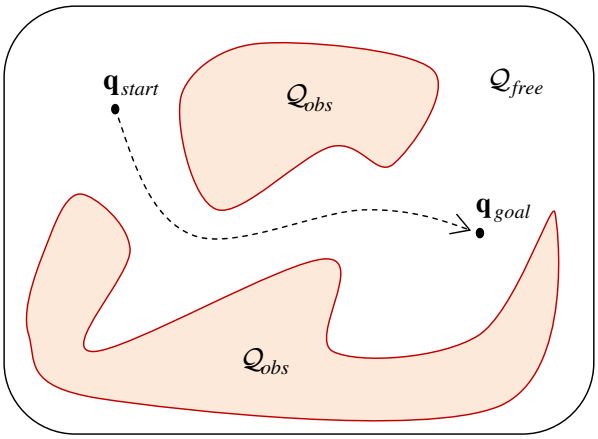


Figura 2.12: Problema básico de planejamento de movimento. Solução conceitualmente simples usando ideia de espaço de configuração.

2.4.3 Abordagens Alternativas

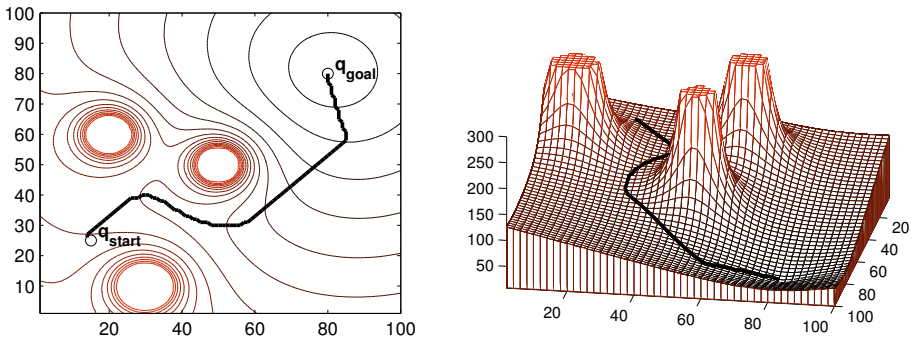
Roadmap Methods. As abordagens alternativas apresentados na Fig. 2.9 usam o espaço de configuração de diversas maneiras para resolver o problema de planejamento de movimentos. Nas metodologias tipo *Roteiro* o algoritmo planejador encontra rotas parciais no $\mathcal{Q}_{free}$. Logo juntando todos os caminhos, o planejador constrói um mapa que depois é usado para chegar de uma configuração à outra da mesma maneira que as pessoas fazem com seus veículos nas cidades.

Nos métodos tipo *Roteiro*, os *Mapas de visibilidade* tendem a ser aplicados aos espaços de configuração com obstáculos poligonais. Os nós do mapa são os vértices dos polígonos, dois nós dentro de um *mapa visibilidade* compartilham uma borda se os vértices correspondentes estão na linha de visão um do outro. Por outro lado, as *Retrações de deformação* são análogas ao derretimento do gelo ou queima de pastagem. Quando um desses processos ocorre, aparece um contorno que se afasta da fonte inicial. Estes contornos são aproveitados como espaços livres de colisão para gerar o mapa. A representação usada para *métodos silhueta* é construída projetando repetidamente uma sombra do espaço livre multidimensional do robô em espaços de menor dimensão até que é formada uma rede unidimensional.

Cell Decompositions. Também temos um tipo diferente de representação do espaço livre chamado de *decomposição por células*. Estas estruturas representam o espaço livre pela união das regiões simples chamadas células. Os limites comuns de células muitas vezes têm um significado físico, tal como uma mudança no obstáculo mais próximo, ou uma alteração na linha de visão entorno dos obstáculos. Duas células são adjacentes se eles compartilham uma fronteira comum. Um gráfico de adjacência, como o próprio nome sugere, codifica as relações de adjacência das células, onde um nó corresponde a uma célula e uma borda liga nós de células adjacentes. Assumindo que a decomposição é calculada, o planejamento de caminhos com uma decomposição celular é normalmente feito em duas etapas: em primeiro lugar, o planejador determina as células que contêm a configuração inicial e final respectivamente; em seguida, o planejador procura a rota dentro do gráfico de adjacência para juntar as duas configurações.

Cada um dos métodos anteriores é baseado em uma representação simbólica da geometria. Devido a isto, quando a dimensão do espaço de configuração cresce, estas estratégias tornam-se inviáveis para espaço de busca multidimensional, como é o caso dos manipuladores redundantes e sistemas multi-robôs [12, 76].

Potential Functions. Sem dúvida, um dos métodos mais populares é aquele que estabelece *Funções Potenciais* e foi descrito primeiramente em 1985 por Khatib [77]. Este método não precisa de uma forma explícita do espaço de configuração, usa o conceito de atração do robô à configuração final ao mesmo tempo em que é



(a) (b)

Figura 2.13: Planejamento de rotas usando funções potenciais. (a) Representação em isocurvas, o robô se move na presença de três obstáculos circulares. (b) Representação 3D em descida ao mínimo global.

repelido pelos obstáculos. Portanto, é necessário definir uma função potencial para a atração do alvo e de funções potenciais de repulsão devida a cada obstáculo. O robô como um ponto navega descida através de um vale constituído pela soma de funções potenciais (Fig. 2.13). Visto que o algoritmo está baseado no gradiente, a estratégia é susceptível a estagnação por mínimos locais.

Embora a maioria destas abordagens alternativas não sejam aplicáveis a ambientes multidimensionais, eles continuam sendo objeto de estudo, por exemplo, no caso dos gráficos de visibilidade seu estudo geral é apresentado em [78–80], para veículos autônomos em [81, 82] e em outras áreas em [83, 84]. Para as estrategias de retração de deformação tem-se os estudos [85–88]. Outro dos métodos populares de estudo são os Decomposição por Células, entre os recentes estudos temos [89–92] e diretamente para aplicações de cobertura de áreas com AUVs em [93, 94].

Provavelmente os métodos baseados em funções potencias artificiais são as estrategias mais estudadas, alguns trabalhos recentes são apresentados em [95–97] e na área de veículos aquáticos em [98–100]. Finalmente, também temos aqueles trabalhos que juntam duas metodologias, como são as funções potenciais e a descomposição por células que são mostradas em [101, 102].

2.4.4 Algoritmos Baseados em Amostragem

A idéia principal no planejamento baseado em amostragem (*Sampling-Based Algorithms*) é utilizar a detecção de colisão para determinar se uma configuração $\mathbf{q}$ (uma amostra) está no espaço livre de colisão $\mathcal{W}_{free}$ de maneira consecutiva. Com essa idéia, um planejador usa diferentes configurações (amostras) para a construção de uma estrutura de dados que armazena uma 1-D curva (rota) no espaço de configuração, que representa o caminho livre de colisão no espaço de trabalho.

Deste modo, um planejador baseado em amostragem não tem acesso direto aos

obstáculos no $\mathcal{C}$, mas apenas através da ação do detector de colisão em $\mathcal{W}$ e a estrutura de dados construída. Usando esse nível de abstração, os planejadores são aplicáveis a uma ampla gama de problemas.

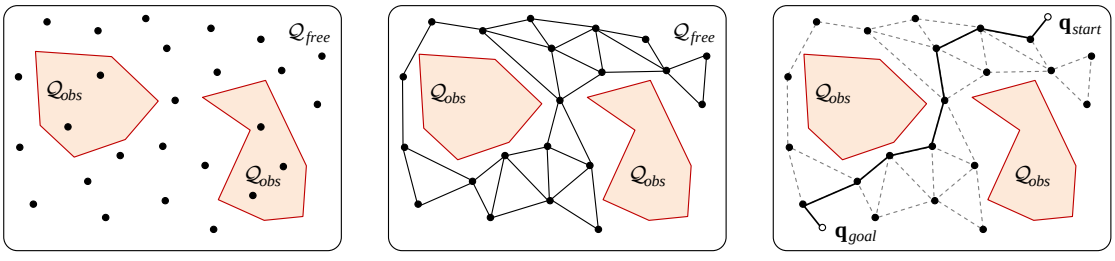
Os primeiros trabalhos neste contexto foram realizados no começo da década dos anos 1990 usando configurações aleatórias (*Randomized Path Planning*) posicionadas por algum critério geométrico ou heurístico, exemplos destes trabalhos são [103–105]. Estes métodos serviram como inspiração para os modernos algoritmos baseados em amostragem chamados como *Planejamento Probabilístico* (*Probabilistic Path Planning* na literatura inglesa).

2.4.5 Planejamento Probabilístico

Diferentes planejadores seguem diferentes enfoques de como realizar as amostras e o tipo de estruturas de dados que eles constroem. A classificação típica dos planejadores baseados em amostragem é feita em duas abordagens: de múltiplos questionamentos e de questionamento único.

Na primeira categoria, os algoritmos de múltiplos questionamento têm duas etapas: aprendizado e questionamento. Na primeira etapa os planejadores constroem um mapa G (*Roadmap*) que é pré-computado usando um sistema de amostragem no espaço livre de configuração e validando suas características. Após desta etapa, vários questionamentos do ambiente podem ser realizados e respondidos apenas com o mapa construído. Estas estratégias são comumente chamadas de *Mapa de Rotas Probabilístico* ou PRM do inglês *Probabilistic Roadmap Method*.

A Fig. 2.14 apresenta um resumo dos conceitos gerais dos PRMs. A Fig. 2.14(a) ilustra um ambiente bidimensional com dois obstáculos poligonais, estes obstáculos não são conhecidos no $\mathcal{Q}$ e são mostrados na figura para facilidade de entendimento visual. Também a Fig. 2.14(a) mostra a primeira etapa do PRM, onde o espaço de busca tem espalhadas amostras devido a alguma função de densidade. Logo são



(a) (b) (c)

Figura 2.14: Conceito geral de PRM. (a) Ambiente com amostras. (b) Representação topológica do mapa gerado. (c) Adição das configurações inicial e final. Apresenta-se uma rota usando a informação do mapa.

excluídas aquelas amostras não validas, isto é, aquelas amostras (configurações) que produzem colisões no $\mathcal{W}$. Depois usando um algoritmo de conexão entre amostras, também chamadas de nós, é criado o mapa apresentado na Fig. 2.14(b). Finalmente a Fig. 2.14(c) indica a segunda etapa do PRM onde as configurações $\mathbf{q}_{start}$ e $\mathbf{q}_{goal}$ são adicionadas o mapa e logo com um algoritmo roteador, como A* e Dijkstra, é encontrada a rota entre estas duas configurações usando a informação do mapa.

O algoritmo original do PRM foi introduzido no ano 1996 por Kavraki [106], posteriormente houve muitos trabalhos que tentam melhorar o desempenho do método. Um resumo destes estudos pode ser encontrado em [2, Cap.5] e [12, Cap.7].

Para a segunda categoria de classificação (dos métodos de Planejamento Probabilístico), os algoritmos de questionamento único usam uma estrutura de dados a maneira de arvore; adicionalmente, a fase de questionamento é realizada ao mesmo tempo do crescimento da estrutura. Ao contrario dos PRM, estes algoritmos tentam se concentrar em explorar progressivamente o espaço de busca para resolver um questionamento específico o mais rápido possível.

Existem vários algoritmos de questionamento único, os dois mais representativos são: *Árvores em Espaços Expansivos*, o EST (do inglês, *Expansive-Spaces Trees*), e *Árvores Aleatórias para Exploração Rápida*, o RRT (do inglês, *Rapidly-exploring Random Trees*). O EST proposto em [107] empurra a construção da árvore para partes inexploradas do espaço de configuração usando amostragem longe das áreas densamente amostradas. Por outro lado, o RRT, mais pratico e mais popular, utiliza uma estratégia de direção aleatória que puxa a árvore nas partes inexploradas do espaço de configuração. Portanto, em vista disso, o RRT foi selecionado para a abordagem desta tese, sendo apresentado com detalhe na seguinte seção. Uma boa descrição em língua portuguesa destes algoritmos e sua implicação na robótica móvel é apresentada em [108].

2.4.6 Árvores Aleatórias para Exploração Rápida

Ao contrario dos planejadores de múltiplos questionamentos, como o PRM, os planejadores de questionamento único não estão baseados na geração de um mapa que representa exaustivamente a conectividade do espaço de configuração livre de colisões $\mathcal{C}_{free}$. Na realidade, eles tendem a explorar apenas um subconjunto de $\mathcal{C}_{free}$ que é relevante para a solução do problema de planejamento do movimento. Isto resulta na redução do tempo necessário para calcular a solução; o algoritmo de Árvores Aleatórias para Exploração Rápida (RRT) faz isso excepcionalmente bem em ambientes multidimensionais.

Essa característica faz do RRT um algoritmo popular de geração de rotas nos últimos anos, além de sua concepção geral que se estende a outras áreas, tais como

Algoritmo 1: BUILD_RRT(q_{init})

```
1 T.init(q_init);
2 for k = 1 to K do
3 |   q_rand ← RANDOM_CONFIG();
4 |   EXTEND(T,q_rand)
5 return T
```

Figura 2.15: Algoritmo básico RRT (extraído de [111]).

a dobragem molecular [109] e as seqüências de montagem e desmontagem [110].

A idéia por trás do RRT é como segue: uma árvore T é constituída por nós e bordas, cuja raiz é a configuração inicial $\mathbf{q}_{init}$. Em cada iteração, de K iteraçoes, primeiro é escolhida uma configuração aleatória $\mathbf{q}_{rand}$, em seguida, determina-se o nó de T mais próximo a $\mathbf{q}_{rand}$, esse nó e chamado de $\mathbf{q}_{near}$. Logo partindo de $\mathbf{q}_{near}$ uma nova borda é expandida em direção de $\mathbf{q}_{rand}$. A ponta final da nova borda, $\mathbf{q}_{new}$, é uma configuração que será adicionada na árvore se sua borda não colide com obstáculo algum no espaço de trabalho. Eventualmente será gerado um nó suficientemente próximo da posição de destino, terminando o algoritmo.

O RRT foi concebido originalmente por LaValle e Kuffner em [112, 113]. A Fig. 2.15 mostra o pseudocódigo do RRT que utiliza recursivamente a operação EXTEND da Fig. 2.16. Explicação mais formal do funcionamento do RRT é como segue: a árvore de exploração T começa com a configuração de partida $\mathbf{q}_{init}$, logo usando a função RANDOM_CONFIG é gerada uma configuração aleatória $\mathbf{q}_{rand}$ de acordo com uma distribuição normal em $\mathcal{C}$, depois na operação EXTEND da Fig. 2.16 a função NEAREST_NEIGHBOR determina o nó $\mathbf{q}_{near}$ de T mais próximo a $\mathbf{q}_{rand}$ (percebe-se que na Fig. 2.16 $q \equiv \mathbf{q}_{rand}$). Em seguida, a função NEW_CONFIG é responsável por duas ações: primeira, escolhe dentro do $\mathcal{C}$ uma configuração $\mathbf{q}_{new}$,

Algoritmo 2: EXTEND(T, q)

```
1 q_near ←NEAREST_NEIGHBOR(q, T);
2 if NEW_CONFIG(q, q_near, q_new) then
3 |   T.add_vertex(q_new);
4 |   T.add_edge(q_near, q_new);
5 |   if q_near = q then
6 |   |   return Reached;
7 |   else
8 |   |   return Advanced;
9 return Trapped
```

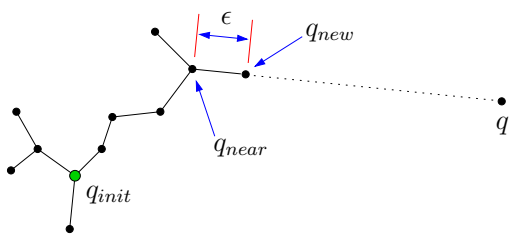


Figura 2.16: Operação EXTEND (figura original extraída de [111]).

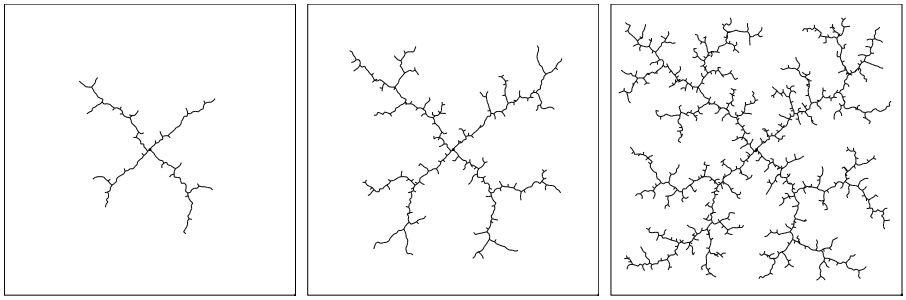


Figura 2.17: Crescimento da árvore de busca pelo algoritmo RRT básico.

em direção a $\mathbf{q}_{rand}$ desde $\mathbf{q}_{near}$ com uma separação de comprimento ϵ. Adicionalmente, a segunda ação de NEW_CONFIG é validar a configuração $\mathbf{q}_{new}$ e sua borda até $\mathbf{q}_{near}$. Para isso, externamente ao RRT é preciso usar um algoritmo de detecção de colisões no $\mathcal{W}$, pois os obstáculos não estão definidos no $\mathcal{C}$. Se NEW_CONFIG aprova $\mathbf{q}_{new}$ e sua borda no passo 2 da Fig. 2.16, são adicionadas as informações do novo nó e da nova borda na árvore T no passo 3 e 4 da Fig. 2.16. Se a configuração $\mathbf{q}_{rand}$ está a uma distância menor que o parâmetro de crescimento ϵ, tem-se $\mathbf{q}_{new} = \mathbf{q}_{rand}$, logo é retornado um sinal de nó alcançado (*Reached*) no passo 6; caso contrario, é retornado um sinal de estendido (*Advanced*) no passo 8. Finalmente, se $\mathbf{q}_{new}$ e sua borda são rejeitados por NEAREST_NEIGHBOR, a operação EXPAND retorna um sinal de falho no passo 9; nesta condição o algoritmo RRT prossegue com uma nova iteração no passo 2 da Fig. 2.15.

A Fig. 2.17 mostra como o RRT realiza a rápida exploração num espaço de configuração $\mathbb{R}^2$. No caso de haver obstáculos no espaço de trabalho $\mathcal{W}$, a estrutura da árvore permanece no $\mathcal{Q}_{free}$, a maneira de abraçando os $\mathcal{Q}_{obs}$ (Ver Fig. 2.18).

Originalmente o RRT foi desenvolvido para um planejamento *kinodinâmico*, isto é, levar em conta não só restrições cinemáticas mais também velocidades e ace-

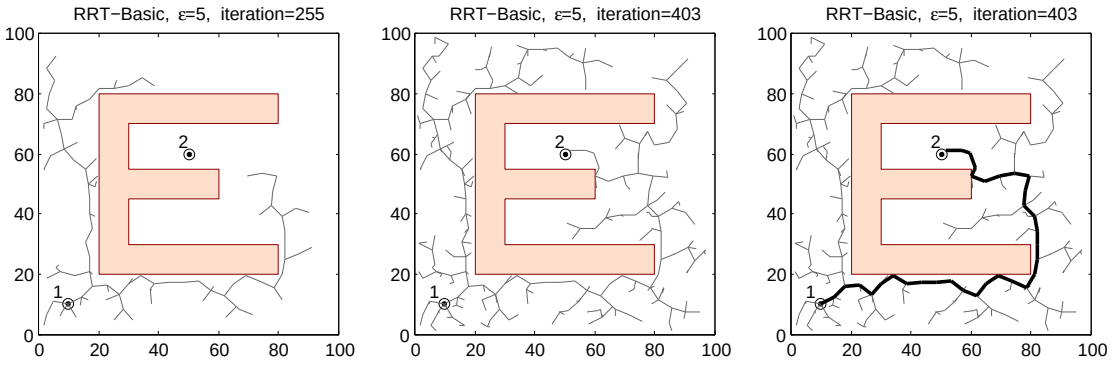


Figura 2.18: Crescimento do RRT-Basic. (a) e (b) Mostram crescimento no espaço livre de colisão. (c) Rota obtida.

lerações. Idealmente um robô móvel poderia cumprir esse propósito do RRT, encontrar uma rota em um espaço de estados em lugar do espaço de configuração. Mas as imprecisões no controle podem requerer um modelamento explícito da dinâmica do sistema para garantir trajetórias livres de colisões caso esta dinâmica seja muito rápida, como no caso dos helicópteros. A idéia do planejamento *kinodinâmico* com RRT é usar um estado x do robô definido por $x = (\mathbf{q}, \dot{\mathbf{q}})$, o espaço de estados $\mathcal{X}$ seria o conjunto de todos os estados possíveis do robô, e $\mathcal{X}_{obs}$ é uma representação dos obstáculos do $\mathcal{W}$ mais as restrições dinâmicas impostas, de similar maneira dos obstáculos mapeados como $\mathcal{C}_{obs}$ no espaço de configuração $\mathcal{C}$.

2.4.7 Algoritmo RRT-Connect

Uma melhoria substancial do algoritmo básico RRT é a versão de busca bidirecional que usa duas árvores de exploração com raízes nas configurações inicial e final. Este algoritmo chamado de RRT-Connect foi apresentado pelos mesmos autores em [111]. Esta versão não é aplicável no caso de espaço de estados. Ela é adequada para casos em que a dinâmica não é levada em consideração ou a dinâmica é considerada como uma etapa posterior do planejamento de rotas. Os bons resultados obtidos com o RRT-Connect tornaram-lhe um dos algoritmos mais populares de geração de rotas com muitas variantes que fazem do RRT o alvo de vários estudos.

Com tudo, as estratégias baseadas no RRT têm uma vantagem em relação ao PRM: o numero de nós usados. No PRM usa-se um número fixo de nós e um mesmo número de chamados de detecção de colisão. Para obter um mapa PRM com alta resolução, precisa-se de uma densidade maior de amostras, incrementado o tempo de computação. Por outro lado o RRT faz uma exploração incremental de nós que permite obter resultados rápidos; esta vantagem tende a aumentar com os graus de liberdade do sistema. Entretanto, ao comparar com os PRM, observa-se que a rota resultante com RRT possui ziguezagues, de modo que é requerido um pós-procesamento para suavizá-la e assim evitar movimentos desnecessários no espaço de trabalho.

A Fig. 2.19 ilustra o funcionamento típico do RRT-Connect. O ambiente é um $\mathcal{Q}$ bidimensional que poderia ser a representação do espaço de configuração do manipulador de 2-GDL da Fig. 2.11(a). Retornando à Fig. 2.19(a), o espaço de busca têm três obstáculos poligonais e as duas configurações de partida $\mathbf{q}_{start}$ e chegada $\mathbf{q}_{goal}$. Já na Fig. 2.19(b) e 2.19(c) são apresentados os crescimentos das duas árvores, tentando se aproximar uma à outra. Na Fig. 2.19(d) é indicada a rota encontrada pelo algoritmo. O passo de crescimento ϵ determina a qualidade da solução assim como o tempo de busca. Se ϵ é pequeno, os zigue-zagues são menores em comprimento, mas o tempo de computação é maior, pois a exploração é mais

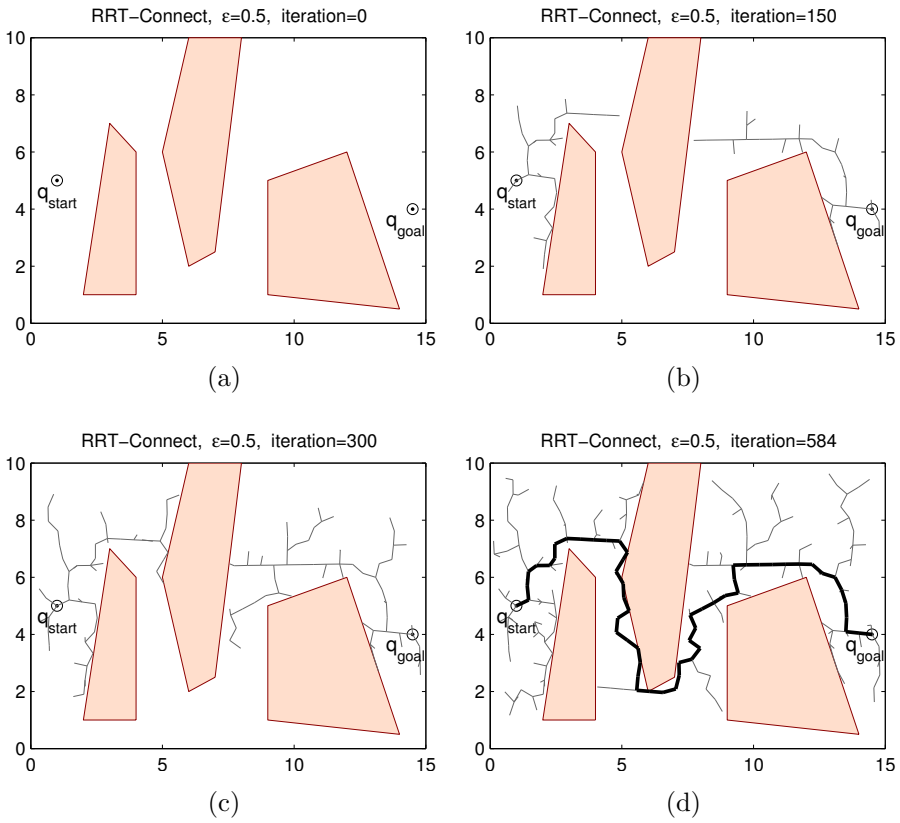


Figura 2.19: Funcionamento do RRT-Connect. (a) Espaço de busca bidimensional com três obstáculos representados por polígonos. (b) e (c) Manobras de aproximação entre arvores de exploração. Apresenta-se uma rota depois da conexão entre arvores.

intensa. No entanto, se ϵ é grande o tempo de computação é menor, mas, corre-se o risco de não descobrir uma rota entre dois obstáculos, esta condição e chamada de passagens estreitas. As passagens estreitas são possivelmente a maior desvantagem do RRT, pois se o ambiente é muito complexo ou só existe uma única rota de $\mathbf{q}_{start}$ até $\mathbf{q}_{goal}$ por uma passagem estreita, o algoritmo pode consumir muito tempo na entrega de uma solução e no pior dos casos, ficar estagnado. Contudo, o uso de um único parâmetro ϵ no algoritmo RRT permite-lhe ter uma robustez que poucos algoritmos alcançam.

O funcionamento do algoritmo RRT-Connect pode ser explicado de maneira lógica usando a Fig. 2.20 mediante os seguintes passos:

1. para cada iteração do algoritmo, tem-se duas árvores e uma delas é designada para crescimento normal RRT. Na Fig. 2.20(a) mostra as duas árvores com raiz nas configurações inicial e final. Neste caso a árvore com raiz em $\mathbf{q}_{goal}$ é designado para o crescimento;

2. Na Fig. 2.20(b) a árvore da direita exibe um crescimento de um novo nó. Esse nó foi criado a partir de uma amostra aleatória no $\mathcal{Q}$ como na Fig. 2.16;

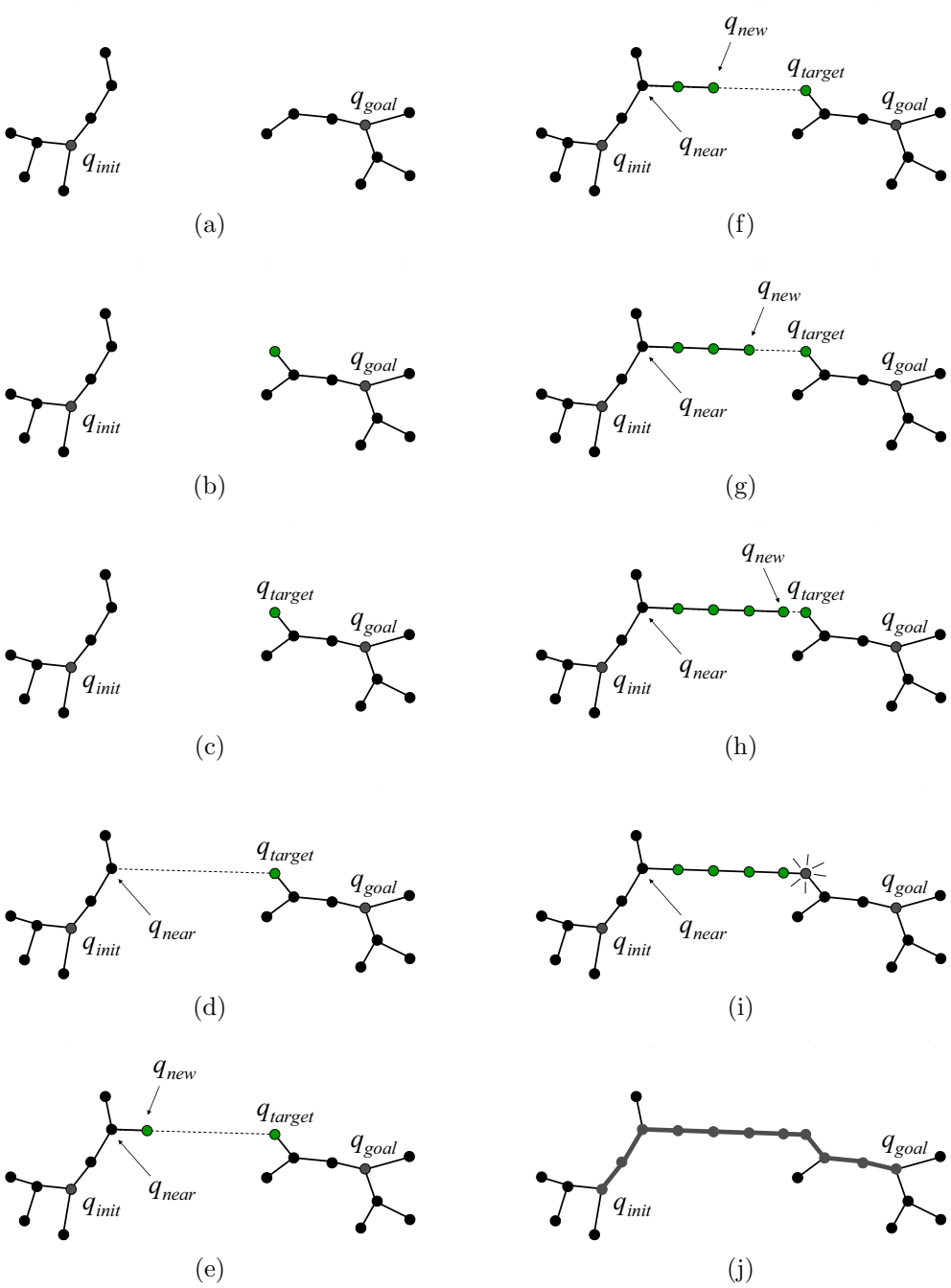


Figura 2.20: Ilustração do funcionamento do algoritmo RRT-Connect na conexão de árvores.

3. Esse novo nó $\mathbf{q}_{target}$ é agora alvo da árvore esquerda (Fig. 2.20(c));

4. Logo é encontrado o nó $\mathbf{q}_{near}$ que fica mais perto do nó alvo $\mathbf{q}_{target}$ (Fig. 2.20(d));

5. usando $\mathbf{q}_{target}$ tenta-se obter um novo nó $\mathbf{q}_{new}$ para a árvore esquerda (Fig. 2.20(e));

6. Da Fig. 2.20(f) até Fig. 2.20(h) é mantida uma sucessão de extensões em

Algoritmo 3: RRT_Connect(q_{init},q_{goal})

```
1 T_a.init(q_init); T_b.init(q_goal);
2 for k = 1 to K do
3     q_rand ← RANDOM_CONFIG();
4     if not EXTEND(T_a,q_rand) = Trapped then
5         if EXTEND(T_b,q_new) = Reached then
6             return PATH(T_b,T_b);
7     SWAP(T_a,T_b);
8 return Failure
```

Figura 2.21: Algoritmo RRT-Connect.

direção a $\mathbf{q}_{target}$;

7. Se é possível até atingir o nodo q_{target}, é realizada uma conexão entre as árvores (Fig. 2.20(i));

8. Finalmente, é traçada uma rota entre as raízes, como é mostrado na (Fig. 2.20(j)).

A formulação formal do RRT-Connect é apresentada na Fig. 2.21. O pseudocódigo mostra na linha 7 a operação SWAP, que é responsável pela troca das funções das árvores, de modo que em uma iteração a árvore T_a cresce normalmente e a árvore T_b tenta se conectar com T_a, logo na próxima iteração a árvore T_b cresce tipo RRT e a árvore T_a será quem tente se conectar com T_b.

O algoritmo RRT mostrou ser probabilisticamente completo em [113], por sua robustez e aplicabilidade em sistemas multidimensionais, podem se encontrar inúmeros estudos e variações dos métodos baseados em amostragem desenvolvidos na última década. Uma revisão que condensada as principais contribuições até ano 2014 é encontrado no texto de Elbanhawi e Simic [114].

Capítulo 3

Análise do Manipulador Submarino

As estruturas com um número elevado de graus de liberdade são matéria de interessante estudo na mecânica. Na robótica, os robôs redundantes seriais são um tema de consideração constante, e no caso dos manipuladores submarinos é relevante sua análise porque seu entendimento permite gerar novos desenvolvimentos e melhorar o controle destas máquinas.

O alvo deste capítulo é descrever as características cinemáticas do manipulador TITAN-4^{TM} da *Schilling Robotics*, qual foi eleito por ser o braço robótico freqüentemente utilizado entre os ROVs, além de fornecer uma estrutura comum entre os diversos manipuladores submarinos existentes. Especificamente é apresentado o estudo cinemático direto e inverso considerando a análise do Jacobiano e revelando a multiplicidade para a cinemática inversa.

(a) (b)

Figura 3.1: Sistema robótico TITAN-4^{TM} da *Schilling Robotics*. (a) Manipulador escravo . (b) Manipulador mestre.

3.1 Cinemática Direta

O TITAN-4 é um manipulador remoto servo-hidráulico construído principalmente de titânio, leve e com resistência estrutural elevada para evitar os danos causados por colisão. Suas características gerais são: alcance do sistema de manipulação de $1.922mm$, capacidade de carga de $122kg$, classificação de profundidade padrão de $4.000msw$. As especificações detalhadas podem ser encontradas na Internet no site da *Schilling-TITAN-4*. A Fig. 3.1 mostra a aparência externa do braço robótico (robô escravo) e do dispositivo do controle do operador (robô mestre). No Apêndice A são apresentadas as especificações dos cumprimentos padrões do manipulador, bem como o espaço de trabalho gerado pelos limites cinemáticos das juntas.

A cinemática direta é de fácil obtenção usando os tradicionais métodos descritos na seção 2.3.2. Particularmente, a Fig.3.2 indica os sistemas de coordenadas do manipulador e na Tabela 3.1 indica a matriz de parâmetros Denavith-hartenberg usando a notação padrão, com a qual é calculada a cinemática direta (Apêndice B).

Tabela 3.1: Matriz de parâmetros Denavith-Hartenberg (padrão) do TITAN-4.

i	a_i	α_i	d_i	θ_i
1	a_1	$\pi/2$	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3
4	a_4	$-\pi/2$	0	θ_4
5	0	$\pi/2$	0	$\theta_5 + \pi/2$
6	0	0	d_6	θ_6

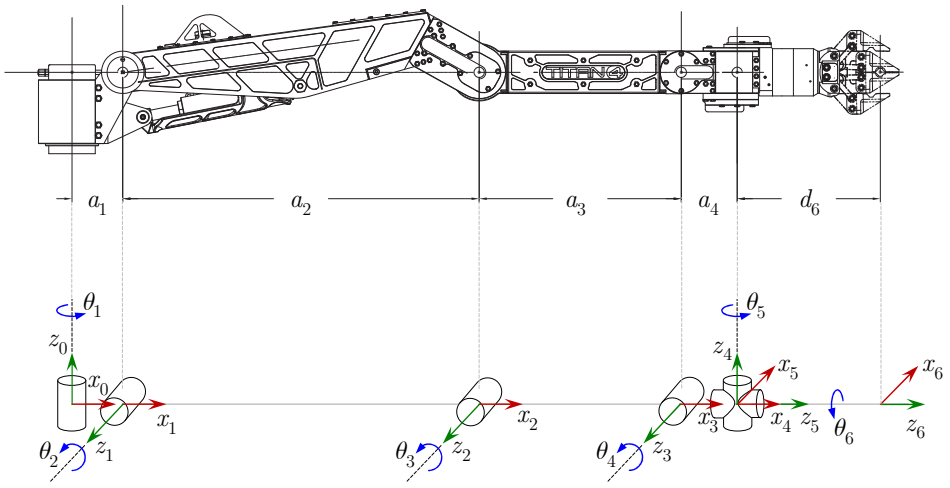


Figura 3.2: Descrição do sistema de coordenadas do manipulator TITAN-4, para obter a cinemática direta usando notação padrão Denavith-Hartenberg.

Ainda convém lembrar que o sistema de coordenas do EF, {e}, na *notação padrão*

é o mesmo que o sistema do último elo, {6}, ao contrario da *notação modificada* na qual os eixos de cada sistema tem a origem em sua respectiva junta.

3.2 Cinemática Inversa

A diferença com os tradicionais manipuladores industriais de 6-GDL, onde os três últimos GDL se intersectam em um mesmo ponto, o TITAN-4 não tem desacoplada a posição e orientação do EF no pulso. Isto é, somente os dos últimos GDL, θ_5 e θ_6 são coincidentes, situação que complica o cálculo da cinemática inversa.

Uma maneira prática de obter sua cinemática inversa é usando o analise por *Deslocamentos Sucessivos de Screws* (*successive screw displacements*) descrito em [61]. Na Fig. 3.3 ilustra o posicionamento dos eixos screw e a Tabela 3.2 mostra o posicionamento dos mesmos eixos com respeito a uma configuração de referência (posição zero) que pode ser arbitrária, e no caso, por conveniência, é a mesma configuração para o analise da cinemática direta; essa é a vantagem deste método.

Tabela 3.2: Matriz de posicionamento dos eixos screw para o TITAN-4.

i	$\mathbf{s}_i$	$\mathbf{s}_{oi}$
1	$(0,0,1)$	$(0,0,0)$
2	$(0,-1,0)$	$(a_1,0,0)$
3	$(0,-1,0)$	$(a_1+a_2,0,0)$
4	$(0,-1,0)$	$(a_1+a_2+a_3,0,0)$
5	$(0,0,1)$	$(a_1+a_2+a_3+a_4,0,0)$
6	$(1,0,0)$	$(0,0,0)$

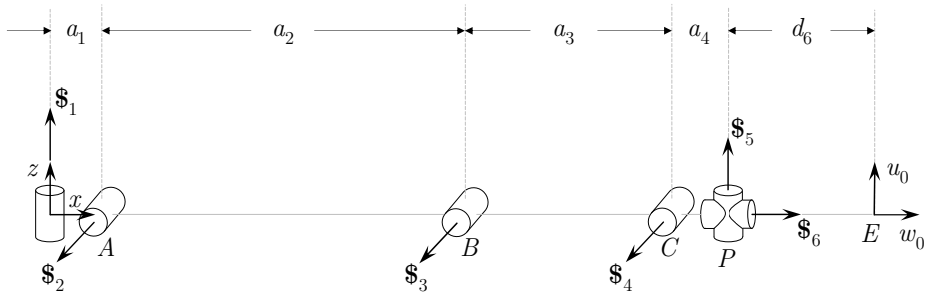


Figura 3.3: Posições de referencia dos eixos screw $\$_i$ para o TITAN-4.

O método começa definindo a orientação de referencia do EF, por conseguinte temos:

$$\mathbf{u}_0 = [0,0,1]^T, \qquad \mathbf{v}_0 = [0,-1,0]^T, \qquad \mathbf{w}_0 = [1,0,0]^T \tag{3.1}$$

Sendo os dois últimos screws coincidentes no ponto P, o vetor de referência é:

$$P_0 = [a_1+a_2+a_3+a_4, 0, 0]^T \tag{3.2}$$

Uma orientação qualquer do EF é definida como:

$$\mathbf{u} = [u_x, u_y, u_z]^T, \qquad \mathbf{v} = [v_x, v_y, v_z]^T, \qquad \mathbf{w} = [w_x, w_y, w_z]^T \tag{3.3}$$

E do ponto P será dada por:

$$P = [p_x, p_y, p_z]^T \tag{3.4}$$

Logo, aplicando o método são encontradas as seguintes matrizes de transformação A_i, onde percebe-se que não são as mesmas matrizes de transformação homogênea T_{ab}.

$$\begin{aligned}
A_1 &= \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (A_1)^{-1} = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
A_2 &= \begin{bmatrix} c_2 & 0 & -s_2 & -a_1(c_2-1) \\ 0 & 1 & 0 & 0 \\ s_2 & 0 & c_2 & -a_1 s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} c_3 & 0 & -s_3 & -(a_1+a_2)(c_3-1) \\ 0 & 1 & 0 & 0 \\ s_3 & 0 & c_3 & -s_3(a_1+a_2) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
A_4 &= \begin{bmatrix} c_4 & 0 & -s_4 & -(c_4-1)(a_1+a_2+a_3) \\ 0 & 1 & 0 & 0 \\ s_4 & 0 & c_4 & -s_4(a_1+a_2+a_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
A_5 &= \begin{bmatrix} c_5 & -s_5 & 0 & -(c_5-1)(a_1+a_2+a_3+a_4) \\ s_5 & c_5 & 0 & -s_5(a_1+a_2+a_3+a_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} e \quad A_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_6 & -s_6 & 0 \\ 0 & s_6 & c_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{3.5}$$

Sendo s_i equivalente a $\sin(\theta_i)$ e c_i equivalente a $\cos(\theta_i)$, com $i = 1, \ldots 6$. A transformação do ponto central do pulso P é dada por

$$\bar{P} = A_1 A_2 A_3 A_4 \bar{P}_0 \tag{3.6}$$

Multiplicando ambos os lados da igualdade por $(A_1)^{-1}$, obtemos

$$(A_1)^{-1} \bar{P} = A_2 A_3 A_4 \bar{P}_0$$

$$A_1{}^{-1}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = A_2A_3A_4\begin{bmatrix} a_1+a_2+a_3+a_4 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.7}$$

E substituindo $(A_1)^{-1}$ encontra-se as seguintes equações

$$p_xc_1 + p_ys_1 = a_1 + a_2c_2 + a_3c_{23} + a_4c_{234} \tag{3.8}$$
$$p_yc_1 - p_xs_1 = 0 \tag{3.9}$$
$$p_z = a_2s_2 + a_3s_{23} + a_4s_{234} \tag{3.10}$$

Sendo $s_{23} = \sin(\theta_2 + \theta_3)$, $c_{23} = \cos(\theta_2 + \theta_3)$, $s_{234} = \sin(\theta_2 + \theta_3 + \theta_4)$ e $c_{23} = \cos(\theta_2 + \theta_3 + \theta_4)$. Da Eq. 3.9 duas soluções de θ_1 são obtidas diretamente com

$$\theta_1 = \text{atan2}(p_y, p_x) \tag{3.11}$$

O robô não tem posição e orientação desacopladas, mas podemos usar a relação

$$R_1^T\mathbf{w} = R_2R_3R_4R_5\mathbf{w}_0 \tag{3.12}$$

Onde R_i indica a submatriz superior esquerda 3×3 de A_i (Matriz de rotação). Expandido a Eq. 3.12 obtemos

$$w_xc_1 + w_ys_1 = c_{234}c_5 \tag{3.13}$$
$$w_yc_1 - w_xs_1 = s_5 \tag{3.14}$$
$$w_z = s_{234}c_5 \tag{3.15}$$

Da Eq. 3.14 obtemos duas soluções de θ_5 (a segunda solução é $\theta_5 = \pi - \theta_5$)

$$\theta_5 = \text{asin}(-w_xs_1 + w_yc_1) \tag{3.16}$$

Conhecidas θ_1 e θ_5, da Eq. 3.13 e 3.15 obtemos

$$\theta_{234} = \text{atan2}(w_z/c_5, (w_xc_1 + w_ys_1)/c_5) \tag{3.17}$$

Reescrevendo a Eq. 3.8 e 3.10

$$a_2c_2 + a_3c_{23} = k_1 \tag{3.18}$$
$$a_2s_2 + a_3s_{23} = k_2 \tag{3.19}$$

Sendo $k_1 = p_xc_1 + p_ys_1 - a_1 - a_4c_{234}$ e $k_2 = p_z - a_4s_{234}$. Da soma dos quadrados

de Eq. 3.18 e 3.19 obtemos θ_3 (a segunda solução é $\theta_3 = -\theta_3$).

$$\theta_3 = \text{acos}\left(\frac{k_1^2 + k_2^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \tag{3.20}$$

Com o valor de θ_3 e usando as Eq. 3.18 e 3.19 é possível determinar θ_2 assim

$$\theta_2 = \text{atan}\left(\frac{(a_2 + a_3c_3)k_2 - a_3s_3k_1}{(a_2 + a_3c_3)k_1 + a_3s_3k_2}\right) \tag{3.21}$$

Sendo $\theta_{234} = \theta_2 + \theta_3 + \theta_4$, o valor $\theta_4 = \theta_{234} - \theta_2 - \theta_3$. Para resolver θ_6, podemos aplicar a transformação do vetor unitário $\mathbf{u}$:

$$(R_1R_2R_3R_4)^T\mathbf{u} = R_5R_6\mathbf{u}_0 \tag{3.22}$$

Expandindo a Eq. 3.22, obtemos

$$\begin{aligned} u_xc_1c_{234} + u_ys_1c_{234} + u_zs_{234} &= s_5s_6 && (3.23)\\ -u_xs_1 + u_yc_1 &= -c_5s_6 && (3.24)\\ -u_xc_1s_{234} - u_ys_1s_{234} + u_zc_{234} &= c_6 && (3.25) \end{aligned}$$

A obtenção de s_6 é usando as Eq. 3.23 e 3.24 asim

$$s_6 = s_5(u_xc_1c_{234} + u_ys_1c_{234} + u_zs_{234}) - c_5(-u_xs_1 + u_yc_1) \tag{3.26}$$

Das Eq. 3.25 e 3.26 obtemos uma única solução para θ_6

$$\theta_6 = \text{atan2}(s_6, c_6) \tag{3.27}$$

A obtenção de θ_1 até θ_6 é baseada na posição de P, a qual pode ser calculada com ajuda das matrizes de transformação homogênea obtidas a partir da cinemática direta. Na Fig. 3.2, P visto desde a origem da última junta $\{6\}$ é $P_6 = [0, 0, -d_6]^T$. E sendo um parâmetro de entrada para a cinemática inversa a pose do EF:

$$T_{06} = \begin{bmatrix} u_x & v_x & w_x & e_x \\ u_y & v_y & w_y & e_y \\ u_z & v_z & w_z & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.28}$$

Sendo $E = [e_x, e_y, e_z]^T$ as coordenadas desejadas do EF. Logo, usando a notação da Eq. 2.11, a posição do EF respeito à base $\{0\}$ é

$$\bar{P}_0 = T_{06}\bar{P}_6 \tag{3.29}$$

De este modo, as coordenadas precisadas do ponto P em notação simplificada $\bar{P} = T_{06}\bar{P}_6$ são:

$$\bar{P} = \begin{bmatrix} e_x - d_6 w_x \\ e_y - d_6 w_y \\ e_z - d_6 w_z \\ 1 \end{bmatrix} \tag{3.30}$$

Consequentemente para o estudo da cinemática inversa, $p_x = e_x - d_6 w_x$, $p_y = e_y - d_6 w_y$ e $p_z = e_z - d_6 w_z$.

No entanto, a análise da cinemática inversa é fechada, e apresentam-se oito soluções possíveis, apresentando deficiências na obtenção de algumas soluções devido às singularidades próprias do robô. Uma análise destas singularidades é feita na seção seguinte.

3.3 Análise do Jacobiano

O estado de velocidade, $\dot{\mathbf{x}}$, do efetuador final pode ser expressado de diversas maneiras. Talvez as definições mais comuns são o *Jacobiano convencional* (também chamado de *analítico*) e o *Jacobiano baseado em screw*.

No *Jacobiano convencional* o estado da velocidade é expressado em termos da velocidade linear, $\boldsymbol{\upsilon}_e$, da origem do sistema de coordenadas do EF, $\{e\}$, e da velocidade angular do EF, $\boldsymbol{\omega}_e$. Lembre-se que com a notação padrão Denavith-Hartenberg, $n = e$.

$$\dot{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\upsilon}_e \\ \boldsymbol{\omega}_e \end{bmatrix} \tag{3.31}$$

O *Jacobiano baseado no screw* é definido em termos da velocidade angular do EF, $\boldsymbol{\omega}_e$, e da velocidade linear de um ponto de referência, $\boldsymbol{\upsilon}_0$, no EF que é instantaneamente coincidente com a origem do sistema de coordenadas no qual o screw é expressado.

$$\dot{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\omega}_e \\ \boldsymbol{\upsilon}_0 \end{bmatrix} \tag{3.32}$$

De acordo com a teoria apresentada em [61] um screw unitário para uma junta de revolução está definido como:

$$\hat{\$} = \begin{bmatrix} \mathbf{s} \\ \mathbf{s}_0 \times \mathbf{s} \end{bmatrix} \tag{3.33}$$

Seja $\dot{q}$ a intensidade do giro do screw, por conseguinte $\$ = \dot{q}\hat{\$}$. Onde $\dot{q} = \dot{\theta}$ para uma junta rotacional. As três primeiras coordenadas do screw, $\$$, representam a velocidade angular e as três últimas a velocidade linear.

Para um manipulador serial a cinemática instantânea de primeira ordem pode ser escrita como

$$\$_n = \sum_{i=1}^{n} \dot{q}_i \hat{\$}_i \tag{3.34}$$

Consequentemente, para uma notação padrão Denavith-Hartenberg, o estado da velocidade do EF

$$\dot{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\omega}_n \\ \boldsymbol{v}_0 \end{bmatrix} = \sum_{i=1}^{n} \dot{q}_i \hat{\$}_i \tag{3.35}$$

Observe-se que o Jacobiano é simplesmente o conjunto de screws unitários associados com os eixos das juntas do manipulador

$$J = \left[\hat{\$}_1, \hat{\$}_2, \ldots, \hat{\$}_n \right] \tag{3.36}$$

Convenientemente, usando esta formulação e o procedimento iterativo apresentado em [61], o jacobiano do manipulador TITAN-4 é:

$$J = \begin{bmatrix} s_{234} & 0 & 0 & 0 & 0 & s_5 \\ 0 & -1 & -1 & -1 & 0 & -c_5 \\ c_{234} & 0 & 0 & 0 & 1 & 0 \\ 0 & a_3 s_4 + a_2 s_{34} & a_3 s_4 & 0 & 0 & 0 \\ j_{(5,1)} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_4 + a_3 c_4 + a_2 c_{34} & a_4 + a_3 c_4 & a_4 & 0 & 0 \end{bmatrix} \tag{3.37}$$

Onde $j_{(5,1)} = a_1 + a_2 c_2 + a_3 c_{23} + a_4 c_{234}$.

3.3.1 Análise de Singularidades

As singularidades do TITAN-4 são reveladas com o determinante do jacobiano:

$$\det(J) = -a_2 a_3 j_{(5,1)} s_3 s_5 \tag{3.38}$$

Sendo $a_2 \neq 0$ e $a_3 \neq 0$, pode-se concluir que as singularidades aparecem quando pelo menos um dos três fatores s_3, s_5 ou $j_{(5,1)}$ são iguais a zero. Especificamente, o manipulador perde: a) 1-GDL se qualquer s_3, s_5 ou $j_{(5,1)}$ é igual a zero; b) 2- GDL se dois termos s_3, s_5 e $j_{(5,1)}$ é igual a zero simultaneamente, e c) 3-GDL se s_3, s_5 e $j_{(5,1)}$ são todos igual a zero. O manipulador não pode perder mais de três GDL.

A situação $s_3 = 0$ é satisfeita quando $\theta_3 = 0$ ou π. Nesta condição, os eixos x_1 e x_2 da Fig. 3.2 são colineares, fazendo que os elos dois e três produzam as configurações do braço recolhido ou estendido. Esses estados são considerados como singularidades limítrofes do espaço de trabalho pelo ponto de vista estrutural.

A situação $s_5 = 0$ é satisfeita quando $\theta_5 = 0$ ou π. Nesta condição, os eixos

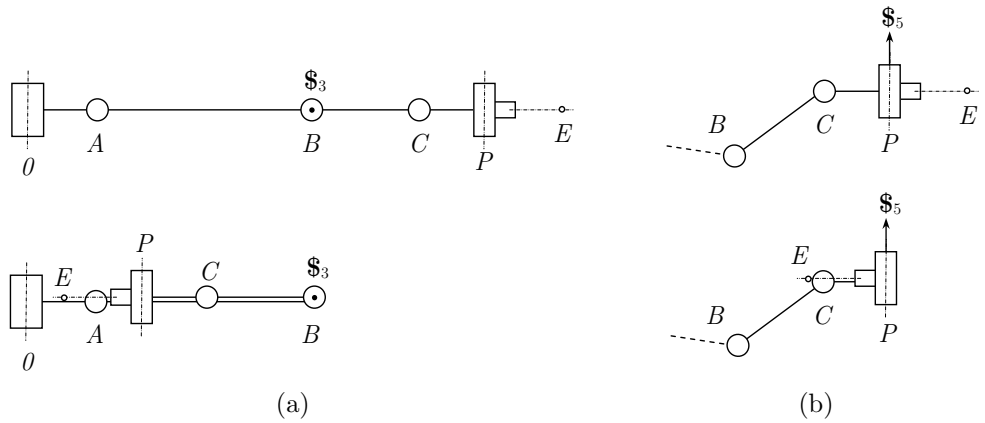


Figura 3.4: Representação linear em vista lateral do TITAN-4 em configurações singulares. (a) No caso $\sin(\theta_3) = 0$. (b) No caso $\sin(\theta_5) = 0$.

x_4 e z_5 de Fig. 3.2 são colineares, fazendo que o quarto elo (comprimento a_4) e o EF (comprimento d_6) produzam as configurações de pulso recolhido ou estendido. Estes estados ocorrem dentro do espaço de trabalho portanto são chamadas de singularidades interiores. Neste caso o manipulador perde 1-GDL. (Ver Fig. 3.4(b)).

A situação $j_{(5,1)} = a_1 + a_2c_2 + a_3c_{23} + a_4c_{234} = 0$ acontece quando os eixos da primeira e quinta juntas são coincidentes, neste caso qualquer movimento da segunda junta não gera movimento algum da posição do pulso, ponto P (Redundância cinemática para P).

3.3.2 Redundância e Multiplicidade da Cinemática Inversa

O Jacobiano também diz sobre as condições especiais do manipulador TITAN-4. Aproveitando a análise do determinante do jacobiano, temos para a condição $s_3 = 0$, uma condição especial em θ_3 com duas soluções na cinemática inversa. Como resultados são produzidas as clássicas posições de cotovelo acima e abaixo. Esta condição é evidenciada na Eq. 3.20.

Para um manipulador de 6-GDL, a execução de uma tarefa em 6-GDL é intrinsecamente não redundante [8]. Entretanto, o manipulador TITAN-4 possui multiplicidade para a cinemática inversa no caso de redundância do posicionamento do ponto P. Lembre-se que P é a intersecção dos eixos da sexta e quinta junta.

Para a situação $j_{(5,1)} = a_1 + a_2c_2 + a_3c_{23} + a_4c_{234} = 0$ a combinação dos valores dos ângulos θ_2, θ_3 e θ_4 produzem condições especiais. Estas condições são reveladas quando os eixos das juntas restantes, θ_5 e θ_6 são colineares ou interceptam o eixo da junta independente θ_1. Outra maneira de facilitar a visualização destas condições especiais é colocando o ponto P na linha de ação do eixo da junta independente θ_1. Consequentemente, temos os seguentes casos:

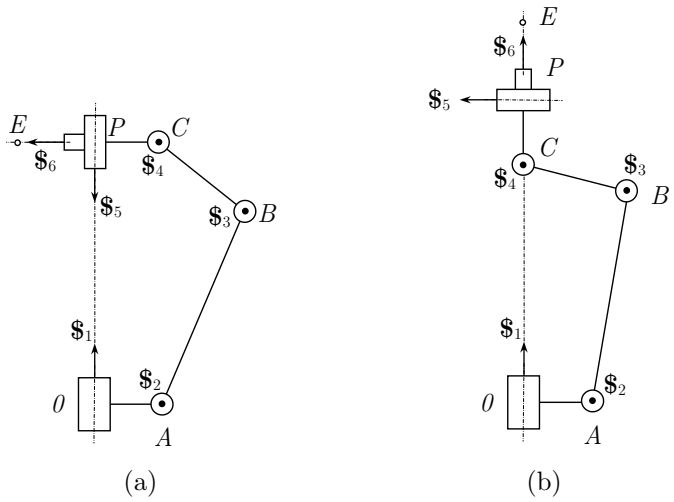


Figura 3.5: Representação linear em vista lateral do TITAN-4 em configurações com multiplicidade da cinemática inversa. (a) Caso 1. (b) Caso 2.

Caso 1. Quando o eixo da quinta junta é colinear ao eixo da primeira junta. Este caso é apresentado na Fig 3.5(a), onde percebe-se que para qualquer movimento da primeira junta, a quinta junta pode idealmente responder em sentido inverso proporcionando a mesma pose para o EF (cuja origem é o ponto E). Esta condição produz infinitas soluções da cinemática inversa evidenciada em um estudo mais detalhado da expressão da Eq. 3.7. Além disso, o ponto P pode atingir infinitas posições na linha de ação da primeira junta, produzindo infinitos casos de múltiplas soluções da cinemática inversa, tanto quando os comprimentos dos elos o permitam.

Caso 2. Este acontece quando o eixo da sexta junta é colinear ao eixo da primeira junta, além o eixo da quinta junta é intersectante. De maneira similar ao caso anterior, a multiplicidade da cinemática inversa aparece para um movimento livre da primeira junta em compensação com o giro da sexta junta sem produzir muda alguma na pose do EF (Fig. 3.5(b)).

Por outro lado temos um **Caso 3** de multiplicidade da cinemática inversa. A situação $s_5 = 0$ revelou singularidades internas, mas os valores do ângulo $\theta_5 = -\pi/2$ e $-\pi/2$ também produzem condições especiais que geram soluções indeterminadas da cinemática inversa quando é aplicada a Eq. 3.17. Para esclarecer este caso particular, e utilizando um plano z_0x' que é perpendicular aos eixos da segunda, terceira e quarta junta, Fig. 3.6(b). As origens destas três juntas são denotadas com as letras A, B e C na Fig. 3.6(a). Na notação padrão de Denavith-Hartenberg x' é paralelo ao eixo x_1. Em qualquer caso, o ângulo formado entre x_0 e x' é sempre θ_1.

A condição especial aparece quando o eixo da sexta junta e perpendicular ao plano z_0x', situação $c_5 = 0$. Neste caso é formado entre os pontos A, B, C e P um mecanismo de quatro barras, Fig. 3.6(a). Quando A, B, C e P não são colineares o mecanismo de quatro barras pode interagir livremente no plano z_0x'. A sexta junta

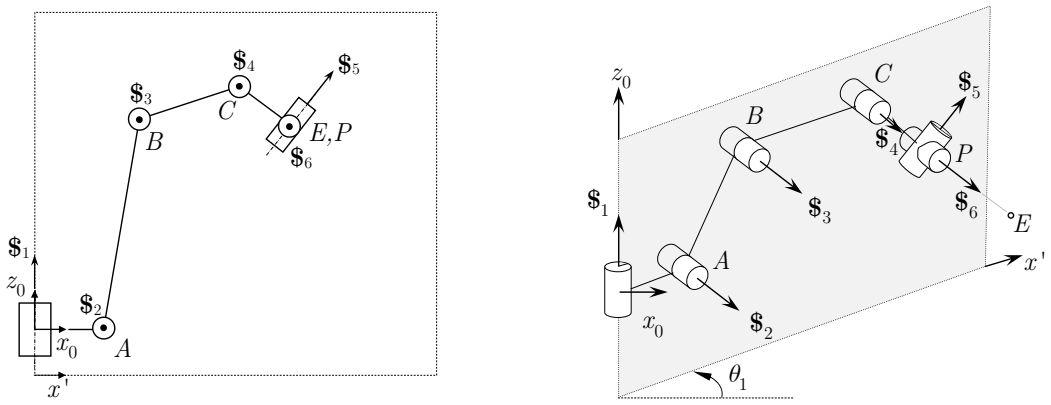


Figura 3.6: Representação do TITAN-4 com multiplicidade da cinemática inversa para o Caso 3, Entre A, B, C e P é formado um mecanismo de quatro barras. (a) Representação linear, os vetores $\$_{2,3,4,6}$ estão saindo do plano. (b) Vista em projeção.

preserva a orientação do EF e as coordenadas de E são garantidas pelo deslocamento da quinta junta, pois seu eixo permanece no plano z_0x'. Fig. 3.6(b).

O Caso 3 acontece com mais frequência que os dois casos anteriores, pois o ponto P não está restrito a uma linha coincidente com o eixo da primeira junta. No Caso 3 o ponto P aparece em um sólido de revolução do eixo da primeira junta, produzindo infinitas situações onde a cinemática inversa tem multiplicidade.

No caso real, onde o manipulador TITAN-4 é montado na base do ROV (ver Fig. 3.7), os Casos 1 e 2 são improváveis que aconteçam na teleoperação normal, mesmo que as restrições reais das juntas o permitam (Apêndice A). Isto se deve a visão obtida pelas câmeras não permite que um seguimento do EF seja movido até posições descritas para esses casos, onde o EF fica diretamente cima ou abaixo da primeira junta.

A Fig. 3.7 mostra que a estrutura da base do ROV, que limita a ocorrência dos

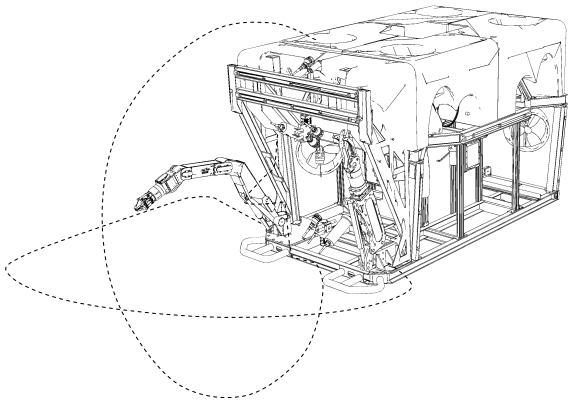


Figura 3.7: Representação do manipulador TITAN-4 montado na base de um ROV, com as linhas generatrizes do espaço de trabalho.

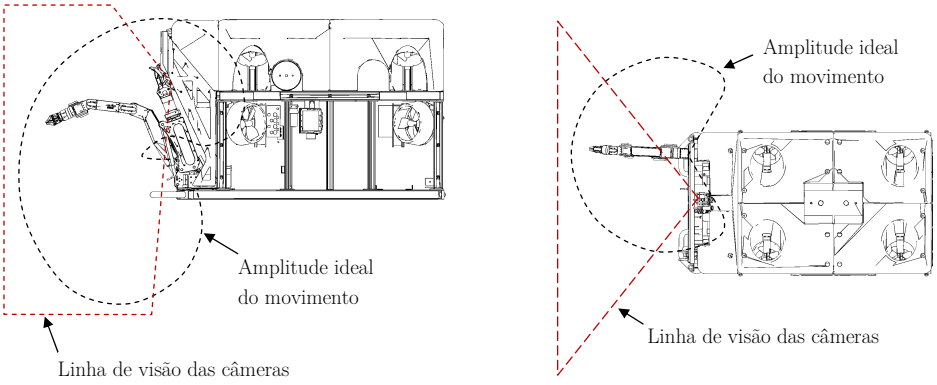


Figura 3.8: Representação do espaço efetivo de operação com a intersecção das áreas generatrizes de visão e movimentos. (b) Vista em lateral. (b) Vista de teto.

Casos 1 e 2 para quando o EF fica acima do eixo da primeira junta. Além disso, a Fig. 3.8 indica como a visão das câmeras interfere com o seguimento do braço robótico. Uma aproximação do espaço efetivo de operação do manipulador pode ser interpretada como a interseção das áreas limitadas pela *linha de visão das câmeras* e a *amplitude ideal do movimento*. Este espaço efetivo de operação fazem que o aparecimento de casos 1 e 2 sejam improváveis. Porem, o Caso 3 pode aparecer em diversas ocasiões, onde a quinta junta é deslocada $\pi/2$ respeito ao plano principal z_0x' do manipulador.

Capítulo 4

Aplicação de Realidade Aumentada

Como foi analisado no primeiro capítulo as ineficiências na manipulação remota com ROVs são um fator de atraso nas intervenções submarinas e impactam nos custos operacionais. A solução proposta neste trabalho para essa problemática é o uso de dois processos: o primeiro é um sistema de retorno sensorial aos pilotos; o segundo depende dos dados numéricos obtidos pelo primeiro sobre o ambiente real, para que o segundo processo sugira um planejamento dos movimentos do manipulador.

Este capítulo expõe a Realidade Aumentada como uma primeira estratégia de ajuda aos pilotos dos ROVs. A realidade aumentada, AR, é um processo de visão por computador que permite misturar objetos 3D gerados no computador com o vídeo extraído de uma câmera. AR já é comum em áreas do entretenimento e neste caso é tomada como uma estratégia de servovisão (Ver Capitulo 2).

A realidade aumentada foi designada neste trabalho como estratégia de aquisição de dados e retorno de controle aos pilotos principalmente pelas seguintes razoes: as condições adversas subaquáticas impedem na atualidade um sistema confiável de feedback dos sensores desde os sistemas mecânicos móveis que conformam o manipulador, além dos efeitos adversos do comportamento próprio do conjunto ROV (Capitulo 2). O uso somente da câmera como sistema sensor tem a vantagem de fácil interação com os pilotos. A implementação de novo hardware no manipulador é restrito, pois a maioria de fornecedores de braços robóticos submarinos impede essas manobras. A tecnologia AR já foi provada para diversas aplicações. As novas câmeras digitais de alto desempenho e o incremento do poder de processamento externo das imagens fazem destas tecnologias uma alternativa viável no futuro da robótica.

O capitulo está dividido em três secções, a primeira apresenta os conceitos básicos da realidade aumentada, a segunda mostra exemplos e características das linguagens de programação capazes de utilizar a AR, e finalmente o terceiro indica como foi

desenvolvida a ferramenta orientada no estudo de caso da tese.

4.1 Conceitos Básicos

O uso de objetos 3D gerados no computador para melhorar a percepção humana tem sido um tema de desenvolvimento contínuo por décadas. A pesquisa em Realidade Aumentada (AR) tem como objetivo desenvolver tecnologias que permitam a fusão em tempo real do conteúdo digital gerado por computador com o mundo real de maneira interativa para o usuário. Ao contrário da tecnologia de realidade virtual (VR), que imerge completamente os usuários dentro de um ambiente sintético, a realidade aumentada permite que o usuário veja os objetos virtuais tridimensionais sobrepostos ao mundo real. Tanto AR como VR são parte de um amplo termo denominado *Realidade Misturada* (*Mixed Reality*) conceituado por Milgram e Kishino em 1994 [115]. Como pode ser visto na Fig. 4.1, AR fica mais perto da realidade; no entanto a virtualidade aumentada (AV) fica mais perto de um ambiente imaginário como o pode ser a VR.

Uma sutil diferença entre AR e AV pode ser explicada como segue: AR tem a maior parte da realidade com uma pouca informação virtual em ela, e a AV tem a maior parte da virtualidade com uma pouca de realidade. Por outro lado, a VR é considerada como a imersão do humano em um ambiente totalmente artificial [116].

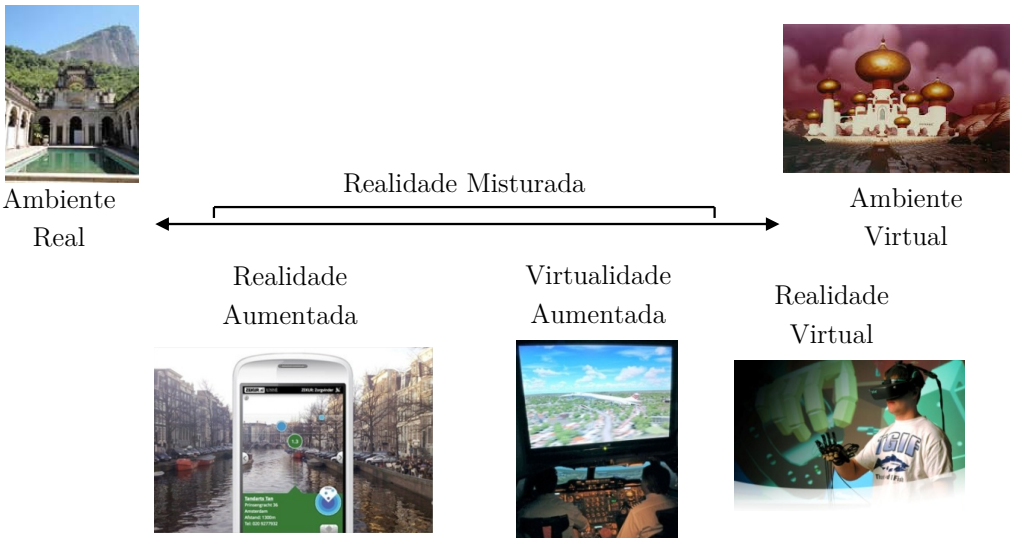


Figura 4.1: Conceito de continuidade realidade-virtualidade de Milgram [115].

A concepção geral da AR pode envolver outros sentidos como olfato e paladar e seus estudos começam desde a década de 1970. O conceito moderno, que usa imagens geradas por computador sobrepostos no vídeo, foi criado na década de 1990. O surgimento desta tecnologia em dispositivos móveis foi de apenas anos atrás e inúmeros artigos foram escritos na década passada. Ainda que a AR moderna esteja

em constante evolução alguns livros já apresentam um resumo destas tecnologias e estratégias, como em [117–119].

Sem dúvida, entre os diversos tipos de AR, as aplicações que usam objetos virtuais 3D sincronizados com o vídeo são as mais proeminentes. Essas aplicações têm a capacidade de inferir informação do espaço tridimensional real usando só as informações fornecidas pelos frames bidimensionais do vídeo. O presente trabalho aproveita essa característica para obter dados do ambiente e para que o robô possa interagir com os objetos da cena. Na Fig. 4.2 é mostrado o conceito geral para este tipo de realidade aumentada.

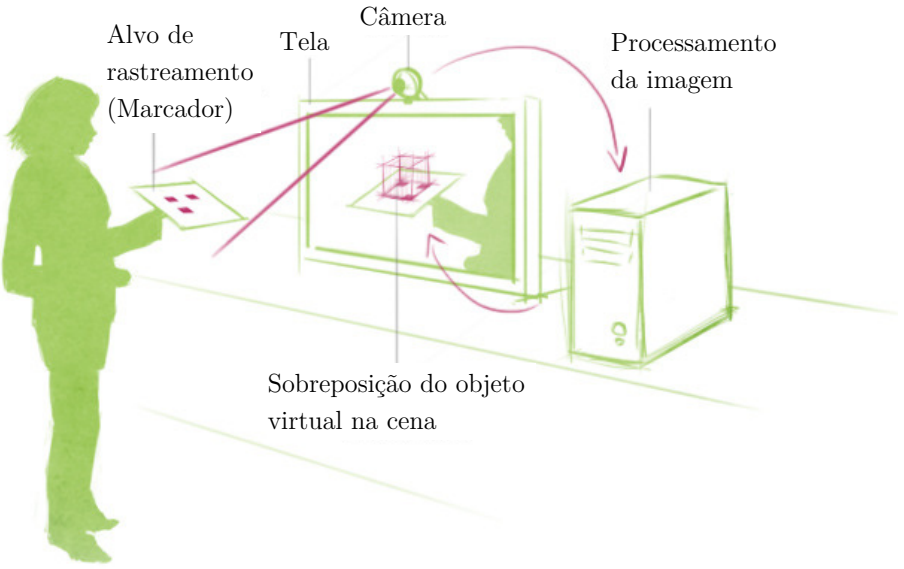


Figura 4.2: Princípios da Realidade Aumentada.[1]

A visão computacional mostra os objetos virtuais em 3D desde o mesmo ponto de vista que a câmera apresenta as imagens da cena real. Computacionalmente a AR tem dois processos: o processamento da imagem e a sobreposição de objetos. Normalmente estes processos são conhecidos como *Seguimento* e *Reconstrução*.

4.1.1 Seguimento e Reconstrução

Seguimento. Este processo é relacionado com o processamento da imagem visto na seção 2.2.1. São utilizados marcadores fiduciais, imagens ópticas, ou pontos de interesse como os usados na servovisão baseada na imagem (Secção 2.2.3). O *Seguimento* pode usar detecção de características, detecção de bordas, ou outro método de processamento de imagem. Na área da visão computacional, as técnicas de *Seguimento* podem ser separadas em duas classes: as baseadas em características e as baseadas em modelos. As baseadas em características trabalham descobrindo

[1]Imagem modificada de http://www.crossemedia.fr/publication-web/realite-augmentee

a ligação entre as características das imagens 2D e seus correspondentes sistemas de coordenadas global 3D [120]. Os métodos baseados em modelos usam as características distinguíveis dos objetos monitorados em relação com modelos CAD ou silhuetas [121].

Reconstrução. Uma vez outorgada a relação entre imagens 2D e um sistema global de coordenadas 3D, é possível encontrar a pose da câmera projetando as coordenadas 3D de uma característica dada dentro da imagem observada e minimizando o comprimento com a correspondente característica em 2D. Geralmente esse processo de estimação da pose da câmera é feito usando pontos específicos da imagem. A fase de *Reconstrução* utiliza os dados obtidos na primeira fase para reconstruir um sistema de coordenadas real e logo sobrepor os dados de objetos virtuais na cena transmitida ao usuário. Os processos de *Reconstrução* da AR são semelhantes aos processos de Extração de Características e Pose Estimada da servovisão baseada na posição apresentada Fig. 2.4.

O sucesso da Realidade Aumentada depende da maneira eficiente para estabelecer a posição e rotação de algum objeto real com respeito ao sistema de coordenadas da câmera usando os processos de Seguimento e Reconstrução. Em conseqüência disso, os pesquisadores e desenvolvedores de aplicativos usam as bibliotecas existentes, como ARToolKit (Ver Fig 4.3). ARToolKit foi desenvolvida em 1999 por Hirokazu Kato na Universidade de Washington [122], sua versão estendida é o ARToolKitPlus, que no entanto, não foi mais desenvolvida. Seu sucessor é Studierstube Tracker, que tem conceitos similares ao ARToolKit, mas seu código é completamente diferente, não é open source, nem disponível para descarregar. Sob o exposto, o trabalho desenvolvido nesta tese usa as bibliotecas de ARToolKit, além de sua compatibilidade com o software disponível no mercado e popularidade para realizar novas melhoras.

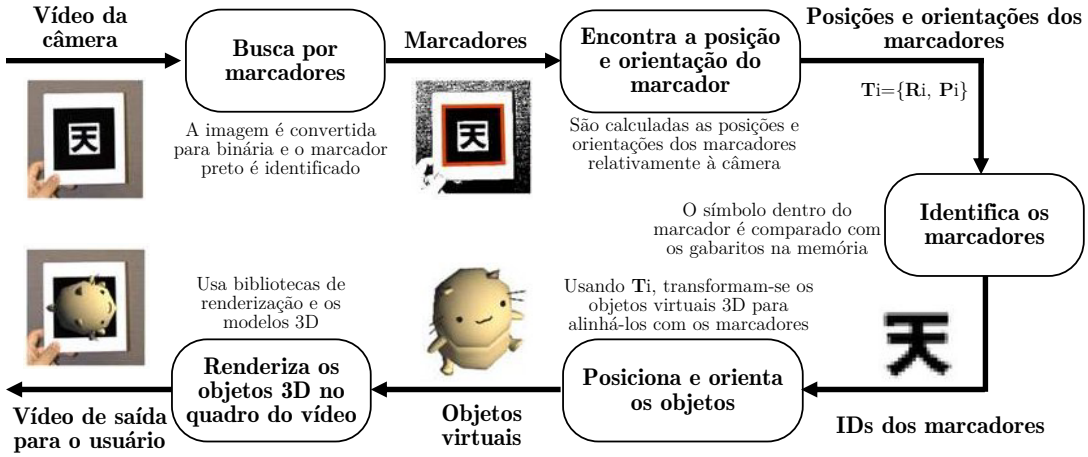


Figura 4.3: Funcionamento da AR usando ARToolKit.

(a) (b) (c)

Figura 4.4: Usos da AR. (a) Em catálogo iterativo da IKEA, ano 2013. (b) e (c) Reconstrução virtual das ruínas em Yuangmingyuan [124].

Convém lembrar que, na pratica, para qualquer aplicação AR, as condições físicas das lentes influenciam o processo de Reconstrução, Uma descrição da teoria por trás a realidade aumentada baseada no reconhecimento de marcadores e como é feita a calibração das câmeras é apresentada em [123].

4.1.2 Aplicações

Pode-se afirmar que onde existam as câmeras digitais, lá pode ser usada a AR, e a popularidade destes aparelhos fez com que AR fosse aplicada em todo tipo de dispositivos eletrônicos móveis. Como já foi dito, AR está em constante evolução e não há uma categoria das aplicações. No entanto, algumas as aplicações da tecnologia atual, que são rentáveis e que valem a pena ser mencionadas são apresentados a seguir:

Publicidade. A invasão da AR na publicidade é grande, uma aplicação representativa são as atrativas propagandas dos produtos nas capas e folhas internas das revistas. Um exemplo da capacidade desta tecnologia é apresentada no catálogo de móveis na Fig. 4.4(a), onde o usuário pode apreciar de outra maneira um produto com seu modelo em 3D. Por exemplo, uma mesa o sofá modular que somente ficava como imagem impressa no papel; além disso, o usuário pode interagir abrindo portas e gavetas dos móveis virtuais.

Entretenimento e educação. As aplicações para entretenimento e educação incluem aplicativos culturais como passeios em sítios turísticos e orientação em museus. Outro exemplo de aplicações práticas da AR é reconstrução virtual de ruínas, onde o usuário pode ver no sitio como era o aspecto de uma construção no passado [124]. Observe-se a Fig. 4.4(b) e 4.4(c).

Aplicações Móveis. Na atualidade a maioria das aplicações são pensadas para funcionar nos dispositivos móveis como smartphone e tablet. No entanto, ha aplicativos pensados exclusivamente para o usuário normal, por exemplo aqueles que usam os recursos de GPS e acelerômetro para oferecer informações das rotas e sítios

(a) (b) (c)

Figura 4.5: Usos da AR. (a) Para seleção e informações de produtos na lojas. (b) Linhas misturadas em jogo da liga NFL norte-americana (c) Informações visuais usadas nas transmissões de futebol na Copa das Confederações 2013.

de interesse como restaurantes e estações de serviço dentro das cidades [125]. Uma aplicação promissora da AR é a interatividade do usuário com os produtos disponíveis nas lojas, oferecendo, além do preço, as características e o total da conta (Fig. 4.5(a)).

Esportes Uma área que tem-se beneficiado do potencial da AR são os esportes. Por exemplo em transmissões televisivas ao vivo é possível mostrar na tela informações visuais sobreposta no campo de jogo (Fig. 4.5(c)). Inclusive, pode-se misturar linhas com os jogadores sem ser percebidas na realidade pelos o telespectadores, ver Fig. 4.5(b).

4.1.3 AR na Engenharia

No caso da engenharia, desenvolver aplicações industriais é mais complicado, pois tudo é centrado na precisão, a qual é uma desvantagem da atual tecnologia. Espera-se que com os avanços das câmeras digitais e dos algoritmos de processamento, no futuro próximo, seja possível utilizar a AR em aplicações que precisam de um controle formal como é acostumado na maioria de processo automatizados. Enquanto isso, os principais usos da AR estão limitados para os casos onde a precisão não é um fator determinante.

Varias aplicações da AR estão focadas para ao diagnostico ou como ajuda do planejamento de operações subseqüentes. Por exemplo no caso da engenharia civil, a visualização de tubulações no campo usando os desenhos CAD facilita as decisões dos projetistas e técnicos [126–128].

Na robótica, ainda que as aplicações práticas sejam escassas, no últimos anos tem-se uma acolhida da AR na pesquisa tanto para robôs móveis como para manipuladores. Por exemplo no caso da robótica aérea autonoma em [129], na simulação de operações com manipuladores em [130], para ajudar no posicionamento do efetuador final do manipulador montado na estação espacial [131]. De similar maneira foram realizados estudos centrados na teleoperação, como no caso do robô móvel

controlado desde uma tela táctil em [42].

4.2 Linguagens Para Geração de Aplicações

Como o desenvolvimento do aplicativo AR da tese é baseado nas bibliotecas ARToolKit, é de importância conhecer os diversos softwares e linguagens de programação, de modo que a compreensão dessas ferramentas pode ser de proveito para gerar novos desenvolvimentos.

Uma distinção não-formal entre diferentes linguagens de programação pode ser definida em três grupos: o primeiro compreende o software que é destinado aos usuários de todo tipo com conceitos elementares de programação. O segundo grupo são aquelas ferramentas destinadas a desenvolvimentos próprios da engenharia. E o último grupo é formado pelas linguagens de programação compiladas de propósito geral. Eventualmente os dois primeiros grupos são plataformas feitas sob o primeiro grupo; portanto, pode-se considerar o primeiro grupo mais perto ao usuário que o terceiro por seu nível de complexidade.

A continuação apresenta-se a análise de três ambientes experimentados, PROCESSING, MATLAB e C++ os quais são softwares representativos do primeiro, segundo e terceiro grupo respectivamente.

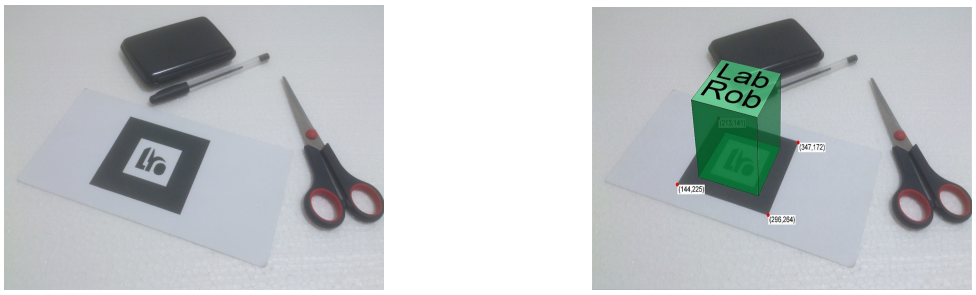
4.2.1 Ambiente PROCESSING

PROCESSSING tem programação baseada em Java, portanto, herda todas suas funcionalidades e sua maneira construtiva de processos. O software foi criado por Casey Reas e Ben Fry no ano 2001, ambos ex-membros do Grupo de Computação do Instituto Tecnológico de Massachusetts, MIT. Um dos objetivos do PROCESSING é atuar como uma ferramenta para aqueles desenvolvedores não-programadores, mas iniciados com a programação, através da satisfação imediata com um retorno visual. Isto é, com poucas bases em programação é possível obter aplicações com alta qualidade das imagens geradas.

Essa característica faz do PROCESSING[1] uma das linguagens mais populares entre o gênero das artes visuais, e que agora é espalhado nas faculdades de engenharia pela facilidade de programação e sua integração com a plataforma de prototipagem eletrônica de hardware livre ARDUINO[2].

Entre as vantagens de PROCESSING, temos: é software de livre distribuição e código aberto. Tem uma ampla ajuda online e boa documentação oficial de referência [132, 133]. Os colaboradores estão diversificados e produzem diversos *plugins*

[1]Mais informação do ambiente em http://www.processing.org/reference/environment/
[2]Mais informação em http://www.arduino.cc/



(a) (b)

Figura 4.6: Usos da AR em PROCESSING. (a) Condição inicial com marcador personalizado. (b) Inclusão da informação virtual.

(utilitários) para facilitar a interoperação com otras linguagens e bibliotecas como é ARToolKit.

ARToolKit não interage diretamente com o PROCESSING, pois as bibliotecas de ARToolKit ficam na linguagem C. Conseqüentemente, precisa-se de uma versão compatível em Java. Para esse fim, foi testada a biblioteca NyAR4psg iniciada em 2010 por Ryo Iizuka.

Além disso, como toda aplicação AR, é necessário um ambiente visual de entrada e saída do vídeo, e um renderizador para gerar objetos virtuais. Nesse sentido, foram usadas as bibliotecas: GSvideo[1] que fornece suporte de vídeo stream (reprodução, captura, e criação de arquivos de vídeo); e OpenGL[2] para PROCESSING, que é uma biblioteca de funções para geração de figuras 2D e 3D.

A Fig. 4.6 apresenta um dos testes em PROCESSING feitos com as bibliotecas mencionadas. A inclusão das bibliotecas na árvore de diretórios é representada na Fig 4.7 para a versão estável 1.5.1 do 2011.

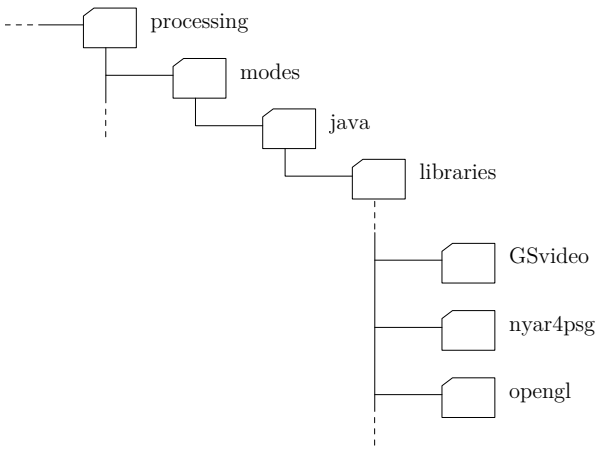


Figura 4.7: Árvore de diretórios das bibliotecas em PROCESSING para o uso da AR.

[1]De livre descarga em http://gsvideo.sourceforge.net/
[2]Mais informação em http://wiki.processing.org/w/Advanced_OpenGL

Com OpenGL é possível gerar sólidos simples de maneira rápida, mas a inclusão de objetos 3D mais complexos pode ser feita com a importação de modelos 3D feitos em software especializados. Para isto, foram testadas bibliotecas de importação para PROCESSING, mas elas produzem uma redução na velocidade de processamento, pois o PROCESSING deve lidar com diferentes processos, como interfaces de vídeo e a compatibilidade do ARToolKit, além disso, estas bibliotecas de importação são dependentes de OpenGL como sistema de renderização dos sólidos.

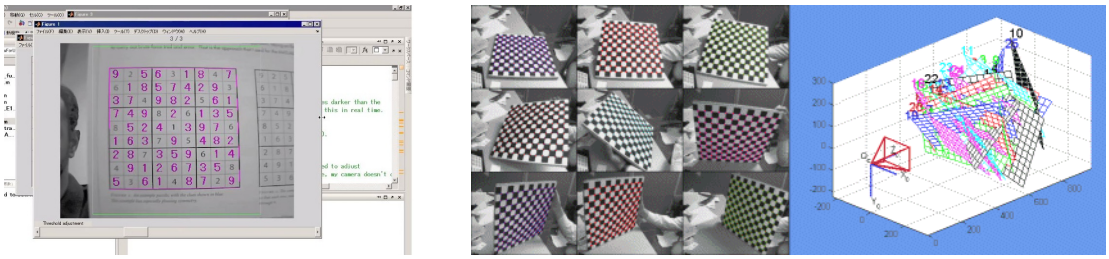
Em conclusão, o PROCESSING tem as vantagens de ser de livre acesso, de rápida obtenção de resultados na experiência com AR. As aplicações desenvolvidas podem ser portáteis e trabalhar em diversos sistemas operativos. Ainda que sob PROCESSING consegue-se trabalhar com sólidos 3D complexos, a aplicação sofre demoras e atrasos na visualização dos resultados. Tendo em vista aspectos observados nos testes, é recomendável usar o PROCESSING como sistema de AR no caso de inserção de informações virtuais simples e elegantes.

4.2.2 Ambiente MATLAB

MATLAB é um software conhecido por suas capacidades de computação numérica baseados em sua ampla gama de bibliotecas (toolbox) dirigidas ao desenvolvimento da engenharia. A realidade aumentada precisa de um ambiente visual para o usuário, nesse sentido, MATLAB possui dos ambientes: o ambiente *nativo* (comando *figure*) e o ambiente *VR* (do inglês Virtual Reality).

O ambiente *VR* do MATLAB é governado por seu toolbox baseado em *Virtual Reality Modeling Language* (VRML). A designação das letras *VR* é diferente ao conceito formal de Realidade virtual apresentado na seção 4.1. E Neste caso VRLM é mais um das linguagens para o intercambio de modelos 3D. Infelizmente o ambiente *VR* de MATLAB não permite a mistura entre as imagens transmitidas por uma câmera e os modelos 3D que precisam ser pré-carregados na memória do computador, impossibilitando os desenvolvimentos de AR.

Por outro lado, o ambiente *nativo* é capaz de receber imagens e sobrepor linhas em elas. Alguns trabalhos de AR já foram feitos sob MATLAB, mas esses são de âmbito acadêmico ou demonstrativo. Por exemplo, em [134] é realizado um processamento da imagem, reconhecimento de números, logo um algoritmo resolve um típico jogo matemático Sudoku, e finalmente incrementa a percepção do usuário sobrepondo a solução do Sudoku no ambiente *figure* de MATLAB que mostra em tempo real o vídeo que captura a folia de papel com o Soduku proposto (Ver Fig 4.8(a)). Nesse caso, a vantagem é a implementação direta do algoritmo que resolve o Sudoku com a AR, mas como não usa alguma biblioteca externa de AR, deve-se programar os processos de *Seguimento* e *Reconstrução* vistos na secçao anterior.



(a) (b)

Figura 4.8: Aplicativos para AR em MATLAB. (a) Solução do típico jogo Sudoku. (b) Processo de calibração de câmera usando múltiplas fotogramas.

Essa vantagem que tem MATLAB de interação entre a programação bocada na engenharia e seus toolbox permite desenvolvimentos como a metodologia de calibração de câmeras apresentada por Jean-Yves Bouguet[1]. O toolbox de Bouguet permite obter os parâmetros[2] de câmera necessários em qualquer aplicação AR. O sistema de servovisão (Por exemplo os parâmetros a da PBVS na seção 2.2.3), obtidos a partir de diversos fotogramas de vídeo em tempo real. Ver Fig. 4.8(b) .

Pode-se mencionar, que esse toolbox para calibração de câmeras tornou-se uma referência e ponto de partida de vários artigos. Por exemplo, no ano 2012, os autores de [135] propõem um método de calibração de câmeras de visão estéreo para navegação de robôs móveis.

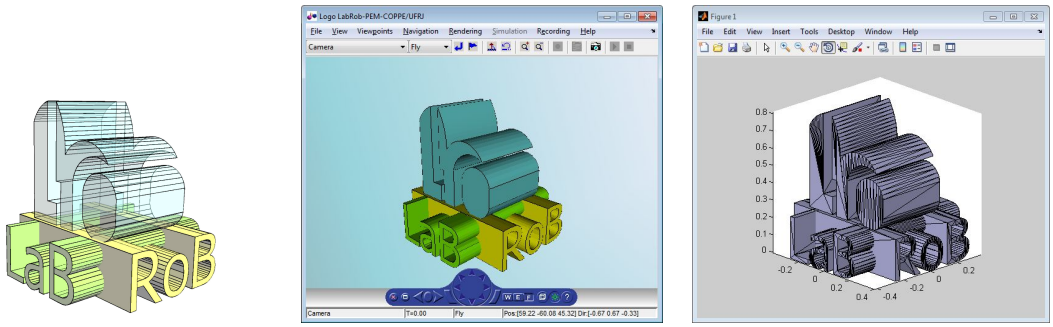
O ambiente MATLAB não tem uma interação direta com ARToolKit, isto em parte porque as bibliotecas de ARToolKit estão orientadas a trabalhar com OpenGL e não tem acesso direto ao ambiente *nativo* de MATLAB. Em [136] apresenta uma maneira de intercambiar informações entre ARToolKit e MATLAB usando o protocolo de transmissão de dados UDP. Os autores mostram como usar os dados de ARToolKit dentro de MATLAB como sistema de servovisão PBVS (Ver Fig. 2.4).

Nesse caso, a câmera esta montada no EE obtendo os dados de um marcador associado a um alvo, não há uma apresentação de sólidos virtuais em uma tela como as aplicações AR.

No entanto, a obtenção de sólidos é uma tarefa difícil no ambientes nativo de MATLAB, existem funções de livre acesso na Internet para a importação de modelos sólidos, más seu aspecto está longe das imagens bem-definidas obtidas com outros programas de visualização (Fig. 4.9(c)). Além disso, nos testes feitos com o movimento de sistemas com multicorpos, apresenta-se uma demora na renderização pelos sólidos complexos. Em resumo, ainda que MATLAB tenha a vantagem de programação estruturada com toobox que facilita a geração de aplicações na engenharia,

[1]Descrição do toolbox em http://www.vision.caltech.edu/bouguetj/calib_doc/

[2]Descrição matemática dos parâmetros em http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html



(a) (b) (c)

Figura 4.9: Visual dos sólidos para ambientes em MATLAB. (a) Desenho original. (b) Ambiente *VR*. (c) Ambiente *nativo*.

o ambiente visual é pobre para o desenvolvimento de aplicações de AR.

4.2.3 Ambiente C++

Provavelmente a maioria dos primeiros desenvolvimentos da AR foram realizados em linguagem C ou C++, como o ARToolKit originalmente escrita em C. Atualmente as versões modificadas de ARToolKit permitem realizar aplicações de maneira rápida para diferentes sistemas operacionais. Por exemplo, na versão ARTag que foi desenvolvida pelo *National Research Council of Canada*, o objetivo foi resolver alguns problemas encontrados na ARToolKit, principalmente no processo de detecção de marcadores. Tais problemas como: falso positivo, que ocorre quando o sistema acusa a presença de um marcador, mas ele não existe; o problema do falso negativo, quando o sistema não acusa a presença de um marcador, mas ele existe; e o problema de confusão, quando o marcador existe no ambiente e o sistema o identifica como sendo outro [137].

Um exemplo de migração de ARToolKit desde linguagem C a outro software é DART (do inglês *Designer's Augmented Reality ToolKit*) [138], que foi desenvolvido como um conjunto de extensões do ambiente de programação multimídia *Macromedia Director* (atualmente *Adobe Director*) no *Georgia Institute of Technology*. Ele é composto por extensões do Director escritas na linguagem LINGO e plugins escritos na linguagem C++. LINGO é usado como suporte para a captura de vídeo, rastreamento e para o processo de reconhecimento de marcadores.

Esta tese utiliza a biblioteca original de ARToolKit, de tal maneira que a análise da AR em linguagem C++ é baseado nela, deixando a teoria computacional de outros desenvolvimentos da AR fora do alcance deste trabalho.

Já na Fig. 4.3 foi apresentado um resumo do funcionamento de AR com ARToolKit. O processo é como segue: primeiramente ARToolKit transforma a imagem de vídeo capturada pela câmera em uma imagem com valores binários em branco

e preto. Em seguida, ele examina essa imagem para encontrar regiões quadradas. Então, o ARToolKit encontra todos os quadrados na imagem binária, e para cada quadrado encontrado, a imagem no seu interior é capturada e comparada com algumas imagens pré-cadastradas.

Por conseguinte, se existir alguma similaridade, o ARToolKit considera que encontrou um dos marcadores de referência. Então, o ARToolKit usa o tamanho conhecido do quadrado e a orientação do padrão encontrado para calcular a posição real da câmera em relação à posição real do marcador. Uma matriz 3×4 conterá as coordenadas reais da câmera em relação ao marcador. Esta matriz é usada para calcular a posição das coordenadas da câmera virtual. Se as coordenadas virtuais e reais da câmera são as mesmas, o objeto virtual pode ser desenhado precisamente sobre o marcador real[1].

Uma generalização da serie de eventos que permitem que esses processos aconteçam são apresentados na Fig. 4.10. Onde as funções *arVideoGetImage()*, *arDetectMarker()* e *arGetTransMat()* são funções próprias da biblioteca ARToolKit, essas três podem fazer parte de uma função geral *MainLoop()*. Tem-se também as funções gerais *Init()*, *Draw()* e *Cleanup()* que contém uma ou varias funções próprias de ARToolKit e de seus utilitários, por exemplo, as bibliotecas de comunicação com o usuário.

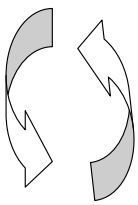
Inicialização	1. Inicializar a captura de vídeo e ler os arquivos de padrões de marcadores e parâmetros de câmera. *Init()*
Loop Principal 	2. Obter um quadro de entrada de vídeo. *arVideoGetImage()* 3. Detectar os marcadores e reconhecer padrões do fotograma da entrada de vídeo. *arDetectMarker()* 4. Calcular a transformação do sistema da câmara em relação aos padrões detectados. *arGetTransMat()* 5. Desenhe os objetos virtuais sobre os padrões detectados. *Draw()*
Encerramento	6. Feche a captura de vídeo e utilitários. *Cleanup()*

Figura 4.10: Princípios de desenvolvimento de aplicações com ARToolKit.

Um das características interessantes da AR para a engenharia é o uso de modelos 3D gerados com programas especializados. O original ARToolKit tem compatibilidade com arquivos WRL do linguagem VRML, isto graças às bibliotecas de funções

[1]Descrição obtida em: http://realidadeaumentada.com.br/home/index.php?option=com_content&task=view&id=6&Itemid=28

OpenVRML[1]. A maneira de exemplo do uso de ARToolKit com C++, na Fig 4.11 é apresentada em blocos a descrição da aplicação *simpleVRML* que está incorporada na distribuição original de ARToolKit e que até agora não estava documentada.

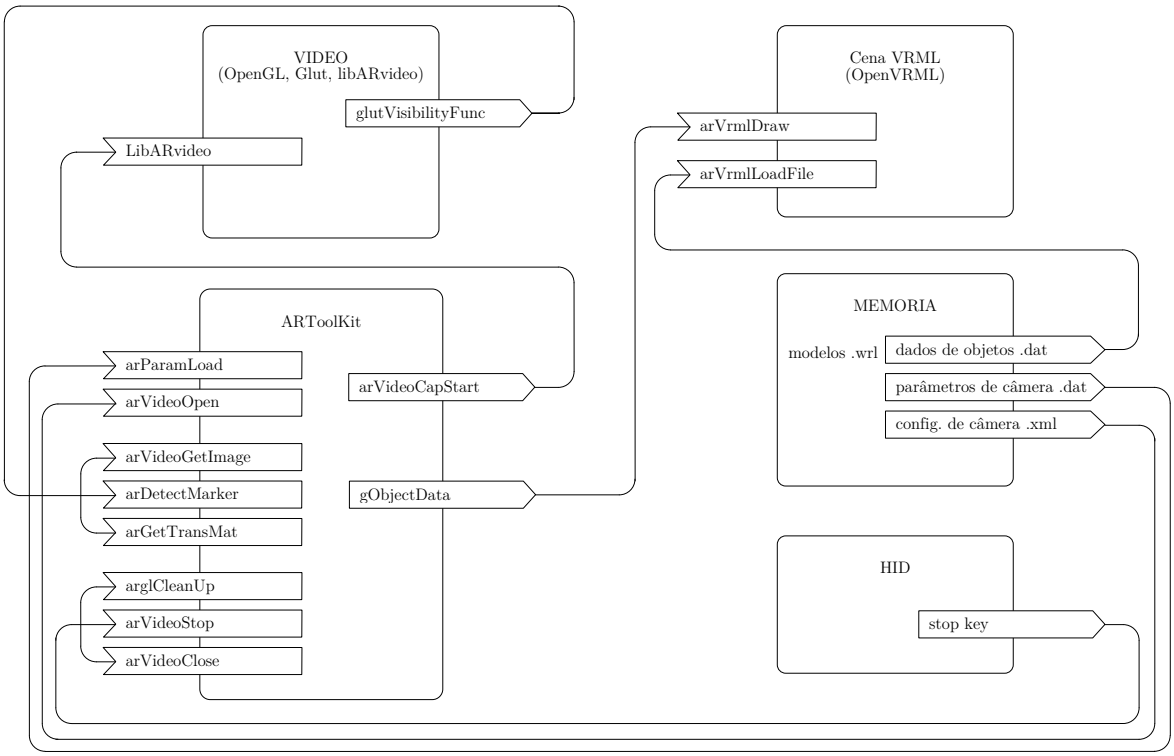


Figura 4.11: Esquema de blocos para implementação *simpleVRML* do ARToolKit.

A Fig. 4.11 mostra que no bloco MEMÓRIA o sistema ARToolKit precisa de três tipos de arquivos: o primeiro tipo (dados de objeto) é um ou vários arquivos *dat* que contem a informação do nome e posicionamento dos modelos 3D (arquivos *wrl*). O segundo tipo é um arquivo *dat* que armazena os parâmetros de calibração da câmera. E o terceiro é um arquivo *xml* que possui a informação para o *driver* controlador da câmera no sistema operacional. As informações do primeiro tipo de arquivo *dat* alimentam à biblioteca OpenVRML para pré-carregar os arquivos *wrl* e posteriormente serem usados por ARToolKit. O segundo tipo de arquivo *dat* é usado para inicializar o processo de ARToolKit nos cálculos de transformações de coordenadas. O terceiro tipo de arquivo é utilizado para inicializar a biblioteca LibARvideo, que é encarregada de apresentar as imagens de vídeo em uma tela. Esses processos são os correspondentes à função geral *init()*.

Conforme os passos dois, três e quatro na Fig. 4.10, os processos da função geral *MainLoop()*, no caso do exemplo *simpleVRML*, são regidos pela aquisição dos dados no ambiente de VIDEO (Fig. 4.11) e avaliadas pelas bibliotecas de ARToolKit usando as funções *arVideoGetImage()*, *arDetectMarker()* e *arGetTransMat()*.

[1]Mais informação em http://openvrml.org/

Usando a função *gObjectData()* a biblioteca OpenVRML redesenha a cena VRML, mas a renderização é feita no bloco VIDEO usando as bibliotecas Glut de OpenGL. A interação entre OpenVRML e OpenGL é feia de maneira fechada e o usuário não pode personalizar seu comportamento.

Com o propósito de encerrar o processo todo, um dispositivo de interface humana, HID (do inglês *Human Interface Device*), neste caso o teclado, produz um fechamento do sistema ARToolKit usando as funções *arglCleanUp()*, *arVideoStop()* e *arVideoClose()*. Tendo em vista que deve-se coordenar diversas bibliotecas, as funções apresentadas na Fig 4.11 são as representativas para uma compreensão rápida do funcionamento de ARToolKit com objetos virtuais armazenados em arquivos em formato *wrl*.

O sucesso deste exemplo, *simpleVRML*, é devido a facilidade para usar qualquer arquivo *wrl*, pois só é necessário modificar os arquivos *dat* para obter rápidos resultados e sem modificar o código C. Entretanto, não é possível manipular os modelos 3D sem entender as restrições apresentadas pela biblioteca OpenVRML. A Fig. 4.12 ilustra os resultados da AR usando o código do exemplo *simpleVRML*.

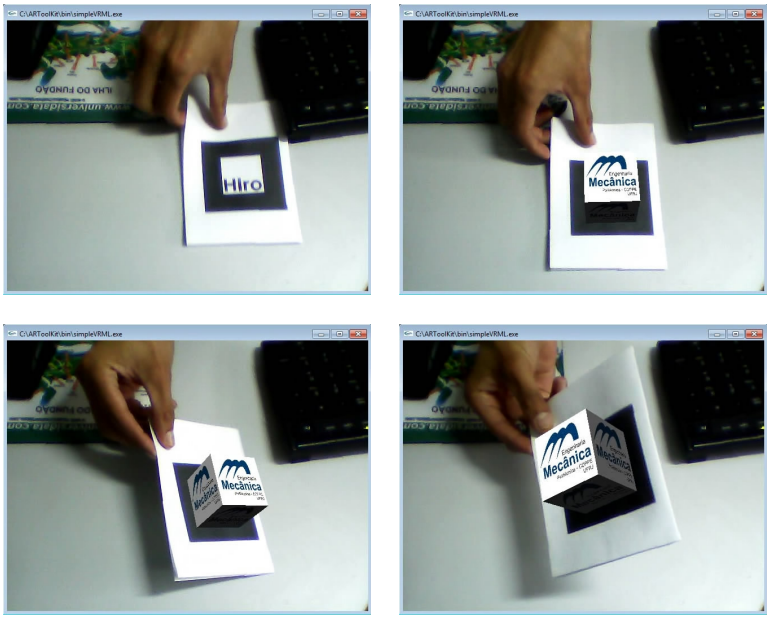


Figura 4.12: Resultados usando o exemplo *simpleVRML* do ARToolKit.

4.3 Desenvolvimento da Aplicação

Esta seção apresenta o uso dos modelos virtuais do manipulador TITAN-4 na AR usando as bibliotecas ARToolKit. Depois da analise feita anteriormente pode-se afirmar que o ambiente C++ é adequado para o desenvolvimento deste trabalho. As bibliotecas originais de ARToolKit permitem obter a posição e orientação (*pose*)

de um ou vários marcadores respeito ao sistema de coordenadas da câmera. Como o uso de modelos *wrl* é restrito pelas bibliotecas OpenWRL, são usados arquivos *obj*.

4.3.1 Sistemas de Coordenadas do ARToolKit

O sistema de coordenadas de ARToolKit é definido usando as matrizes de transformação. A Fig. 4.13 mostra que um vértice do marcador p_m descrito em seu sistema de coordenadas $\{m\}$ pode ser posicionado no plano do sistema de coordenadas ideal da tela $\{s\}$ e logo com outra transformação de matrizes ser descrito no sistema de coordenadas da câmera $\{c\}$. Mas as imagens produzidas pelas lentes reais apresentam na tela uma distorção, portanto, precisa-se de uma função de distorção para mapear a posição real obtida na tela, p_d, até sua correspondente representação ideal p_s.

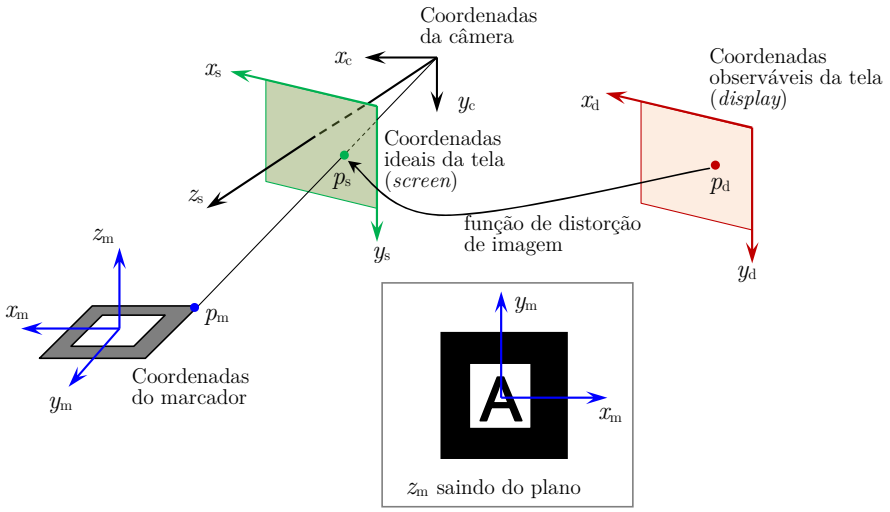


Figura 4.13: Sistema de coordenadas do ARToolKit.

A identificação da pose do marcador com respeito ao sistema de coordenadas é obtida usando os quatro vértices. Dois desses vértices geram uma reta que é paralela à outra reta feita pelos outros dois vértices. Quando é feita a transformação do sistema de coordenadas ideal da tela ao sistema da câmera, é possível gerar planos que contenham as projeções das retas. Usando o produto vetorial entre os vetores normais a esses planos, pode-se reconstruir a geometria da periferia do marcador. A descrição matemática e definição das matrizes de transformação desse processo, junto com a explicação do uso dos parâmetros da câmera, podem ser encontradas em [122].

Para o trabalho atual de AR foi usado o ambiente C++ com as funções originais da biblioteca ARToolKit. A vantagem dessa biblioteca é que tem incluidas no código C o processo de extração da *pose* do marcador com respeito à câmera explicado anteriormente e não requerendo ajuste algum. Como todo sistema sensor tem

suas desvantagens, neste caso a limitante aparece pelas condições de iluminação e à resolução das câmeras usadas. Quanto menor são as dimensões do marcador, menor é a distância de reconhecimento dos padrões dentro dos marcadores. Uma condição deficiente de luminosidade do ambiente prejudica o processo de reconhecimento de bordas produzindo falsos positivos e falsos negativos. Por conseguinte, a dedução do erro depende do equipamento real e em maior parte das condições de trabalho.

Pela observação, em diversas operações com ROV, onde precisa-se da manobrabilidade dos manipuladores, as condições de luminosidade são previsíveis e até controláveis e idealmente poderia ser utilizada a AR como sistema de sensoriamento para apoiar as tarefas dos pilotos de ROV.

4.3.2 Atingido o Alvo do Efetuador Final

Neste caso o objetivo com a AR é proporcionar ao operador (o piloto do ROV encarregado do manipulador) um esquema visual da configuração do manipulador para atingir um alvo com o efetuador final. Além disso, deve-se garantir que é possível cinematicamente obter uma solução adequada para a ação.

Uns dos principais problemas da teleoperação são as informações bidimensionais que as câmeras fornecem à visão estereoscópica do humano. A Fig. 4.14 mostra um exemplo típico deste tipo de problema, onde é requerida uma tarefa de encaixe de um objeto no furo vertical. A dificuldade aparece pela falta de noção de profundidade na visão monocular. Se a peça tem um reconhecimento visual, como um marcador fiducial utilizado em ARToolKit, e possível inferir a posição real do objeto em relação à câmera. Nesse sentido, a AR torna-se um sistema de sensoriamento aumentado do humano para cumprir uma tarefa de manipulação tridimensional.

Por médio da AR, se é determinada a posição da base de um manipulador com respeito à câmera, o efetuador final do robô pode se posicionar no ambiente real com

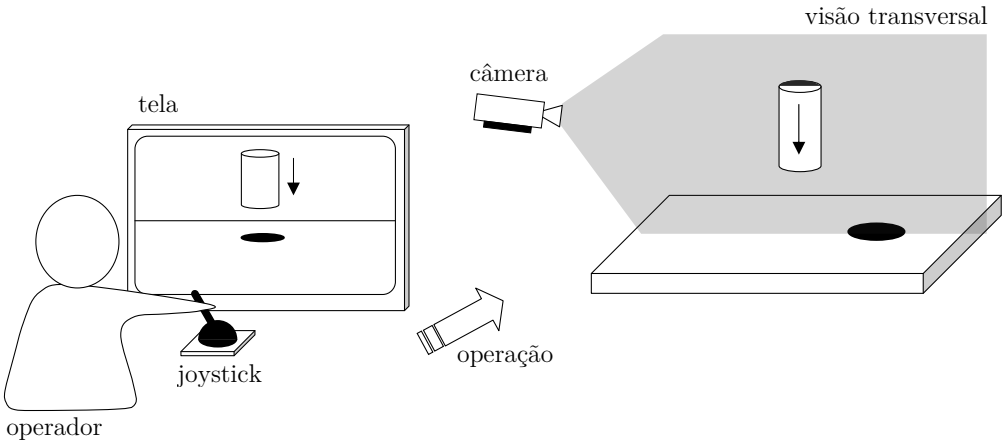


Figura 4.14: Perda da sensação de profundidade na visão monocular.

ajuda da cinemática direta. Agora com a circunstância anterior e com a cinemática inversa pode-se acertar ao alvo do efetuador final, se são conhecidas a posição e rotação desejadas com respeito à câmera.

A Fig. 4.15 ilustra os sistemas de coordenadas para o sistema de um manipulador e seu alvo definido por marcadores na AR. No processo da teleoperação, o braço robótico pode ser independente de um sistema global de coordenadas e ser referenciado a o ponto de visão da câmera $\{0\}$. A transformação de coordenadas T_{hb} entre o sistema $\{h\}$ do marcador de referência do manipulador e o sistema de coordenadas de sua base $\{b\}$ é conhecido pelo projetista. Os sistemas $\{h\}$ e $\{m\}$ são determinados por AR. Logo as matrizes de transformação homogênea T_{0h} e T_{0m} são calculáveis.

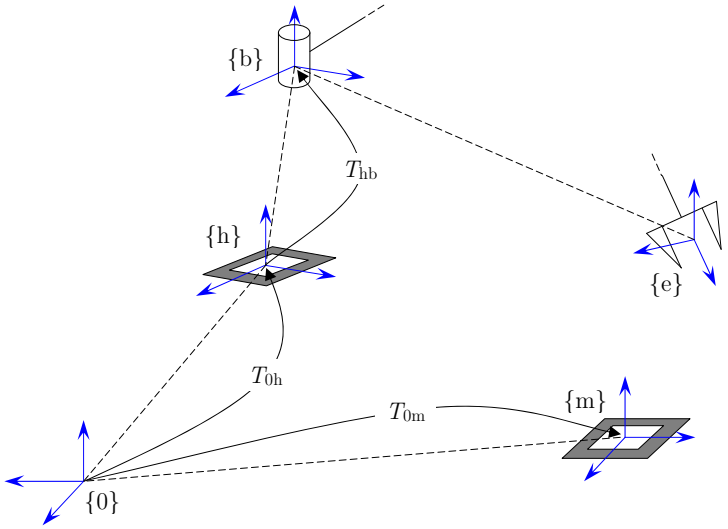


Figura 4.15: Sistema de coordenadas para manipulação na realidade aumentada.

A pose do efetuador final, $\{e\}$, em relação a sua base é determinada pela cinemática direta T_{be} (Eq. 2.13). Portanto, a transformação T_{0e} que expressa a pose do EF respeito à câmera é obtida por:

$$T_{0e} = T_{0h}T_{hb}T_{be} \tag{4.1}$$

Convém lembrar que para atingir o alvo, a solução da pose do EF é calculada com a cinemática inversa usando o sistema de coordenadas da base, por conseguinte, se o alvo do EF fica na origem do sistema de coordenadas do marcador $\{m\}$, a descrição desse sistema de coordenadas com respeito à base pode ser calculada assim:

$$T_{bm} = T_{b0}T_{0m} \tag{4.2}$$

$$T_{bm} = [T_{0b}]^{-1}T_{0m}$$

$$T_{bm} = [T_{0h}T_{hb}]^{-1}T_{0m} \tag{4.3}$$

Para agilidade computacional, a matriz inversa de T_{0b} pode ser obtida usando a propriedade:

$$T_{b0} = \left[\begin{array}{c:c} R_{0b}^T & -R_{0b}^T P_{0b} \\ \hdashline \mathbf{0}_{1\times 3} & 1 \end{array}\right] = [T_{0b}]^{-1} \tag{4.4}$$

Sendo:

$$T_{0b} = \begin{bmatrix} R_{0b} & P_{0b} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.5}$$

Os valores da matriz de rotação R_{0b} e das coordenadas de posição P_{0b} da origem do sistema $\{b\}$ com relação a $\{0\}$ são calculados de $T_{0b} = T_{0h}T_{hb}$ (Eq. 4.3).

Para o caso de conhecer a pose desejada do EF com relação ao sistema de coordenadas do marcador, isto é T_{me}, a obtenção da cinemática inversa se estabelece com T_{be}, que é obtida a partir de:

$$T_{be} = T_{b0}T_{0e}, \tag{4.6}$$

onde T_{b0} é calculada da Eq. 4.4. A pose desejada do EF em relação à câmera é determinada por:

$$T_{oe} = T_{0m}T_{me} \tag{4.7}$$

4.3.3 Arquitetura da AR

O esquema mostrado na Fig. 4.11 ilustra a visualização de modelos 3D com extensão *wrl*, mas sua estrutura não é apropriada para interagir com sistemas multicorpos. A aplicação AR desenvolvida neste trabalho é capaz de utilizar a cinemática inversa e direta de um manipulador virtual para interagir com uma cena real, sendo os modelos 3D importados de arquivos de extensão *obj*. Esses arquivos *obj* permitem interatuar de maneira clara com as bibliotecas Glut em OpenGL. Às vezes na AR os arquivos *wrl* são preferidos por suas animações embarcadas; no estudo de caso deste trabalho, essa condição não é necessária e o controle dos sólidos virtuais deve-se realizar diretamente desde o aplicativo.

A Fig. 4.16 apresenta de forma modular o desenvolvimento da AR em C++, baseado em ARToolKit, para que o EF de um manipulador virtual atingia um alvo da cena real representado por um marcador. Desde a MEMÓRIA os arquivos *obj* são pré-recarregados no módulo de VIDEO pela Biblioteca OBJ, que traduz as geometrias ao ambiente de renderização OpenGL. Também são extraídas da MEMÓRIA tanto os parâmetros intrínsecos da câmera como a configuração do driver do sistema operativo para o uso da câmera (arquivos *dat* e *xml* respectivamente). Essas informações são utilizadas por ARToolKit (correspondente a função geral *Init()* da seção anterior) para inicializar o ambiente de VIDEO com a biblioteca libARvideo.

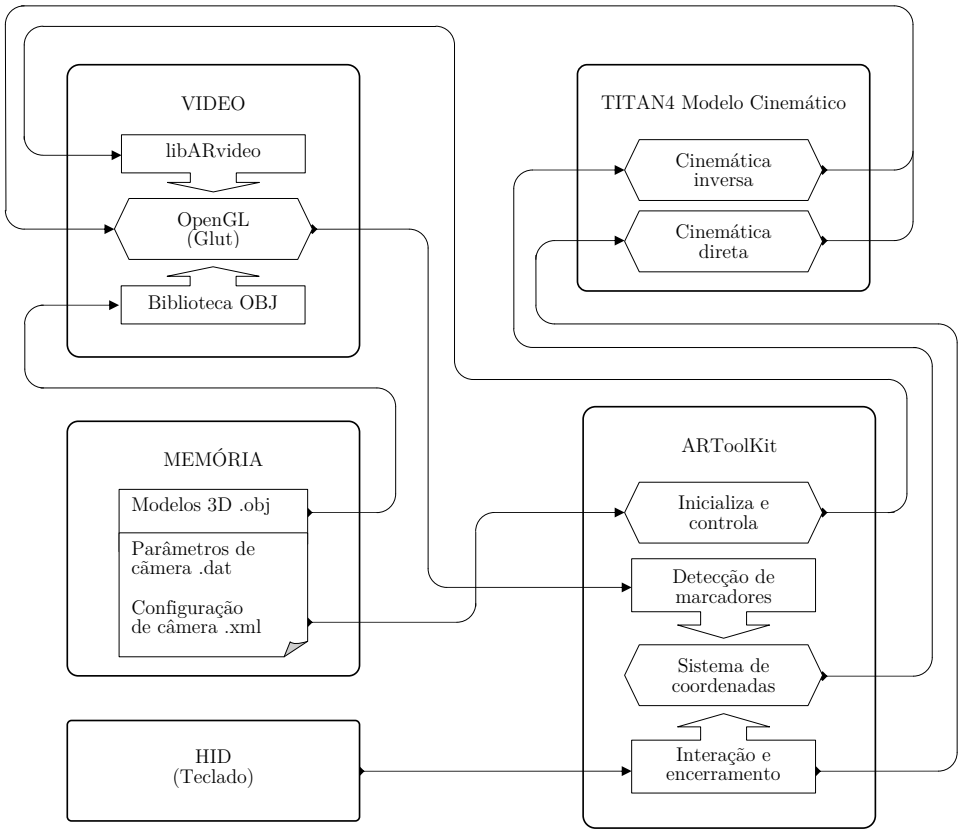


Figura 4.16: Arquitetura geral modular da implementação da AR no caso de manipulador atingindo o alvo.

Quando são detectados os marcadores desde o bloco OpenGL, o modulo ARToolKit tem como saída os sistemas de coordenadas dos marcadores com respeito à câmera. Esse processo é baseado nas funções *arVideoGetImagen()*, *arDetectMarker()*, e *arGetTransMat()* estudadas anteriormente, e correspondem a função geral *MainLoop()* (ver Fig. 4.10).

Desde o bloco de Sistema de Coordenadas do modulo ARToolKit, as informações das coordenadas são levadas ao bloco da Cinemática Inversa, onde esse bloco realiza dos tarefas: a primeira, calcula a pose do alvo respeito à base como é indicado na seção 4.3.2; e segundo, o calculo da cinemática inversa, propriamente dita, usando os conceitos da seção 3.2. Convém lembrar que o Caso 3 de multiplicidade deve-se resolver para a solução fechada.

A apresentação ao usuário do movimento do manipulador atingindo o alvo é resultado da interpretação da cinemática inversa no modulo VIDEO usando o bloco OpenGL. Essa ações são as correspondentes à função *Draw()* da seção anterior. Finalmente, o teclado como HID provê ao usuário da interação com o modelo virtual do manipulador através da cinemática direta, até a representação no modulo de VIDEO. Para a saída do aplicativo, o HID ativa as mesmas funções de encerramento de ARToolKit utilizadas no exemplo da Fig. 4.11.

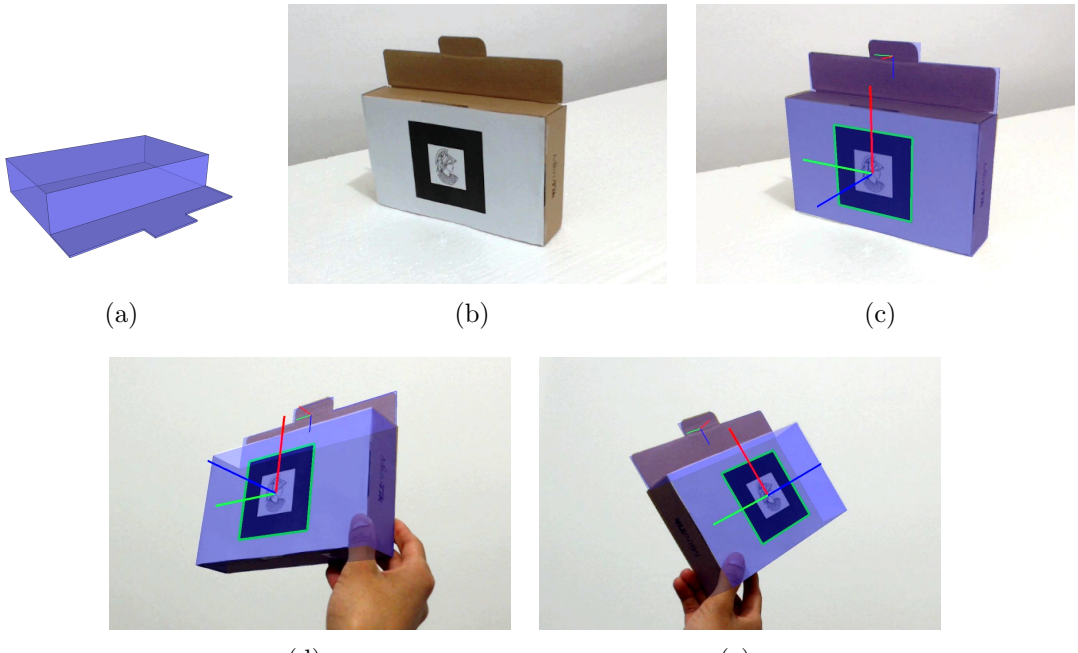
4.4 Manipulação Aumentada

Nesta seção mostra os resultados visuais usando a ferramenta desenvolvida em AR e introduz novos conceitos para o caso da robótica. Se bem que a aplicação de AR apresentada na Fig. 4.16 foi desenvolvida para o robô TITAN-4, essa arquitetura pode ser usada para qualquer tipo de manipulador serial.

4.4.1 Virtualização

Pode-se definir como Virtualização à ação de reconstrução virtual de objetos reais. No caso da AR, com o uso de marcadores é possível determinar a pose de um objeto real em relação com a câmera. Além, esse objeto deve ser conhecido geometricamente. Por exemplo, a Fig. 4.17(a) é a representação de um objeto real da Fig. 4.17(b). Com AR o solido virtual é sobreposto no video com ajuda do marcador central no objeto real, Fig. 4.17(c). Conseqüentemente, é possível determinar o estado do objeto real usando marcadores sendo conhecida a geometria do objeto.

O fato de movimentar o conjunto real e virtual pode ser considerado como a Manipulação Aumentada, tal como é mostrado na Fig. 4.17(d) e Fig. 4.17(e), onde objeto real é virtualizado o tempo todo. Observe-se que o processo de virtualização não requer que o objeto virtual esteja visível ao usuário. A Fig. 4.17 apresenta a virtualização com o objeto virtual translucido para facilitar a compressão.



(a) (b) (c)

(d) (e)

Figura 4.17: Processo de virtualização e manipulação aumentada. (a)-(c) Ações para virtualização. (d),(e) Manipulação aumentada.

O processo de virtualização da realidade é ainda um tópico em desenvolvimento. A virtualização de objetos flexíveis não é contemplada neste trabalho. Não obstante, com o advento de novas tecnologias no processamento de imagem, espera-se no futuro acrescentar os benefícios da manipulação aumentada.

4.4.2 Visualização Exógena

A Fig. 4.15 é o caso tipico deste tipo de visualização, onde o manipulador e posicionado dinamicamente em relação à câmera. A Fig. 4.18 apresenta o caso real usando o modelo 3D do manipulador TITAN-4 (escala 1:1). O alvo do efetuador final é definido como um sistema de coordenadas que fica na parte superior do solido virtualizado. Esse sistema de coordenadas é fixo em relação com o sistema de coordenadas do marcador e varia somente quando há manipulação aumentada.

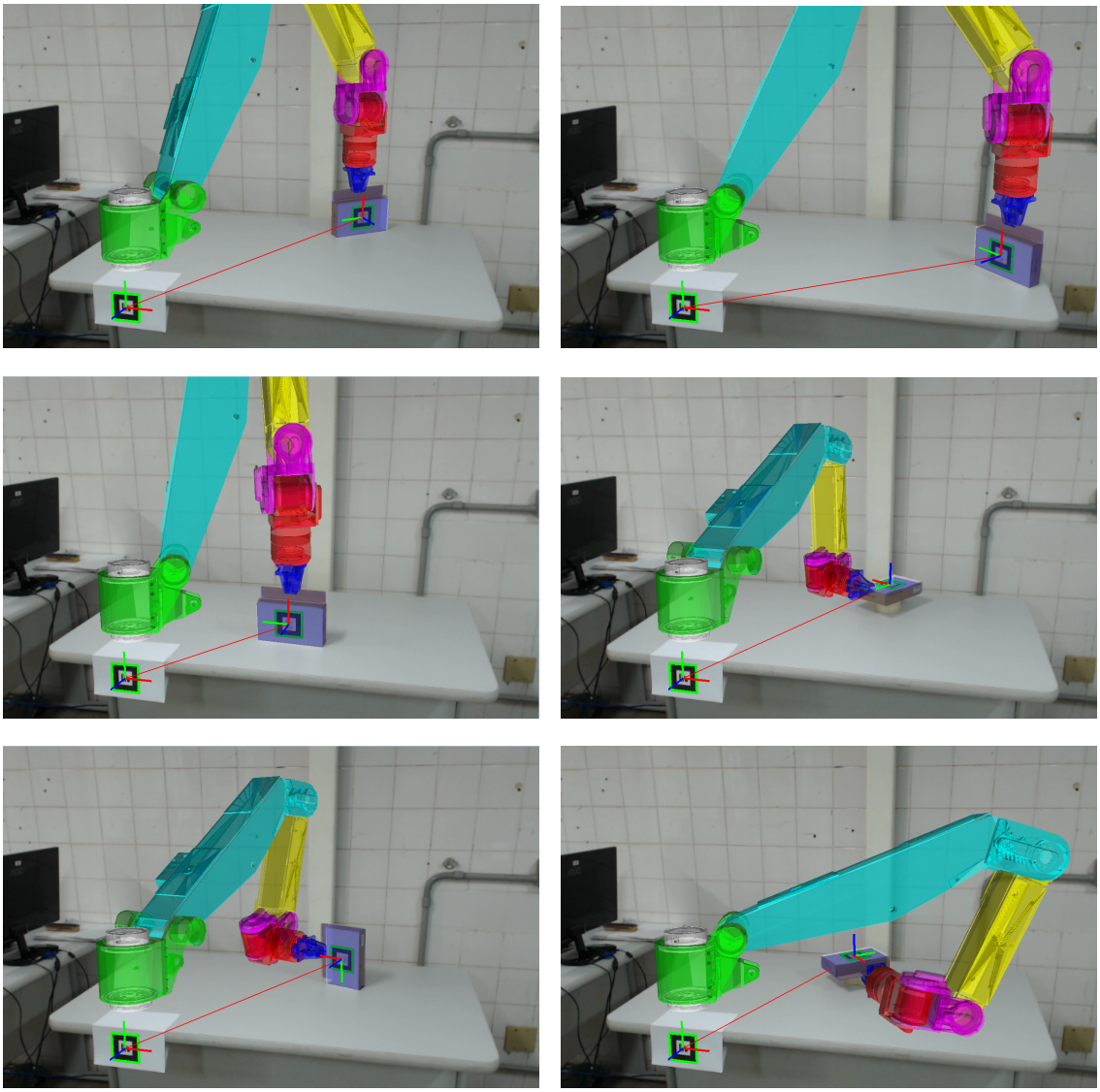


Figura 4.18: Capturas da visualização exógena em diversas situações.

Em virtude aos resultados visuais obtidos pela implementação AR, o TITAN-4 na Fig. 4.18 consegue atingir o alvo, entretanto, apresenta-se as seguintes características:

- Com os marcadores fixos sem movimento, o manipulador apresenta oscilações, as quais podem ser agrupadas em dos grupos de fatores: o primeiro são as características próprias da AR como iluminação, dimensões dos marcadores e sua orientação; o segundo grupo é estabelecido pelas condições da cinemática de cadena aberta do manipulador, quanto maior os comprimentos dos elos, também são mais evidentes as oscilações.

- O uso da cinemática inversa fechada disponibiliza até quatro soluções viáveis (sem contar a multiplicidade das soluções), mas, o seguimento do alvo na manipulação aumentada não é fluente no caso de intercambio de soluções (por exemplo na posição do cotovelo). Esta característica é um típico problema de controle robótico.

- A velocidade com que os objetos virtuais são sincronizados com os objetos reais dependem de dois condições: a primeira são os múltiplos processos de cálculo descritos Fig. 4.16, como as cinemáticas e renderizações. A segunda condição é a taxa de amostragem de captura de dados da biblioteca ARToolKit. Não tem-se evidencia de retardo apreciável, isto graças ao desenvolvimento feito em C++. Esta característica de sincronização é determinada pela qualidade da câmera e do poder de processamento computacional na AR.

Pode-se concluir que para minimizar os efeitos nocivos dessas características, é requerido um ajuste do tamanho dos marcadores, o controle das condições de iluminação e um sistema robusto de solução da cinemática inversa.

O desenvolvimento até aqui feito mostra a viabilidade do uso da AR no caso de visualização exógena. A característica mais predominante neste caso é a oscilação dos sólidos virtuais, a qual pode ser compensada com o uso síncrono de duas câmeras. Outra opção é o uso de um diferente sistema AR, onde seja possível o reconhecimento de padrões visuais sem marcadores.

4.4.3 Visualização Endógena

No caso anterior as oscilações e erros são causados em maior parte pelas condições externas à câmera, como o reconhecimento dos marcadores. No caso de visualização endógena é diferente, pois a *pose* da base do manipulador é fixada o tempo todo em relação ao sistema de coordenadas da câmera sem utilizar marcadores. Quer dizer, que a matriz T_{0b} é conhecida (Fig. 4.15).

Na visualização endógena, na ausência do marcador da base, o manipulador virtual não possui oscilação alguma em relação à câmera. Para o caso de manipulação aumentada, as oscilações são apreciavelmente menores que na visualização exógena. A Fig. 4.19 mostra os resultados visuais da aplicação AR no modo de visualização endógena, tal como é prevista em um ambiente real de operação no ROV.

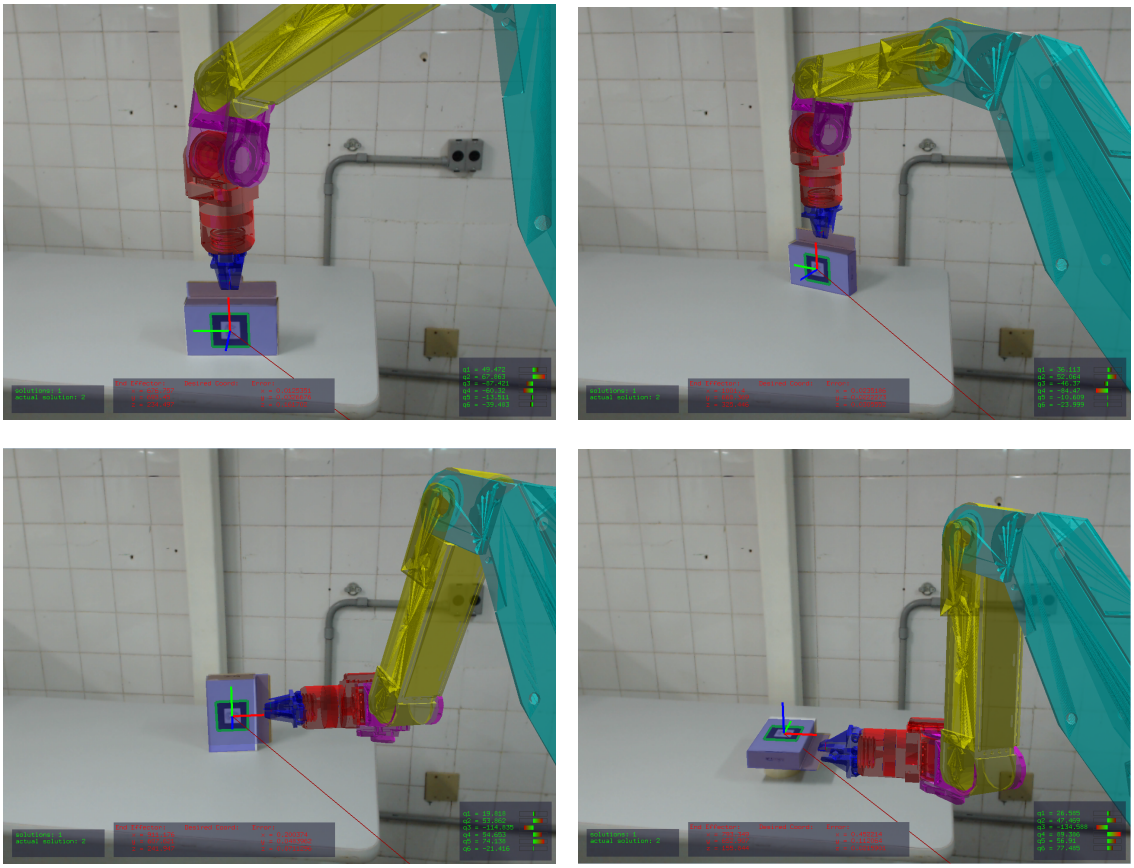


Figura 4.19: Capturas da visualização endógena em diversas situações.

Este tipo de visualização também apresenta as mesmas características descritas no caso exógeno, mas com algumas particularidades:

- As oscilações são menores e proporcionadas em maior medida pelo reconhecimento do solido virtualizado. Quando a câmera treme, o manipulador virtual treme com ela.
- As soluções da cinemática inversa seguem dependendo da robustez do algoritmo, mas apresenta-se frustração do usuário, pois o manipulador não atinge o alvo em algumas poses do objeto real.
- Não há retardo apreciável na visualização na manipulação aumentada.

A insatisfação do usuário quando o manipulador não consegue atingir o alvo de maneira automática indica a dificuldade da manipulação em ROV reais. Tendo em

vista os testes com AR, a análise sugere três causas da dificuldade da manipulação endógena:

1. ***Restrições Cinemáticas***. Abarca a sinergia de dois aspectos: os limites das juntas e a morfologia do manipulador. Por exemplo na Fig. 4.20 é apresentada as condições limites da segunda e terceira junta. São indicadas dos configurações extremas, quando $\theta_2 = 78$, $\theta_3 = -164$ graus e $\theta_2 = -42$, $\theta_3 = 106$ graus. Lembrando que o punho do TITAN-4 fica no ponto P e não no ponto C como no caso dos manipuladores 6-GDL com punho esférico. Os dois últimos GDL são dependentes da pose do elo CP (e não do elo BC como no caso de robôs com punho esférico). Por tanto, para as configurações na borda das circunferências generatrizes do espaço de trabalho, o robô perde um GDL. Essas condições podem ser interpretada como singularidade mecânicas é não são obtidas atravéz da análise do jacobiano.

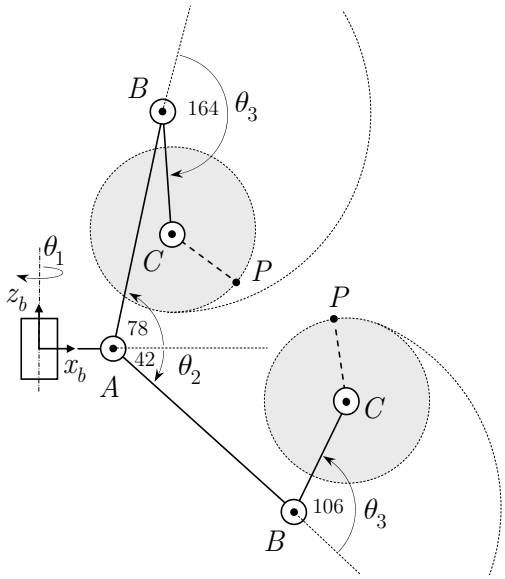
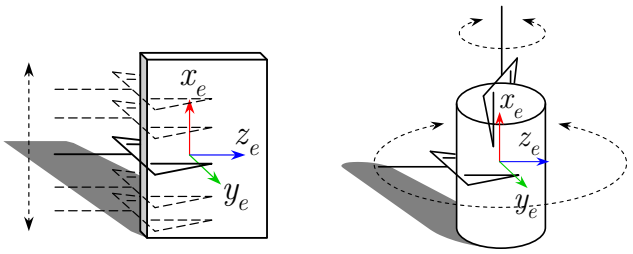


Figura 4.20: Representação de perdida de posto por limites nas juntas no manipulador TITAN-4.

2. ***Rigorosidade do alvo***. Para o manipulador a ação de agarrar do EF depende de uma única posição e orientação do sistema de coordenadas do alvo. Para o humano esta ação é uma tarefa natural e intuitiva, pois a aprendizagem fornece diversas poses da mão. Na Fig 4.21 indica as múltiplas soluções que o EF poderia ter com um sistema flexível de agarramento. Observe-se que o conjunto de soluções depende da geometria do objeto virtualizado, por conseguinte, para facilitar a teleoperação o sistema inteligente deve sugerir um conjunto de poses do EF e determinar a melhor resposta entre todas as cinemáticas inversas calculadas para o grupo de poses. Depende da estratégia usada na seleção de soluções, o tempo de computação pode ser elevado.



(a) (b)

Figura 4.21: Exemplos de sistema flexível de agarramento para o efetuador final. (a) Conjunto de possíveis posições ao longo do ideal eixo x_e. (b) Dois conjuntos de possíveis orientações do efetuador final ao redor do ideal eixo x_e.

3. ***Espaço Visível***. Já na Fig. 3.8 é evidenciada a restrição do espaço do espaço de trabalho do manipulador devido às linhas de visão das câmeras. Esta condição tem duas implicações: a primeira envolve o sistema de feedback visual na teleoperação, pois o humano precisa ver o robô para fixar em sua mente a configuração atual do manipulador e inferir que é possível atingir o alvo. Essa característica acontece em diversas ocasiões quando não são visíveis a primeira e segunda junta, além disso, o mecanismo de propriocepção humana não está acostumado com os comprimentos e restrições cinemáticas do robô. A segunda condição é conseqüência da visão monocular, pois quando a percepção humana indica que o EF pode atingir o alvo é possível que em realidade as coordenadas do alvo fiquem nas bordas do espaço de trabalho do EF onde o manipulador perde um GDL. Essa condição é acrescentada em virtude que em geral o manipulador ficar na direita do campo de vissão nos ROVs.

Os estudos focados no avanço da manipulação em relação com a rigorosidade do alvo são tópico de constate pesquisa na área da cirurgia minimamente invasiva [139–141]. No caso direto do estudo de agarramento para manipuladores tem-se alguns trabalhos como em [142–144]. Já no caso específico de manipuladores submarinos em [29] apresenta-se uma primeira aproximação de representar as características de destreza no espaço de trabalho. Além dos estudos de caracterização do espaço, outros conceitos interessantes de agarramento flexível com novos tipos de efetuador final são descritos em [145–147].

4.5 Precisão e Exatidão

É importante distinguir entre precisão e exatidão. A precisão de um resultado é uma medida da reprodutibilidade de uma observação, ou seja, quão bem é possível reproduzir um resultado, independentemente do perto que esse valor fique do valor

"verdadeiro". Ao erro associado é denominado como "incerteza" do resultado. A exatidão é uma medida do certo que é uma observação, ou seja, quão perto está do valor "verdadeiro".

Esta seção indica como realizar uma análise da precisão e da exatidão na AR usando a aplicação computacional desenvolvida. Três casos são estudados, os dois primeiros envolvem a determinação da precisão e o terceiro a estimativa da exatidão. Em todos os casos são utilizados marcadores fiduciais de comprimento padrão de $80mm$, e o reconhecimento das imagens cadastradas é feito depois da calibração da câmera, arquivos são armazenados para seu uso posterior no aplicativo AR.

A câmera usada é de referência Logitech C920 que permite captura de vídeo HD até 1080p a 30 quadros por segundo, com campo de 78 graus e lente de focagem automática de vidro projetada pela Carl Zeiss. A maneira de informação, o processo de calibração da câmera mostra que sua matriz característica[1] KK está dada pela expressão 4.8. Esta matriz, junto com outros parâmetros de distorção, foi obtida para uma resolução de tela de 640×480 pixels, utilizada nos testes com o aplicativo AR. As condições de iluminação foram artificiais e os comprimentos medidos com instrumentos comerciais.

$$KK = \begin{bmatrix} 877.6915 & 0.2272 & 320.6451 \\ 0 & 869.8953 & 242.8119 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Lembre-se que a estimativa da precisão aumenta com o uso de instrumentos calibrados até os padrões primários de acordo com o conceito de rastreabilidade na metrologia, portanto, as variações das incertezas dependem também da propagação dos erros de calibração dos instrumentos de medição [148]. Como no mercado tem-se muitos tipos de câmera e inúmeras condições ambientais, nesta seção foi apresentada uma estimativa dos erros em casos pontuais, em condições normais, para observar o comportamento da AR e mostrar de maneira geral a viabilidade do uso desta tecnologia no caso da robótica em manobras de teleoperação.

4.5.1 Avaliando a Precisão

Na maioria dos casos, para N observações independentes que foram obtidas sob as mesmas condições de medição, uma estimativa disponível da precisão do valor esperado com variação aleatória é o desvio padrão experimental da média $s_{\bar{x}}$ definido como:

$$s_{\bar{x}} = \frac{s_x}{\sqrt{N}} \tag{4.9}$$

[1]Descrição matemática dos parâmetros em http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html

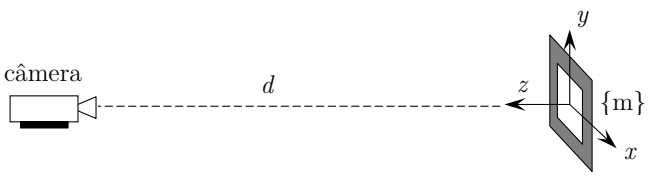


Figura 4.22: Representação do primeiro teste para avaliação da precisão.

Sendo s_x o desvio padrão amostral, para uma média aritmética $\bar{x}$, também chamada de média das N observações. Logo, o valor da grandeza do valor esperado x' e dado por:

$$x' = \bar{x} \pm s_{\bar{x}} \tag{4.10}$$

O primeiro teste é feito como ilustra a Fig. 4.22, onde o marcador fica no frente da câmera numa distância d. Após de 100 capturas reconhecidas do marcador, é possível obter as discrepâncias da distância inferida pela AR. A Tabela 4.1 apresenta os dados obtidos para diferentes distâncias.

Tabela 4.1: Resultados para o primeiro teste para avaliação da precisão, com $N = 100$ observações por série.

Série 1: $d \sim 500$ [mm]			
Coord.	x	y	z
Máx	1,0442	0,7767	505,9640
Mín	1,0038	0,7487	505,6400
Média	1,0280	0,7641	505,7898
Desv.	0,0071	0,0065	0,0671

Série 2: $d \sim 1000$ [mm]			
Coord.	x	y	z
Máx	1,1374	3,8323	997,2500
Mín	1,0531	3,7894	995,7280
Média	1,0885	3,8133	996,4844
Desv.	0,0160	0,0113	0,3676

Série 3: $d \sim 1300$ [mm]			
Coord.	x	y	z
Máx	3,3963	14,4725	1298,9100
Mín	3,3017	14,3246	1295,5000
Média	3,3554	14,3850	1296,7787
Desv.	0,0201	0,0252	0,5402

Série 4: $d \sim 1500$ [mm]			
Coord.	x	y	z
Máx	9,0354	18,9694	1539,0600
Mín	8,8338	18,8956	1534,8000
Média	8,9229	18,9287	1536,6946
Desv.	0,0377	0,0133	0,7901

Série 5: $d \sim 1700$ [mm]			
Coord.	x	y	z
Máx	114,5460	26,8476	1733,2800
Mín	113,9340	26,6144	1724,1400
Média	114,1568	26,7594	1727,4685
Desv.	0,1240	0,0486	1,8604

Série 6: $d \sim 2000$ [mm]			
Coord.	x	y	z
Máx	58,6802	4,0331	2041,1300
Mín	58,3940	3,5828	2029,8200
Média	58,5330	3,8215	2035,0656
Desv.	0,0497	0,0775	2,1142

Com os dados expostos na Tabela 4.1, o fator comum de maior dispersão entre as séries é a coordenada z do centro do marcador em relação com a câmera, pois em

Tabela 4.2: Resumo dos dados de dispersão no caso do primeiro teste na coordenada z. A letra Q representa o respectivo quartil.

	Série					
	1	2	3	4	5	6
Máx	505,9640	997,2500	1298,9100	1539,0600	1733,2800	2041,1300
Mín	505,6400	995,7280	1295,5000	1534,8000	1724,1400	2029,8200
Média	505,7898	996,4844	1296,7787	1536,6946	1727,4685	2035,0656
Desv.	0,0671	0,3676	0,5402	0,7901	1,8604	2,1142
1o Q	505,7473	996,1815	1296,5000	1535,9500	1725,9425	2033,8500
Mediana	505,7885	996,4480	1296,6100	1536,9250	1727,2550	2035,1500
3o Q	505,8320	996,7228	1297,0500	1536,9825	1728,5025	2035,9125

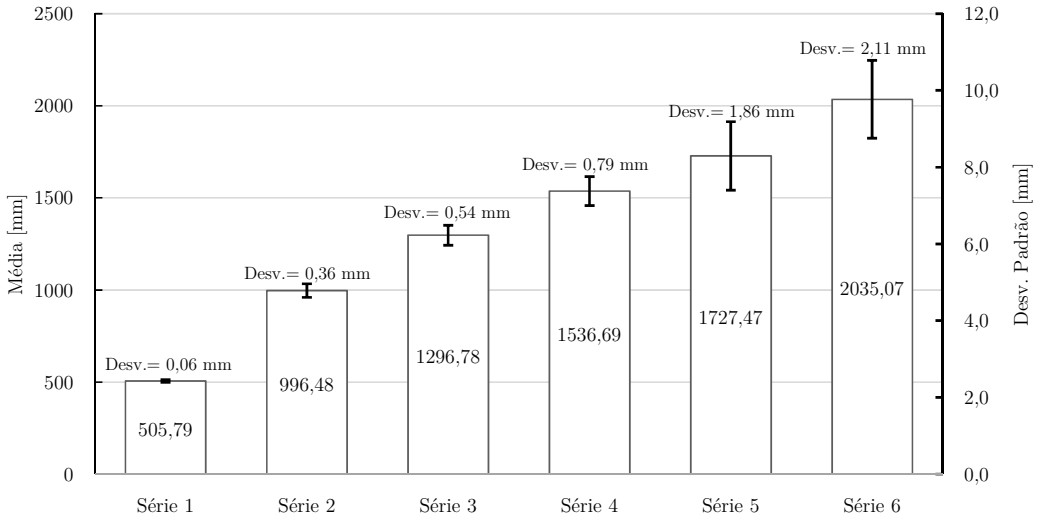


Figura 4.23: Representação da dispersão do primeiro teste na coordenada z por serie cada. Valores centrais da média nas barras retangulares y ordem de grandeza das barras lineares de dispersão em milímetros.

todos os casos o desvio padrão e maior que nas coordenadas x e y.

A Tabela 4.2 apresenta um resumo das séries para a coordenada z; além disso, são apresentados os dados da mediana e dos quartis. Nota-se que até a distancia de 1500mm o desvio padrão não ultrapassa $1mm$. Correspondente, a Fig. 4.23 apresenta um resumo da tendência da dispersão dos dados por série cada. O eixo vertical esquerdo indica o valor da média obtida por serie (barras retangulares); o eixo direito, os valores de desvio padrão ilustrando a ordem de grandeza da dispersão (barras lineares).

A mediana é uma medida de localização do centro da distribuição dos dados, ou seja, divide a distribuição de valores em duas partes iguais. É interessante o uso da mediana, pois ela não é tão afetada pelos valores extremos como a média aritmética. Os quartis dividem ou separam uma distribuição de freqüência em quatro partes iguais. Note-se na Tabela 4.2 que em todos os casos o valor da mediana está perto

dos valores da média, e as faixas entre quartis também é pequeno em relação aos comprimentos d por série. Conseqüentemente, a média pode ser utilizada para a descrição da precisão na faixa de valores amostrais.

Tendo em vista os resultados anteriores, a precisão nos dois casos extremos (série 1 e 6), usando a expressão 4.10, será: $x'_1 = 505,7898 \pm 0,0067$ e $x'_6 = 2035,0656 \pm 0,2114$; mesmo que, as fixas máximas em relação com a média são $\pm 0,1742mm$ e $\pm 6,0644mm$ para a série 1 e 6, respectivamente, na coordenada z (Tabela 4.1). As faixas máximas são calculadas partindo de $\max\{(\text{Máx} - \text{Média}), (\text{Média} - \text{Mín})\}$.

Um segundo teste que mostra como varia a dispersão dos dados fora do centro do marcador é proposto na Fig. 4.24, onde agora o sistema de coordenas do efetuador final é o ponto de comparação, isto é, a estimativa da distância d entre a câmera e o centro do sistema de coordenadas $\{e\}$.

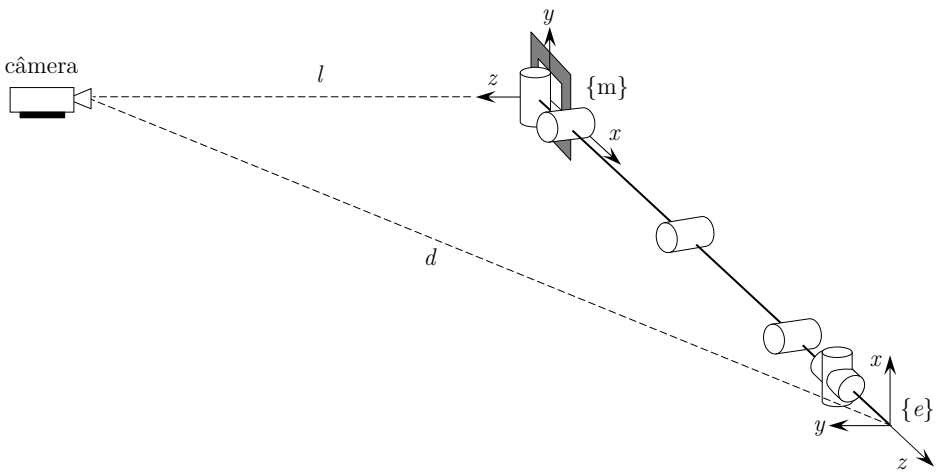


Figura 4.24: Representação do segundo teste para avaliação da precisão.

De maneira similar ao primeiro teste, a Tabela 4.3 apresenta um resumo para quatro séries, correspondentes às distâncias aproximadas ao centro do marcador $l \sim \{\ 500,\ \ 1000,\ \ 1500,\ \ 2000\ \}$ em milímetros para o análise do segundo teste, sendo $N = 100$ observações por série.

Para esse segundo teste é obvio que quanto mais afastado o efetuador final do centro de coordenadas do marcador, maior será a dispersão dos dados obtidos pela AR. O caso mais acentuado acontece com o braço estendido na série 4. A Fig. 4.25 mostra como os dados de desvio padrão das série 3 e 4 aumentam sensivelmente em relação ao primeiro teste na Fig. 4.23. O mesmo acontece analisando o comportamento dos quartis da série 4. Já a discrepância tem ordem de mais de $2mm$. Pode-se inferir que para distâncias $l > 1000mm$ o uso de um único marcador de comprimento padrão $80mm$ é insuficiente para conferir uma precisão menor de $2mm$. Sendo o valor esperado para as séries 1 e 4: $x'_1 = 1962.9327 \pm 0,0526$ e $x'_4 = 2784.6693 \pm 0,5518$.

Tabela 4.3: Resumo dos dados da distância d percebida no caso do segundo teste com $N = 100$ para cada série. Os valores estão em milímetros.

	Série			
	1	2	3	4
Máx	1963,3162	2134,1887	2407,2225	2791,8982
Mín	1962,3938	2131,7919	2398,0351	2777,4443
Média	1962,9328	2132,7625	2402,7820	2784,6694
Desv.	0,5261	1,2301	5,3397	5,5185
1o Q	1962,7860	2132,3903	2400,3763	2782,7907
Mediana	1962,9616	2132,7536	2403,4463	2784,4271
3o Q	1963,1051	2133,0369	2404,6499	2787,0907

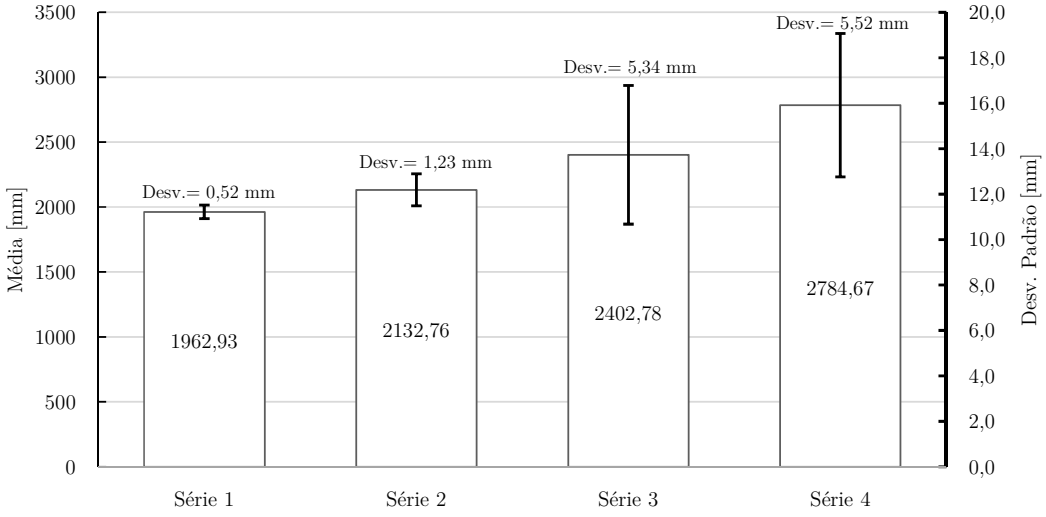


Figura 4.25: Representação da dispersão do segundo teste. Valores centrais da média nas barras retangulares y ordem de grandeza das barras lineares de dispersão em milímetros.

Levando-se em consideração os resultados dos dois testes, a precisão da visualização exógena depende notavelmente do comprimento do marcador usado. Uma melhora para esse caso é a utilização do reconhecimento de múltiplos marcadores para o posicionamento da base. No caso de uma tradicional servovisão, um ou vários marcadores podem ser usados no efetuador final e no pulso para garantir as coordenadas em relação à câmera. Por outro lado, a visualização endógena não é tão sensível aos erros de precisão desde que o sistema de coordenadas do alvo não fique afastado do centro de coordenadas do marcador. Em virtude do segundo teste, conclui-se que o alvo posicionado a distâncias menores de um metro do centro de marcador apresenta níveis de dispersão aceitáveis. Os resultados feitos aqui não são definitivos, mas mostram a viabilidade do uso da AR na robótica como procedimento para obter medições da realidade usando câmeras.

4.5.2 Avaliando a Exatidão

A exatidão do resultado pode ser quantificada por cálculo do erro percentual. O erro percentual só pode ser encontrado se o valor verdadeiro é conhecido usando a expressão:

$$erro[\%] = \frac{x' - x}{x} \times 100\% \tag{4.11}$$

Sendo x' o valor estimado e x o valor verdadeiro. percebe-se que o sinal positivo ou negativo só indica uma tendência e pode ser omitido usando o valor absoluto.

A Fig. 4.26 indica o teste usado para estimar a exatidão do sistema AR desenvolvido. Visualiza-se que o comprimento x é estabelecido de maneira real, e seu x' é estimado usando substração das coordenadas dos centros dos dois marcadores $\{m_1\}$ e $\{m_2\}$ identificados pela aplicação AR.

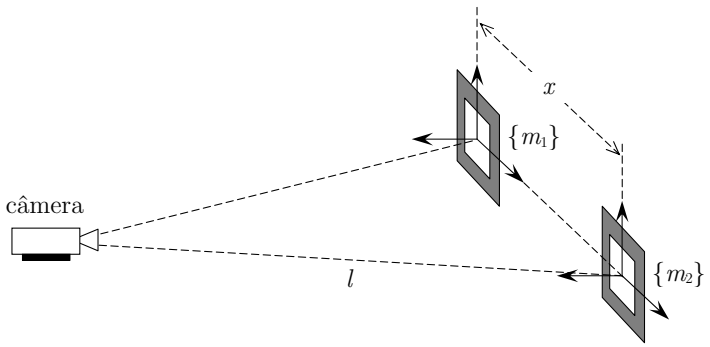


Figura 4.26: Representação do teste para avaliação da exatidão.

Os resultados de este teste são obtidos a partir de quatro seqüências para o valor de $x = \{125, 300, 400, 500\}mm$, em cada seqüência e feito quatro séries com usando os valores da distância $l = \{600, 1000, 1500, 2000\}mm$, de cada série é possível obter uma estimativa de x' realizando 100 observações, de maneira similar como foi realizada na avaliação da precisão.

A Tabela 4.4 apresenta um resumo da seqüência $x = 300mm$, onde em cada série a média representa o valor estimado x'. Já a Fig. 4.27 mostra o comportamento das quatro séries da mesma seqüência. Observe-se que cada série está deslocada em relação à linha $x = 300mm$ conforme ao valor do erro porcentual da Tabela 4.4.

Tabela 4.4: Resumo de dados da seqüência $x = 300mm$ para o teste de exatidão.

	$l \sim 600$	$l \sim 1000$	$l \sim 1500$	$l \sim 2000$
Máx	299.2496	300.5988	301.1004	303.3787
Mín	299.0131	300.1578	300.5339	302.5073
Média	299.1255	300.3340	300.8539	302.9160
Dev.	0.0432	0.0814	0.1088	0.1712
erro [%]	-0.2915	0.1113	0.2846	0.9720

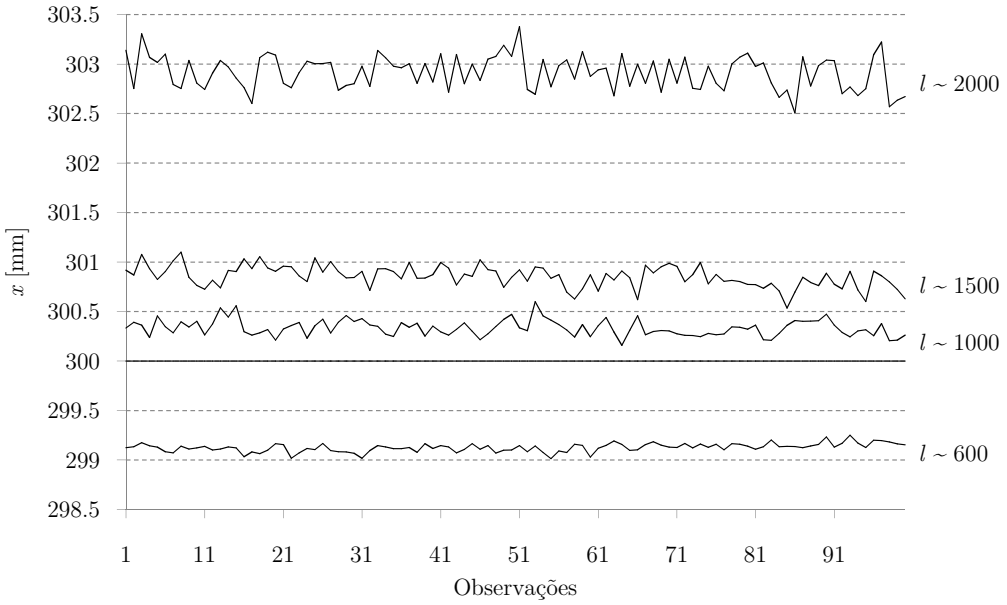


Figura 4.27: Representação das deslocações das séries de dados obtidos para a seqüência $x = 300mm$ do teste de exatidão.

Na série $l \sim 600$ acontece um erro negativo, que pode ser atribuído à proximidade dos marcadores à câmara, o que faz quase eles desaparecer do campo visual, acrescentando os erros pela distorção da lente.

Por outro lado, uma maneira de representar a tendência dos erros percentuais para todas as seqüências do teste é mostrada na Fig. 4.28, onde cada linha de tendência é uma seqüência e os pontos são o erro estimado por cada série. O eixo horizontal indica a distância estimada entre a câmera e os marcadores.

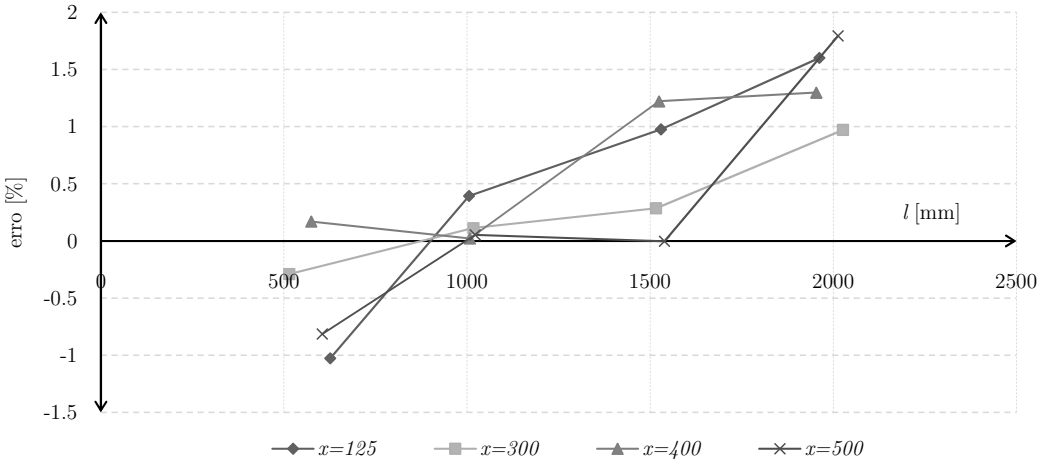


Figura 4.28: Representação da tendência dos erros porcentuais para todas as seqüências no teste de exatidão.

Detecta-se na Fig. 4.28, que os melhores resultados com menor valor do erro porcentual, se concentram em $l \sim 1000$ para todas as seqüências, portanto, uma

teleoperação baseada em AR com marcadores padrões nessas mesmas condições experimentais pode ser realizada com segurança dentro de uma faixa de $500mm$ até $1500mm$ da câmera; o que é uma possível vantagem no caso de manipuladores submarinos pois eles operam nesse espaço de trabalho.

4.6 Tendência e Oportunidades

Dentre das inúmeras possibilidades da AR, no caso da robótica é necessário diferenciar entre os conceitos de *Tendências* e *Oportunidades*. As *tendências* mostram como poderiam ser resolvidas no futuro aquelas problemática atuais da manipulação robótica submarina, dotando automação a estes processos. Enquanto isso, as *oportunidades* são definidas aqui como aquelas aplicações que poderiam ser desenvolvidas em curto prazo para a teleoperação robótica em ambientes não estruturados.

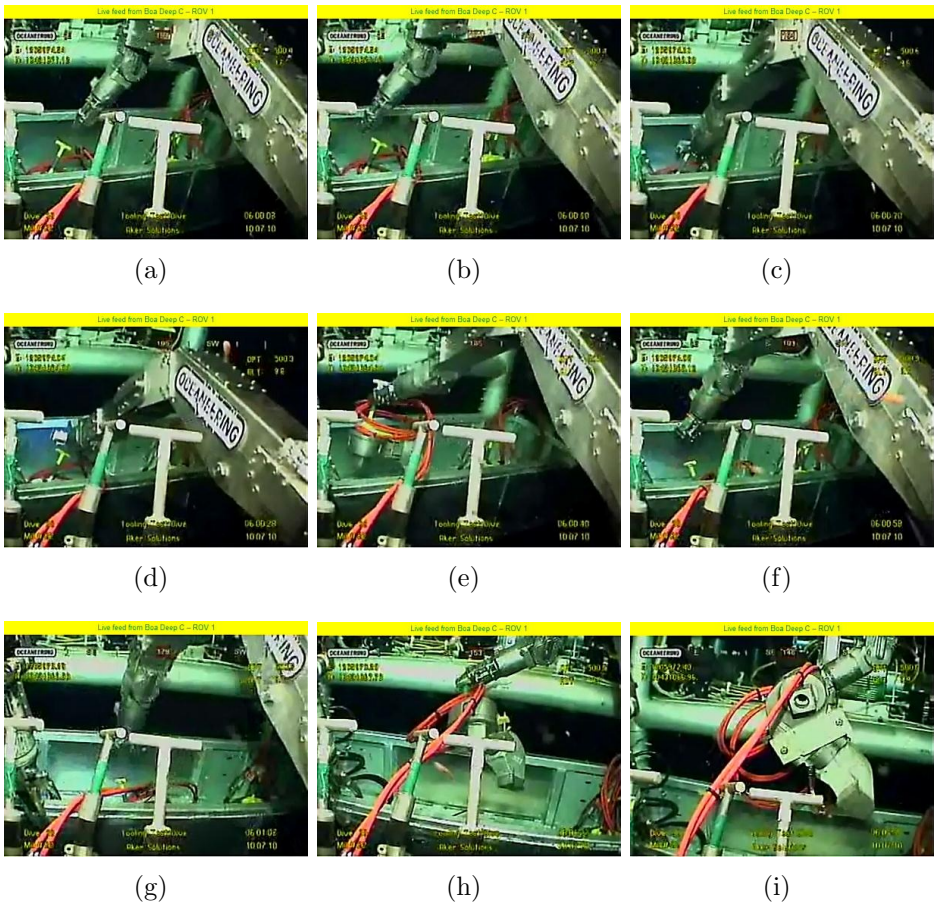
4.6.1 Casos de Aplicação Futura

Provavelmente, uma das operações com maior grau de destreza exigida pelos pilotos de ROV em ambientes não-estruturados é a tarefa de interagir com múltiplos corpos. Na Fig. 4.29 apresenta uma serie de manobras para retirar uma ferramenta de uma gaveta. As situações (a) e (b) indica manobras de aproximação ao alvo, devido as constantes flutuações o piloto deve corrigir constantemente os movimentos do manipulador. Em (c) mostra o processo de agarramento mas a situação (d) indica uma falha no processo, sendo necessário abrir e reposicionar o efetuador final de diversas formas. Neste caso a realidade aumentada conseguiria ajudar nesse processo de obter um pegada correta.

Já na situação (e) apresenta-se o efetuador com a ferramenta, mas em (f) deve ser liberada a ferramenta, pois o piloto não consegue os movimentos certos do manipulador para tirar a ferramenta da gaveta. Em (g) a disposição da cena mostra que os pilotos do ROV movimentaram a base do robô para facilitar a destreza da operação; em (h) tem-se que realizar novamente a operação de agarramento e em (i) a operação de retirada da ferramenta do local.

Para solucionar as dificuldades apresentadas em (e): destreza, e (i): retirada da ferramenta, pode ser implementado um sistema de geração de rotas livre de colisões para garantir a maneira viável e o sucesso da operação. Concluindo, da mesma maneira que os pilotos usam o video para realizar a teleoperação, a AR pode ser usada para um controle semiautônomo no processo de agarramento em colaboração de estrategias de geração de rotas para retirar objetos sem colidir com seu ambiente.

Outra tendência da AR para a teleoperação é a assistência da manipulação com dois manipuladores. A Fig. 4.30 apresenta um exemplo típico, onde são requeridas



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

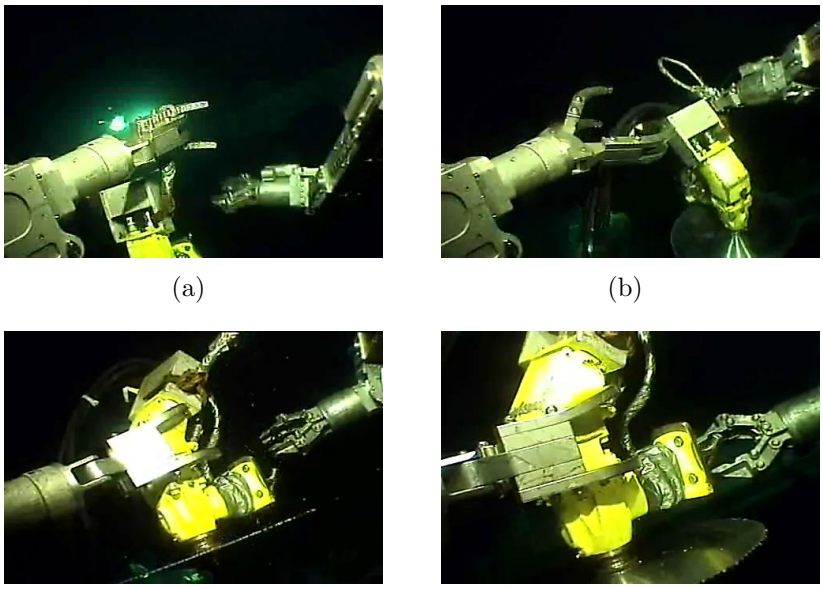
Figura 4.29: Capturas das diversas situações na operação de manipulação de ferramentas.

várias manobras entre um manipulador ao outro para posicionar adequadamente uma ferramenta. Lembre-se que em geral o manipulador da direita de maior destreza é usado para as tarefas de manipulação; e o braço esquerdo, com menor número de GDL, é usado para tarefas de apoio.

Estas operações bi-manuais robóticas requerem maior tempo de execução comparado com a destreza bi-manual humana. Três aspectos devem-se abordar para desenvolvimento de tendências futuras da AR:

- Posicionamento certo dos EF em relação às câmeras. Deve-se lidar com dois temas : a) reconhecimento dos efetuadores finais, por exemplo usando os chamados "marcadores naturais". E b) tratamento das oclusões entre objetos reais, deve-se inferir a posição deles usando estratégias AR mais evoluídas ou com o cálculo da cinemática direta.

- Sistema de detecção de elementos flexíveis. É de uso comum cabos nos elementos de manipulação submarina (Fig. 4.30(b)). O modelamento e reconhecimento destes elementos são ainda tema de pesquisa na AR.



(a) (b) (c) (d)

Figura 4.30: Capturas das diversas situações na operação de manipulação conjunta entre manipuladores.

- Cinemática cooperativa entre dois manipuladores. Requerem-se algoritmos de posicionamento cinemático, por exemplo, com o estudo da cinemática de cadeias fechadas ou com os algoritmos de planejamento dos movimentos (*Path Planning*).

4.6.2 Perto à Realidade

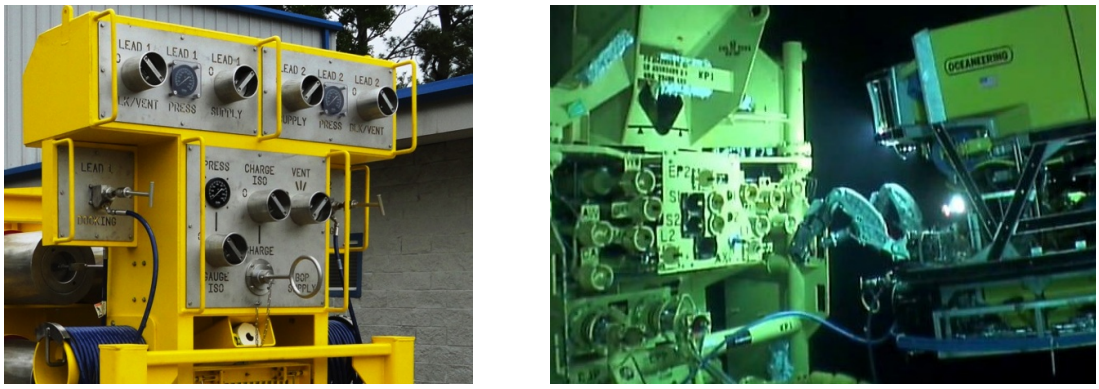
A seguir são explicados alguns dos usos desta tecnologia AR e que podem ser aplicados em curto prazo (*oportunidades*):

Panéis. Uma das oportunidades das técnicas explicadas neste capitulo é a reconstrução ou virtualização dos painéis de operação para ROV. Por exemplo, na Fig. 4.31 as letras de um painel podem ser usadas como marcadores de AR, a virtualização do ambiente do ROV permite conhecer as distâncias entre objetos reais da cena.

Dois processos podem ser derivados da virtualização dos painéis:

- Com os cálculos feitos com AR é possível determinar a posição e orientação do ROV em relação com ao painel ou uma dada estrutura (por exemplo, um *manifold*). Os dados obtidos podem ser usados em novos algoritmos de estabilização dinâmica para coordenar os movimentos do ROV.

- Com a virtualização é possível mostrar para o piloto como atingir o alvo com diversas configurações do manipulador, facilitando a tomada de decisões antes de movimentar o braço robótico. Nota-se que com uma alerta do sistema de



(a) (b)

Figura 4.31: Paneis para intervenções submarinas com ROV. (a) Painel com ROV em ambiente real. (b) Diversas peças que o manipulador usa nas intervenções.

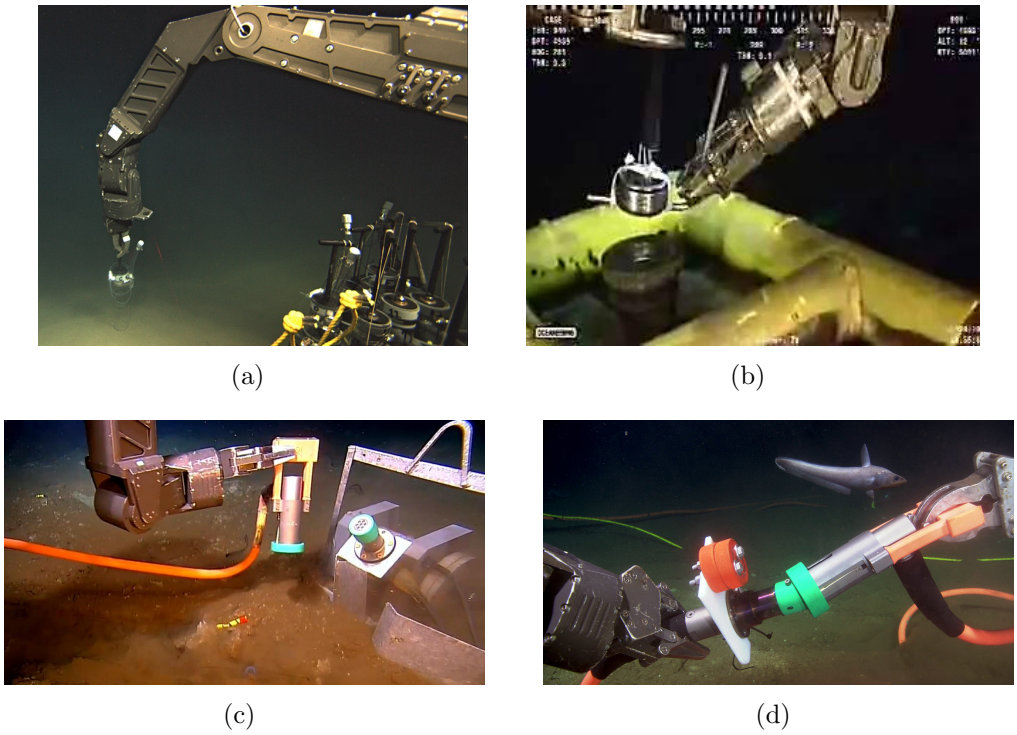
virtualização, os pilotos podem conhecer se o manipulador pode alcançar uma ou várias alavancas com a mesma posição da base do ROV.

Manobras de Aproximação Levando-se em conta o que foi exposto nas secções anteriores e nos testes, possível determinar a pose de um objeto virtualizado em relação com outro. Desta maneira, se um braço robótico está manipulando uma peça em relação a outra, um sistema de AR pode entregar informação aos pilotos do nível de proximidade e orientação relativa entre as duas peças virtualizadas. A Fig. 4.32 apresenta quatro casos reais onde pode ser aplicada esta técnica.

No caso da Fig. 4.32(a) a peça que carrega o efetuador final deve se posicionar dentro da caixa, ou seja, a peça e a caixa requem ser virtualizados. A aplicação AR pode subministrar visualmente uma linha entre a peça e um ponto designado para ser liberada na caixa, a semelhança da linha vermelha indicada na Fig. 4.18. Uma aplicação mais evoluída seria para o caso da Fig. 4.32(b), que é o típico caso de perda da sensação de profundidade apresentado na Fig. 4.14. Para este caso, além de uma linha virtual vermelha que mostre a proximidade da peça respeito ao buraco na Fig. 4.32(b) é possível traçar uma linha no eixo do buraco e quando o eixo da peça coincida com ela, o piloto perceba que pode realizar manobras de operação fina para atingir o alvo.

De forma similar, para o caso apresentado na Fig. 4.32(c), além das facilidades de linhas de visualização para orientação, o piloto poderia visualizar na tela dados de cumprimento entre as bordas da peça e o receptáculo no painel de maneira que o encastre seja conferido. Observe-se que nesse tipo de visualização não é preciso conhecer a posse do EF, as distâncias e orientações podem ser apresentadas com uma simbologia visual ao piloto para facilitar sua tarefa de teleoperação.

A cena da Fig. 4.32(d) é uma variação dos três primeiros casos, pois agora precisa-se virtualizar duas peças agarradas pelos efetuadores finais dos manipuladores (dupla



(a) (b)

(c) (d)

Figura 4.32: Capturas de diversos casos de operação de alineação na manipulação de peças com manipuladores.

manipulação). As ajudas visuais, como linhas e dados na tela, podem ser as mesmas. Provavelmente esse caso de AR deveria ter melhor desempenho, pois o sistema de dupla manipulação é tratado como uma visualização endógena.

Acrescentando a Visão. Em geral os sistemas de controle dos manipuladores comerciais são fechados e poucas modificações podem ser realizadas neles. Entretanto, há diversas melhoras da AR usando os atuais sistemas de visão incorporados nos ROVs, por exemplo:

- Em alguns casos o efetuador final possui uma câmera simples que permite ao piloto olhar mais de perto um objeto para manipulá-lh0. Essa câmera propicia uma melhora dos níveis de exatidão e precisão de um objeto virtualizado, sendo utilizada somente com a AR ou junto com a câmera principal;

- A maioria dos ROV de trabalho possuem uma câmera com opção de zoom e movimentos pan/tilt. Para aproveitar essas características na AR, devem-se modificar as equações de transformação, usando matrizes de translação no caso do zoom e de duas rotações para a função pan/tilt. Para um caso real, essas novas matrizes de transformação podem variar de maneira discreta, em outras palavras, precisa-se calibrar os incrementos dos elementos das matrizes em relação com os movimentos reais da câmera;

- Além das imagens que o piloto vê na tela, com um sistema AR que realize uma reconstrução do ambiente é possível mostrar uma panorâmica externa em outra tela, apresentando a semelhança de um simulador, o comportamento do manipulador ou do ROV inteiro em um ambiente virtual que representa em tempo real as condições reais. Muitas vezes esse tipo de visualização em terceira pessoa permite uma melhor compressão do comportamento do manipulador e é feita de maneira intuitiva pelos pilotos.

Capítulo 5

Estudo do Planejamento de Rotas Baseado em Amostragem

O capítulo anterior apresentou a Realidade Aumentada como um sistema de sensoriamento que permite obter de uma fonte de vídeo a posição e rotação de objetos reais em relação à câmera. Esses dados podem ser aproveitados para determinar se um manipulador consegue se movimentar no ambiente sem colidir com os objetos reais.

Como contribuição desta tese, o presente capítulo introduz novas estratégias de geração de rotas livres de colisões para solucionar o problema dos movimentos autônomos de robôs que operam em ambientes não estruturados.

Os algoritmos aqui introduzidos pertencem ao campo do planejamento de movimentos baseados em amostragem apresentado no capítulo 2. Convém lembrar que os conceitos apresentados não estão restritos ao estudo de caso de manipuladores submarinos, portanto sua generalidade permite usar estes algoritmos em outras áreas, tais como a robótica móvel ou sistemas multirobôs.

Baseando-se nos conceitos do planejamento probabilístico o capítulo será composto por três partes: a primeira estabelece como obter a cinemática inversa para manipuladores resolvendo os casos de multiplicidade; a segunda parte ilustra como resolver o planejamento dos movimentos em ambientes dinâmicos; por último, a terceira parte apresenta conceitos para obter a rotas mais curtas possíveis. As análises e demonstrações destas novas estratégias são conferidas em diversos cenários usando as simulações no ambiente de programação numérica MATLAB.

5.1 Cinemática Inversa

Bem como foi apresentado nos capítulos anteriores, os manipuladores sofrem de multiplicidade de soluções da cinemática inversa. A Fig. 5.1 mostra dois casos

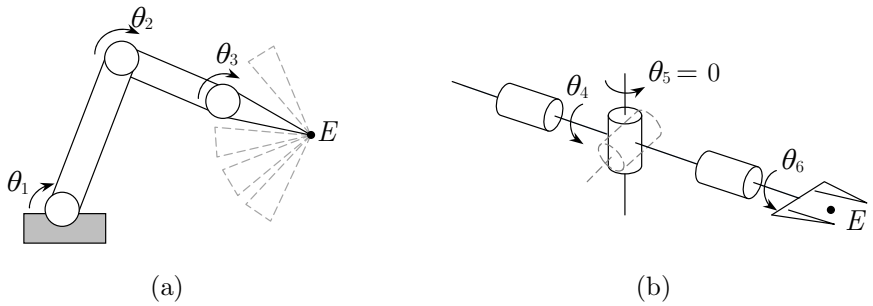


Figura 5.1: Exemplos de multiplicidade da cinemática inversa. (a) Redundância posicional no robô planar 3-GDL. (b) Pulso esférico em configuração singular.

típicos, o caso (a) é um robô planar 3-GDL apresentando a redundância posicional, onde para uma desejada posição do efetuador final (ponto E) o terceiro elo possui inúmeras orientações; já o caso (b) mostra a condição singular de um pulso esférico, onde a quarta e sexta junta são colineares, neste caso o ângulo da quinta junta é $\theta_5 = 0$, por conseguinte, essa quinta junta pode girar no eixo da quarta junta e a sexta compensa o giro mantendo a mesma posição e orientação desejada para o efetor final. Ambos os casos produzem infinitas soluções da cinemática inversa, causando indeterminações matemáticas nas soluções fechadas e dificuldades no controle das soluções baseadas no jacobiano.

Em principio qualquer nível de redundância produz multiplicidade da cinemática inversa. Um estudo que realiza uma comparação dos diversos métodos disponíveis para a resolução da redundância é apresentado em [149], onde os autores avaliam experimentalmente os resultados em um robô antropomórfico de 7-GDL. Provavelmente um aspecto que influencia fortemente as soluções da cinemática inversa são os limites físicos das juntas. Em [150] é parametrizada a cinemática inversa de um robô de 7-GDL para evitar os limites das juntas conforme o conceito de ângulo de braço (*angle arm*), em outras palavras, restringido um ângulo do pulso o sistema muda para uma condição não redundante. Já em [151] é utilizado o mesmo conceito para uma evitar colisões com os obstáculos.

Diversos métodos baseados no jacobiano foram propostos no passado para resolver a cinemática inversa, um texto que realiza a comparativa entre os métodos mais representativos em função de seu desempenho na evasão dos limites articulares e o tempo de resposta é apresentado em [152].

Em relação com às complicações apresentadas com configurações singulares, na ultima década foram estudados os métodos iterativos como parte da solução. Por exemplo, em [153] é calculada a cinemática inversa minimizando as discrepâncias entre a posse meta e a atual posse do efetuador final para um robô 3-GDL espacial. Em [154] os autores indicam como transformar o problema da cinemática inversa em um sistema de otimização resolvido por algoritmo de evolução diferencial.

Uma abordagem interessante para resolver a cinemática inversa vem desde o campo da computação gráfica, onde são aplicados conceitos geométricos, uma das estratégias representativas é a chamada CCD das siglas em inglês *Cyclic-Coordinate Descent*. Um estudo que recompila as variantes de CCD é mostrado em [155].

A maioria dos métodos para solucionar a cinemática inversa não estabelecem uma relação direta com a presença de obstáculos. Em principio, os algoritmos baseados em amostragem podem lidar com estes tipos de condições de colisões. Nesta parte do capítulo, é apresentado um novo algoritmo que usa a amostragem como sistema de busca da cinemática inversa. Não obstante, para entender seu funcionamento, a seguir é mostrada a generalização do problema da cinemática inversa usando o conceito de espaço de configuração do campo do planejamento dos movimentos (*Path Planning*).

5.1.1 Conceituação no Espaço de Configuração

Lembrando o conceito geral de planejamento de movimento apresentado na Fig. 2.12, um robô é caracterizado como um ponto em seu espaço de configuração $\mathcal{Q}$ e a solução do movimento livre de colisões no espaço de trabalho $\mathcal{W}$ corresponde à rota continua desde uma configuração inicial $\mathbf{q}_{start}$ até $\mathbf{q}_{goal}$ no $\mathcal{Q}$.

Um *caso estendido da cinemática inversa* foi definido nesta tese como o problema de alcançar uma desejada pose do EF desde uma configuração inicial sem colidir com os objetos do ambiente. Note-se que para cumprir esse objetivo, deve-se estabelecer uma rota livre de colisões sem conhecer a configuração final.

No caso de manipuladores seriais a cinemática inversa pode ter uma, múltiplas, infinitas ou nenhuma solução, portanto, os atuais métodos de planejamento de movimentos (seção 2.4) não conseguem uma solução da cinemática inversa devido à ausência da configuração final. A Fig. 5.2 representa de maneira visual o *caso estendido da cinemática inversa* definido nesta tese.

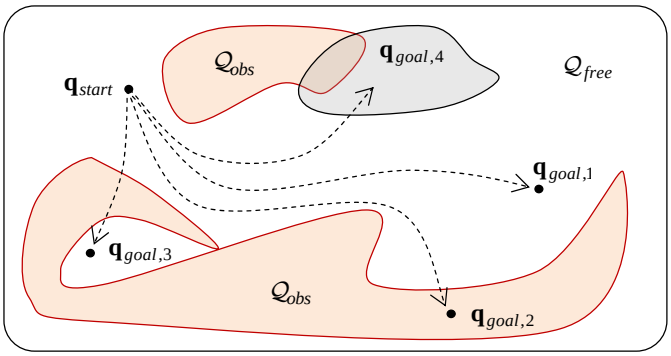


Figura 5.2: Conceituação do problema estendido da cinemática inversa no espaço de configuração.

Uma solução da cinemática inversa é um vetor com os valores das juntas $\mathbf{q}_{goal,i}$ para uma determinada pose do efetuador final EF_{goal} descrito no espaço de trabalho $\mathcal{W}$. Esta concepção geral indica que existem diferentes configurações para a mesma condição do EF, no entanto, atingir estas soluções desde uma configuração atual $\mathbf{q}_{start}$ é basicamente um problema de *Path Planning*.

Fig. 5.2 apresenta essas soluções como pontos no espaço de configuração $\mathcal{Q}$. A configuração $\mathbf{q}_{goal,2}$ indica que robô consegue atingir a pose do EF, mas, para lograr essa configuração em algum momento a estrutura do robô colide com algum objeto no $\mathcal{W}$, essa condição é representada como um ponto dentro do obstáculo $\mathcal{Q}_{obs}$ no $\mathcal{Q}$. Conseqüentemente, não é possível traçar um caminho desde $\mathbf{q}_{start}$ e o caso estendido da cinemática inversa não está resolvido. Por outro lado se uma solução da cinemática inversa está no $\mathcal{Q}_{free}$, tem-se a oportunidade de obter uma rota livre de colisões, como de $\mathbf{q}_{start}$ até $\mathbf{q}_{goal,1}$, mas a solução pode ser inacessível como em $\mathbf{q}_{goal,3}$, indicando que o manipulador não possui uma combinação de movimentos para atingir o alvo sem sofrer colisões. Um caso especial da multiplicidade da cinemática inversa é representada por $\mathbf{q}_{goal,4}$, onde temos uma região de soluções no $\mathcal{Q}$ que mistura as três condições anteriores.

A região $\mathbf{q}_{goal,4}$ é a representação do caso de multiplicidade da cinemática inversa mostrada na Fig. 5.1. A solução destes casos sem produzir colisões com o ambiente é um tema pouco estudado na literatura. A próxima seção mostra como obter a solução geral da cinemática inversa estendida usando planejamento de rotas baseado em amostragem.

5.1.2 Algoritmo IKURT

Esta seção apresenta a estrategia desenvolvida nesta tese, capaz de resolver o caso estendido da cinemática inversa. O algoritmo é chamado de Cinemática Inversa Usando Árvores Aleatórias Para Exploração Rápida (*Inverse Kinematics Using Random Trees Exploration algorithm*) ou IKURT para abreviar. IKURT empresta as características do algoritmo básico RRT apresentado na seção 2.4.6 para explorar sistematicamente o espaço de busca e descobrir uma configuração válida $\mathbf{q}_{goal}$.

O algoritmo IKURT tem dois variantes. A primeira usa a distância euclidiana entre as coordenadas atuais e desejadas do EF no espaço $\mathcal{W}$ como sistema atraente para solução do problema da cinemática inversa posicional. A segunda variante usa dois subespaços de validação para determinar a proximidade com a pose completa de EE, o que significa a comparação da proximidade de posições e orientações.

A Fig. 5.3 ilustra o pseudocódigo do IKURT. As entradas do algoritmo são a configuração inicial $\mathbf{q}_{start}$ e o estado desejado do efetuador final EF_{goal}. De igual maneira que para os algoritmos baseados no RRT, o algoritmo começa criando uma

Algoritmo 4: IKURT(q_{start}, EF_{goal})

```
1  T.init(q_start);
2  while TimeRemaining() do
3      q_near ← NearestDistanceW(T, EF_goal);
4      if Connect(q_near, T) then
5          return Path();
6      else
7          q_rand ←RandConfig();
8          q_near ←NearestDistanceQ(T, q_rand);
9          q_new ←Extend(q_near, q_rand);
10         if Valid(q_new) then
11             AddNodes(q_new)
```

Figura 5.3: Algoritmo genérico IKURT.

árvore T que é composto de vértices (*vertex*) e bordas (*edge*). O primeiro vértice é $\mathbf{q}_{start}$. O algoritmo é executado por um tempo determinado ou até que ele encontre a solução.

Em cada iteração é selecionada uma configuração $\mathbf{q}_{near}$ usando a função *NearDistanceW*. $\mathbf{q}_{near}$ é o vértice que está mais perto do estado EF_{goal}. Para isso, a função *NearDistanceW* realiza uma comparação entre todos os vértices (configurações do robô) e o estado desejado EF_{goal} no espaço $\mathcal{W}$ usando a cinemática direta de cada vértice.

Para o caso da cinemática inversa posicional, o valor de proximidade ao valor desejado é a distância euclidiana entre as coordenadas do EF, correspondente a uma configuração de teste i, e as coordenadas espaciais do EF_{goal}. Assim, a função *NearDistanceW* usa a expressão 5.1 no caso posicional.

$$\mathbf{q}_{near} := \min \{dist\,|EE_{goal} - EE(\mathbf{q}_i)|\} \text{ onde } i = 1, \cdots, \text{total de vértices.} \tag{5.1}$$

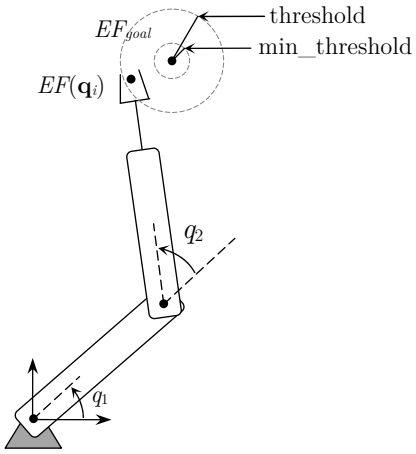
Sendo $\mathbf{q}$ um vetor em $\mathbb{R}^{1\times n}$, e n é o numero de juntas para um manipulador serial.

A estratégia de exploração do IKURT depende de dois processos, o primeiro é a função *Connect* (passo 4 e 5 da Fig. 5.3) que tenta gerar um caminho solução desde $\mathbf{q}_{start}$ até uma configuração $\mathbf{q}_{goal}$ o mais próxima que satisfaça à pose EF_{goal}. Quando a função *Connect* retorna um sinal falhado, o segundo processo é ativado (passo 7 até 11 da Fig. 5.3). Esse segundo processo tem o mesmo comportamento do RRT (Fig. 2.15) e é utilizado para explorar o ambiente para criar novos vértices usados pela função *NearDistanceW* no primeiro processo na iteração seguinte. Em outras palavras, o primeiro processo executa uma exploração direta no $\mathcal{Q}$ verifi-

Algoritmo 5: Connect(q_{near}, T)

1 q_{rand} ←RandConfig();
2 q_{new} ←Extend(q_{near}, q_{rand});
3 **while** TimeRemaining() & Valid(q_{new}) **do**
4 ($dist, q_{best}$) ← BestDistanceW(q_{near}, q_{new});
5 **if** $dist$ <threshold **then**
6 $\delta \leftarrow dist * \alpha$
7 **if** $dist$ >min_threshold **then**
8 **if** $q_{best} = q_{new}$ **then**
9 AddNodes(q_{new});
10 $q_{near} \leftarrow q_{new}$);
11 q_{rand} ←RandConfig();
12 q_{new} ←Extend(q_{near}, q_{rand});
13 **else**
14 AddNodes(q_{best});
15 **return** successful;

(a)



(b)

Figura 5.4: Operação *Connect* do algoritmo IKURT. (a)Pseudocódigo, proporcionalidade: $\alpha = \delta$/threshold. (b) Exemplo para robô planar 2-GDL.

cando a proximidade do estado desejado do EF no $\mathcal{W}$, quando ocorre uma colisão, o segundo processo realiza uma exploração eficiente no $\mathcal{Q}$ para obter novos vértices como candidatos para encontrar a solução da cinemática inversa.

A função *Connect* também realiza uma exploração do espaço de busca com árvores aleatórias, mas seu crescimento é dirigido diretamente à solução da cinemática inversa de maneira iterativa. O pseudocódigo dessa função é apresentado na Fig 5.4. A função *Connect* começa com um crescimento normal RRT usando as funções *RandConfig* e *Extend*, usando o parâmetro de crescimento δ (correspondente ao parâmetro ϵ do RRT). O processo exploratório pode continuar (passo 3) caso o tempo de processamento não ultrapassa um limite ou a configuração $\mathbf{q}_{new}$ seja válida, isto é, se o vértice e sua borda não têm colisão com objeto algum no $\mathcal{W}$.

A função *BestDistanceW* determina qual vértice selecionar, $\mathbf{q}_{near}$ ou $\mathbf{q}_{new}$, seu resultado é chamado de $\mathbf{q}_{best}$. A escolha é feita usando a mesma métrica da função *NearestDistanceW*, conseqüentemente, para o caso de cinemática inversa posicional a função *BestDistanceW* usa o critério da Eq. 5.1.

Quando o estado do EF, definido por $\mathbf{q}_{best}$, está fora do alcance da zona *threshold* (Passo 5) o crescimento à solução é normal. O δ decresce de maneira proporcional à distância do estado EF_{goal} dentro da zona limitada por o *threshold* (Passo 6). O algoritmo finaliza quando o estado do EF chega à distância *min_threshold*, cujo

valor pode ser ajustado dependendo da tarefa do manipulador e é uma medição da qualidade da solução da cinemática inversa obtida com IKURT.

O sistema de aproximação à solução é explicado de maneira simples com o robô planar 2-GDL da Fig 5.4. *dist* é o comprimento entre as coordenadas do EF, em $\mathbf{q}_{best}$, e as coordenadas de EF_{goal}. Quando *dist* é maior que *min_threshold* (passo 7) e sendo $\mathbf{q}_{new}$ uma melhor solução que $\mathbf{q}_{near}$, essa configuração é adicionada à árvore como um novo vértice (passo 8 até 10); pelo contrário, se $\mathbf{q}_{near}$ é a melhor solução, é realizada uma nova exploração (passo 11 e 12). Neste ponto o crescimento à solução depende do valor de δ, se delta decresceu porque *dist* é menor que *threshold* (do passo 5), os valores articulares q_1 e q_2 são incrementados de maneira menor e portanto os movimentos do EF são pequenos em relação com seus movimentos predecessores, permitindo uma convergência do IKURT. Finalmente se *dist* é menor que *min_threshold*, $\mathbf{q}_{best}$ é adicionado à árvore T e o algoritmo é finalizado e entregando da rota no passo 5 da Fig. 5.3 como a solução do caso estendido da cinemática inversa.

A explicação acima da operação de IKURT encaixa perfeitamente para obter a solução da cinemática inversa posicional, como é o caso da Fig. 5.1(a). Para solucionar o problema da cinemática inversa quando o estado desejado do EF envolve uma posição e orientação simultaneamente, a segunda variante de IKURT entra em jogo. Neste caso, além da métrica usada na posição, isto é, a medição do comprimento entre as coordenadas de posição atual do EF e a desejada, é essencial escolher uma métrica adequada entre uma orientação atual e a desejada.

Uma representação matemática comum da orientação é a matriz de rotação R definida no $SO(3)$. Diversas métricas para a orientação foram apresentadas no passado, um estudo comparativo destas estratégias é mostrado em [156]. Os autores concluem que para uma eficiência espacial e computacional é apropriado o uso dos quatérnios. Por conseguinte, aqui é utilizada a métrica baseada no produto interno dos quatérnios unitários. A métrica d_{rot} entre as rotação de teste R_i e a rotação desejada do EF R_e, pode ser descrita como:

$$d_{rot} = \arccos\left(\hat{\mathbf{q}}_{EF} \cdot \hat{\mathbf{q}}_i\right) \quad , \quad [0, \pi/2] \tag{5.2}$$

Onde o $\hat{\mathbf{q}}_{EF}$ e o quatérnio que representa a matriz de rotação R_e, semelhantemente, $\hat{\mathbf{q}}_i$ é a representação da matriz R_i obtida usando a cinemática direta para uma configuração de teste $\mathbf{q}_i$. A função arccos é utilizada analogamente à solução do ângulo do produto interno entre vetores cartesianos. A definição matemática do produto interno entre dois quatérnios $\hat{\mathbf{q}}_1 = \{a_1, b_1, c_1, d_1\}$ e $\hat{\mathbf{q}}_2 = \{a_2, b_2, c_2, d_2\}$ é como segue:

$$\hat{\mathbf{q}}_1 \cdot \hat{\mathbf{q}}_2 = a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2 \tag{5.3}$$

Além da descrição da orientação, a métrica de posição d_{pos} é facilmente descrita

com a distância euclidiana no espaço cartesiano e expressada como:

$$d_{pos} = dist\,|P_{e,goal} - P_{e,i}| \tag{5.4}$$

Sendo, $P_{e,goal}$ as coordenadas do estado desejado EF_{goal} e $P_{e,i}$ as coordenadas do EF calculadas da cinemática direta para uma configuração de teste $\mathbf{q_i}$.

A variação da estrategia IKURT para o calculo da cinemática inversa geral (posição e orientação) não possui mudanças nos pseudocódigos apresentados na Fig. 5.3 e Fig. 5.4. Entretanto, as funções *NearestDistanceW* (passo 3 da Fig. 5.3) e *BestDistanceW* (passo 4 da Fig. 5.4) precisam de uma nova métrica geral que outorga a condição de proximidade entre duas poses do EF. Neste caso, a área de trabalho deve ser estendida para o outro subespaço que é domínio de orientação representada por quatérnios. A Eq. 5.5 fornece uma forma simples de calcular a proximidade entre poses do EF.

$$\mathbf{q}_{near} := \min\left\{\sqrt{(\mu\, d_{pos})^2 + (d_{rot})^2}\right\} \tag{5.5}$$

Onde, μ é o fator de escala usado para interagir as representações de posição e rotação. Dependendo do valor de μ, a pose do EF converge de maneira rápida para o estado de posição em relação com o estado da orientação e vice-versa. Utilizando a Eq. 5.2, a métrica da orientação está no intervalo $[0, \pi/2]$, no caso da posição o intervalo pode ser determinado por a distância entre as coordenadas do EF na condição inicial e a condição desejada.

A seguir são apresentados diversos exemplos que mostram o funcionamento do IKURT, tanto para o caso da cinemática inversa de posição como para o caso de posição e orientação de maneira simultânea.

5.1.3 Solução do Caso Inverso Posicional

A Fig. 5.5 apresenta um exemplo básico para um manipulador linear 3-GDL, o espaço de trabalho contém dois obstáculos retangulares. A Fig. 5.5(a) mostra o problema da cinemática inversa estendida tipo posicional, onde o robô possui uma configuração válida inicial, indicada em preto; é requerido que seu EF, sistema $\{e\}$, chegue até a posição desejada EE_{goal} e simultáneamente que os elos não colidem com os obstáculos.

Já a Fig. 5.5(b) indica a solução obtida com IKURT, apresentando os estados intermediários do verde até ciano dos movimentos do robô desde a configuração inicial em preto até sua solução em azul.

Em virtude do robô ter três juntas, o espaço de configuração $\mathcal{Q}$ pode ser representado como um sistema cartesiano $\mathbb{R}^3$, isso permite que seja visualizada a maneira

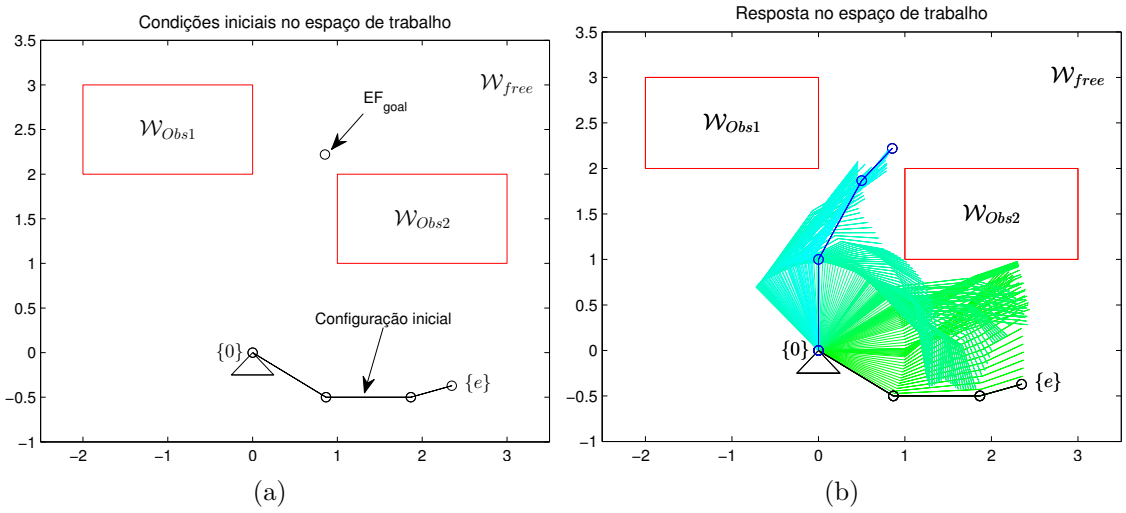


Figura 5.5: Exemplo da aplicação de IKURT para um manipulador planar 3-GDL.

como IKURT realiza a exploração do espaço de busca. Por conseguinte a Fig. 5.6 apresenta o caminho solução de verde ao ciano, de conformidade com a Fig. 5.5(b). Além disso, os vértices e bordas da cor azul indicam explorações mal sucedidas, e as cores vermelhas indicam as configurações não válidas que experimentam colisões no $\mathcal{W}$.

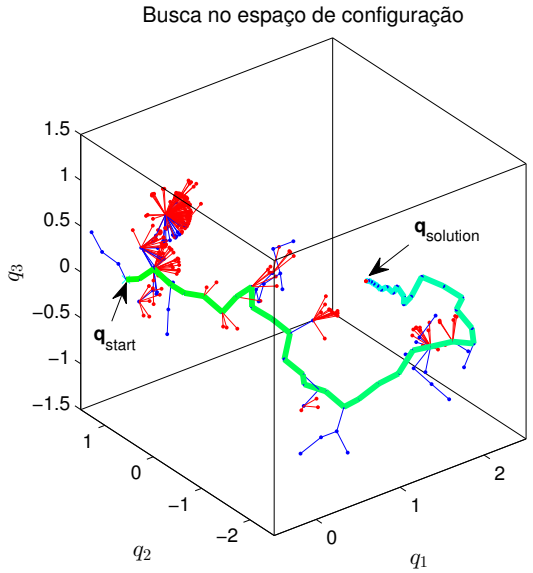
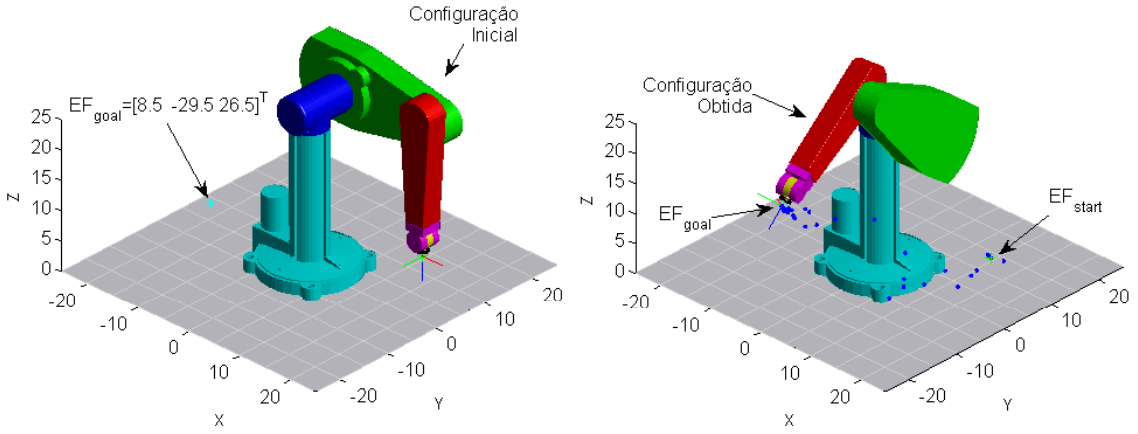


Figura 5.6: Representação da exploração feita com IKURT para um robô planar 3-GDL. Eixos coordenados em radianos.

Um exemplo da aplicação do IKURT para manipuladores industriais é mostrada na Fig. 5.7, onde um robô tipo PUMA 6-GDL precisa levar o EF desde a configuração inicial de repouso até as coordenas EE_{goal}. Neste caso, a cinemática inversa posicional possui infinitas soluções, isso ocorre pois os três últimos GDL são coincidentes no pulso. A Fig. 5.7(b) apresenta uma das soluções obtidas por IKURT, onde

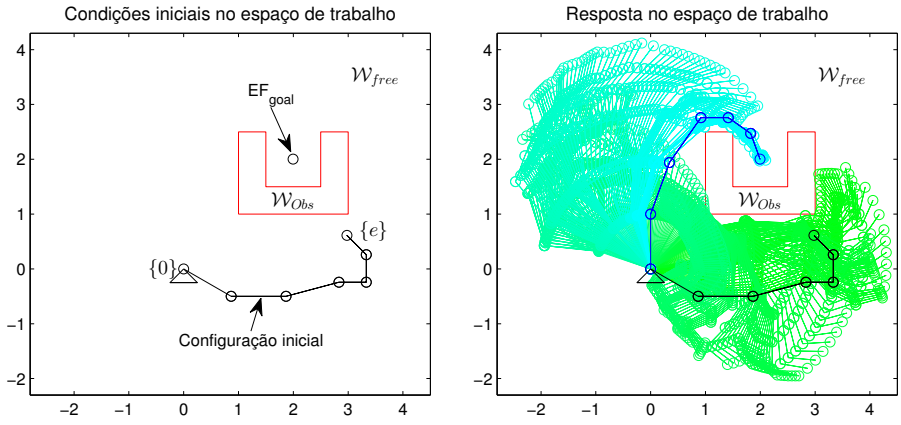


(a) (b)

Figura 5.7: Exemplo da aplicação de IKURT para um manipulador Puma 6-GDL no caso de cinemática inversa posicional.

os pontos azuis indicam as coordenadas do EF para as configurações intermediarias na busca da solução pelo algoritmo.

Outro exemplo em um manipulador redundante planar 6-GDL é oferecido na Fig. 5.8. Neste caso, a posição do EF está na concavidade do obstáculo em forma da letra "C", a qual é uma condição típica de mínimo local para os problemas de *path planning*. De maneira similar ao exemplo anterior, a Fig. 5.8(a) indica uma configuração inicial qualquer do manipulador e a Fig. 5.8(b) mostra a solução obtida pelo IKURT com os movimentos requeridos para evitar o obstáculo e atingir o alvo EF_{goal}. Convém ressaltar que o espaço de configuração $\mathcal{Q}$ não tem representação visual, pois é conceituado como um espaço de busca de seis dimensões.



(a) (b)

Figura 5.8: Exemplo da aplicação de IKURT para um manipulador redundante planar 6-GDL no caso de cinemática inversa posicional.

É apreciável que os movimentos traçados pelo manipulador às vezes tem longas amplitudes e com certeza não são ótimos, o que acontece devido o crescimento das árvores aleatórias no $\mathcal{Q}$. Quanto mais curta e suave seja a rota obtida no $\mathcal{Q}$, mais harmoniosos serão os movimentos dos elos e sem paradas bruscas no $\mathcal{W}$. Na terceira parte deste capítulo é apresentada uma análise e as estratégias para obtenção de rotas curtas.

5.1.4 Solução do Caso Inverso Geral

O robô planar 3-GDL não é a melhor escolha para demostrar a aplicação do IKURT para obter uma pose completa do EF (posição e orientação simultâneas), pois a cinemática inversa tem somente duas soluções. Nesse caso os métodos fechados são razoavelmente mais práticos. Por outra parte, um manipulador planar de 6-GDL é mais adequado para mostrar o funcionamento do IKURT pois é preciso lidar com a redundância própria do sistema. A Fig. 5.9 apresenta uma das solução obtidas com IKURT para esse tipo de robô redundante.

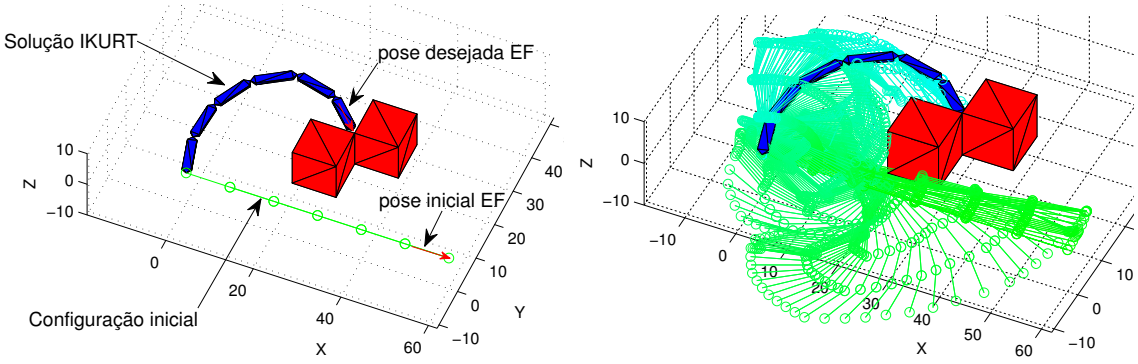


Figura 5.9: Exemplo da aplicação de IKURT para um manipulador redundante planar 6-GDL no caso de cinemática inversa geral.

O manipulador da Fig. 5.9(a) consiste de seis elos poliédricos, onde o espaço de trabalho apresenta dois cubos vermelhos que conformam os obstáculos na condição de mínimo local semelhante ao exemplo indicado na Fig. 5.8. As setas vermelhas indicam a pose inicial e final do EF. A Fig. 5.9(b) ilustra os movimentos obtidos com IKURT para chegar à condição desejada do EF. Para a execução deste exemplo, foi utilizada a biblioteca V-Collide que foi desenvolvida pela *University of North Carolina*[157] e que permite projetar os módulos de detecção de colisões entre objetos poligonais.

Por outro lado, para robôs típicos industriais de 6-GDL, que operam num espaço de trabalho tridimensional, a multiplicidade da cinemática inversa acontece em dois casos: CASO 1, por as singularidades cinemáticas entre as três primeiras juntas, que

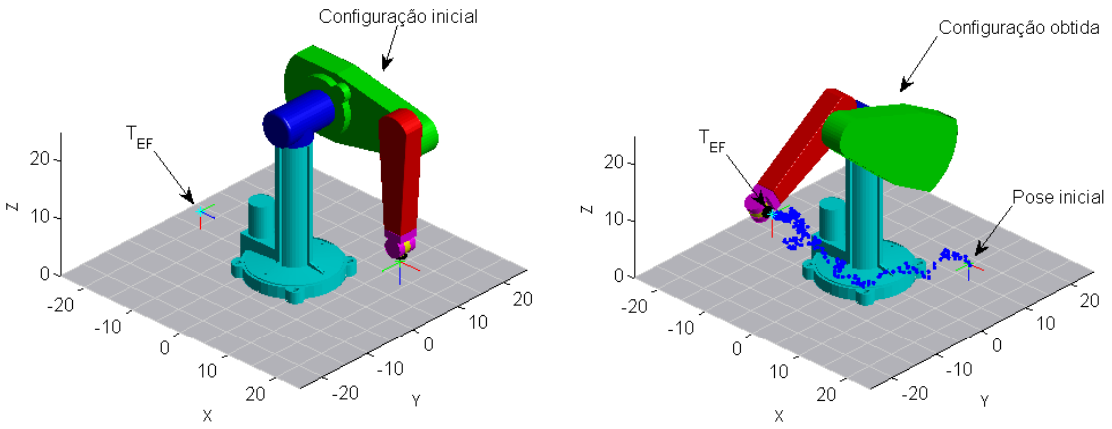


Figura 5.10: Exemplo da aplicação de IKURT para um manipulador Puma 6-GDL no caso de cinemática inversa inversa geral. (a) Condições iniciais. (b) solução obtida com IKURT.

em geral acontecem em número limitado de soluções, e cuja condição pode ser chamada de *multiplicidade discreta da cinemática inversa* (como é apresentado na Fig. 2.7). Caso 2, quando se tem singularidades mecânicas no pulso por desacoplamento das três últimas juntas (Ver Fig. 5.1(b)).

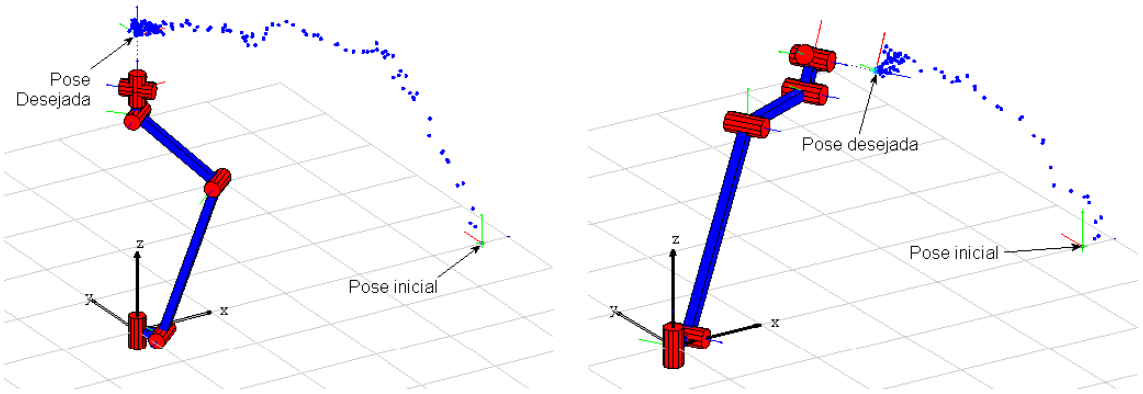
Por exemplo, para uma pose desejada do EF o manipulador PUMA 6-GDL pode acertar até 16 configurações, das quais só algumas são viáveis devido às restrições nas juntas. Essa condição é uma típica *multiplicidade discreta da cinemática inversa*. A Fig. 5.10 mostra a solução obtida pelo IKURT neste tipo de problemas. Lembre-se que a pose do EF pode ser especificada usando a matriz de transformação homogênea T_{EF} que determina sua condição desejada de posição $P_{e,goal}$ e rotação $R_{e,goal}$.

5.1.5 Soluções Sob Singularidades

Levando-se em consideração que o algoritmo IKURT é baseado inteiramente em amostragem (*Sampling-Based Algorithm*) é esperado que o método consiga resolver a cinemática inversa para as condições de singularidade sem o uso do jacobiano. Embora não haja uma comprovação matemática, a probabilidade de encontrar a solução é determinada exclusivamente por uma condição de proximidade ao estado desejado EF_{goal}, ao contrario das condições de atração feita pelo jacobiano inverso que produz indeterminações matemáticas nos métodos analíticos.

Aproveitando a análise realizado na seção 3.3.2, a Fig. 5.11 apresenta a resolução da cinemática inversa para dois casos de singularidade do manipulador TITAN-4 . Na Fig. 5.11(a) é situação singular na que a primeira junta pode girar livremente e a sexta compensa o movimento do braço preservando a pose desejada do EF (Fig. 3.5(b)). O segundo exemplo singular indicado na Fig. 5.11(b) corresponde à

análise realizada para a Fig. 3.6, onde apresenta-se multiplicidade da cinemática inversa quando a quinta junta tem valores de $pi/2$ ou $-pi/2$. Para visualizar os pontos intermediários do EF, na rota obtida por IKURT, a pose inicial do EF foi alocada quando o manipulador está na condição normal estendida.



(a) (b)

Figura 5.11: Aplicação do IKURT para casos singulares do manipulador TITAN-4. (a) Condição do efetuador final alinhado com eixo da primeira junta. (b) Condição de multiplicidade da cinemática inversa devida à quinta junta.

5.1.6 Semelhança e Diferenças de IKURT com Outros Métodos Baseados em Amostragem

Além do IKURT outras estratégias baseadas em amostragem foram desenvolvidas no passado para resolver o problema da cinemática inversa. No trabalho apresentado em [158] os autores utilizam uma heurística de penalidade para determinar a proximidade da pose do EF ao alvo, também adotam funções heurísticas do espaço de trabalho que implicitamente definem regiões objetivo de agarre no espaço de configuração. Uma extensão desse trabalho é mostrado depois em [159, 160], onde a convergência do algoritmo é baseada na pseudo-inversa do jacobiano.

Outro trabalho similar ao IKURT é o proposto em [161], onde a diferença é estabelecida em dois aspectos principais: o primeiro, os autores em [161] determinam a proximidade entre a pose do EF e o alvo usando um erro parametrizado da adição de posição e orientação; o segundo aspecto é mais determinante, pois os autores primeiro calculam uma configuração válida da cinemática inversa e logo calculam a rota livre de colisões entre a configuração inicial e a desejada usando o algoritmo bidirecional RRT-Connect [111]. Essa estratégia pode não convergir devido aos casos de soluções inacessíveis, como a configuração $\mathbf{q}_{goal,3}$ ilustrada na Fig. 5.2.

O algoritmo IKURT, da mesma maneira que outros métodos baseados em amostragem, possui a desvantagem do ajuste de parâmetros para obter respostas de

qualidade e melhorar os tempos de convergência. Uma alternativa interessante de melhora, consiste em usar uma combinação destes métodos com os novos algoritmos iterativos de solução rápida da cinemática inversa. Por exemplo, os algoritmos apresentados em [155, 162] que introduzem novas estratégias geométricas para reduzir os tempos de computação, inclusive até mais rápido que o tradicional método CCD (*Cyclic-Coordinate Descent*).

5.2 Planejamento de Rotas Dinâmicas

Os tradicionais algoritmos iterativos de resolução da cinemática inversa permitem obter uma configuração do manipulador para uma desejada pose do EF, além disso, as estratégias como IKURT permitem obter uma configuração válida na presença de obstáculos no espaço de trabalho do robô.

Para o estudo de caso do manipulador submarino, em condições ideais e estáticas, esses algoritmos cumprem o objetivo de resolver a cinemática inversa. Com o uso da AR apresentada no capítulo anterior, consegue-se mostrar visualmente aos pilotos dos ROVs uma configuração do manipulador que permite o agarre de um objeto virtualizado. Entretanto, quando existem obstáculos (objetos virtualizados) que estão em movimento o custo computacional não permite aplicação prática dos algoritmos de cálculo da cinemática inversa estendida.

Atualmente o calculo de rotas livres de colisões para ambientes dinâmicos está restringido ao processamento *off-line* de condições simuladas. Espera-se no futuro que o poder dos processadores gráficos aumentem, propiciando o calculo *on-line*, pois o principal impedimento continua sendo os inumeráveis chamados de detecção de colisão empregados por os algoritmos de amostragem.

A seguir, apresentam-se como as estratégias de planejamento de movimentos (*Motion Planning*) se encaixam no contexto dinâmico. Depois são expostos os novos algoritmos desenvolvidos nesta tese para geração de rotas dinâmicas como parte da solução do caso dinâmico. O suavizado de rotas permite diminuir o numero de movimentos requeridos na solução do problema. Essas estratégias utilizam os algoritmos para o suavizado e otimização das rotas que serão mostrados na terceira parte deste capítulo.

5.2.1 Contexto Dinâmico

Os diversos estudos sobre *Motion Planning* demonstram que é possível calcular movimentos complexos evitando colisões em ambientes 3D carregados de obstáculos. Esses logros são atribuídos ao desenvolvimento de algoritmos eficientes baseados em amostragem [11, 12, 106, 107, 111, 163, 164]. Na indústria, tais técnicas são

utilizadas para planejar a seqüência de montagem ou desmontagem de produtos compostos de muitas peças com geometria complicada [165].

Por outro lado, tem-se aplicações de sistemas robóticos reais que requerem planejamento de movimento em ambientes dinâmicos. Estes incluem manipuladores espaciais não-lineares[1] e os robôs tipo humanóides, que exigem equilíbrio dinâmico durante a execução das tarefas de locomoção. Outro exemplo de robôs em ambientes dinâmicos são os ROV de trabalho que interagem em ambientes não-estruturados. A dificuldade de planejar os movimentos desses robôs reside na alta complexidade geométrica 3D, na cinemática do movimento de 6-GDL sem colisões e as restrições dinâmicas de robôs e obstáculos. Devido a esta complexidade, tem havido poucos sistemas de planejamento que podem lidar com tais sistemas robóticos dinâmicos.

Uma maneira de controlar esses tipos de sistemas robóticos é formalizada por Laumond e seus colaboradores em [166], cuja representação dessa estratégia é recriada na Fig. 5.12. Esta abordagem constitui um esquema geral e prático que integra um planejador de rotas e um gerador de movimento dinâmico em duas etapas.

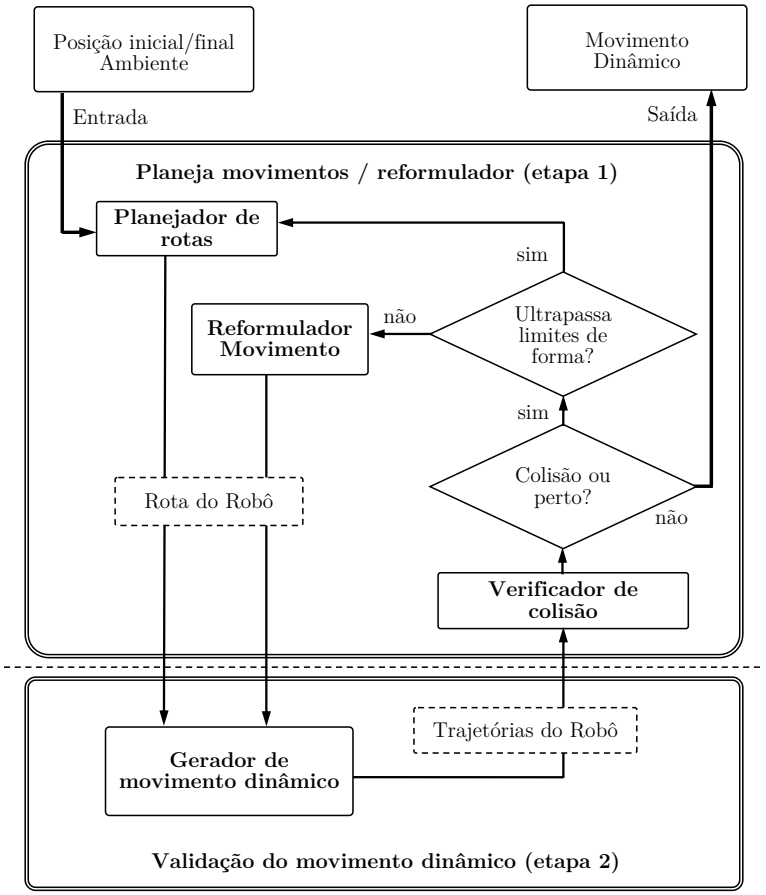


Figura 5.12: Abordagem dos sistemas robóticos dinâmicos que combina o planejamento de rotas e a geração de movimentos do robô. Imagem traduzida de [166].

[1] O "Robonauta" da NASA em http://robonaut.jsc.nasa.gov/, e o "Canadarm" em http://www.asc-csa.gc.ca/eng/canadarm/

Na primeira etapa o *Planejador de Rotas* encontra uma rota geométrica e cinemática livre de colisões no ambiente 3D. Depois, na segunda etapa o *Gerador de movimento dinâmico* transforma a desejada rota em trajetórias executáveis do robô real. Um controlador dinâmico dedicado pode ser colocado dentro, dependendo da aplicação.

As trajetórias geradas podem se desviar da rota planejada devido à dinâmica real do robô, a qual pode provocar colisões inesperadas com obstáculos. Portanto, o módulo *Reformulador de Movimento* é colocado na primeira etapa como mecanismo que interage com o *Gerador de movimento dinânico* para remover essas condições de colisão de maneira local. O *Reformulador de Movimento* pode operar com dois processos heurísticos: usando conceitos de tempo ou conceitos espaciais. Uma explicação mais detalhada do funcionamento desses processos é encontrado em [166]. Os processos dentro do *Reformulador de Movimento* operam por limites de forma ou limites operativos, que quando são ultrapassados, na primeira etapa indicam que deve-se recalcular uma nova rota livre de colisões. Em outras palavras, quando as trajetórias do robô, geradas pelo *Gerador de movimento dinâmico*, não satisfazem as condições limites dos processos de correção, o *Planejador de rotas* entra em jogo novamente.

Esta estratégia foi validada experimentalmente com robôs tipo humanóides quando o ambiente é totalmente conhecido e de maneira *off-line*. Vale ressaltar que uma característica importante para o sucesso desta técnica é um trabalho eficiente dos módulos *Verificador de colisão* e *Planejador de rotas*. Consequentemente, é desejável um algoritmo de geração de rotas livre de colisões que permita uma melhor interação com as condições dinâmicas.

Na seção seguinte são apresentados os conceitos de *estado de obstáculos* que são utilizados para formular um algoritmo que consegue obter rotas livre de colisões de maneira dinâmica com a capacidade de se adaptar às condições cambiantes do ambiente.

5.2.2 Estado de Obstáculos

O *Estado de obstáculos* refere-se à condição previsível ou não das geometrias dos obstáculos extrapolado no $\mathcal{Q}$.

Primeiramente, tem-se o caso quando o estado dos obstáculos é totalmente previsível no tempo. Matematicamente pode ser representada da seguinte forma:

- seja $T = [0, t_f]$ um intervalo de tempo de interesse;
- $\mathcal{O}(t) \subset \mathcal{W}$ um obstáculo variante no tempo $t \in T$;
- $\mathcal{A}(q)$ é um estado do robô devida a uma configuração $\mathbf{q}(t)$;

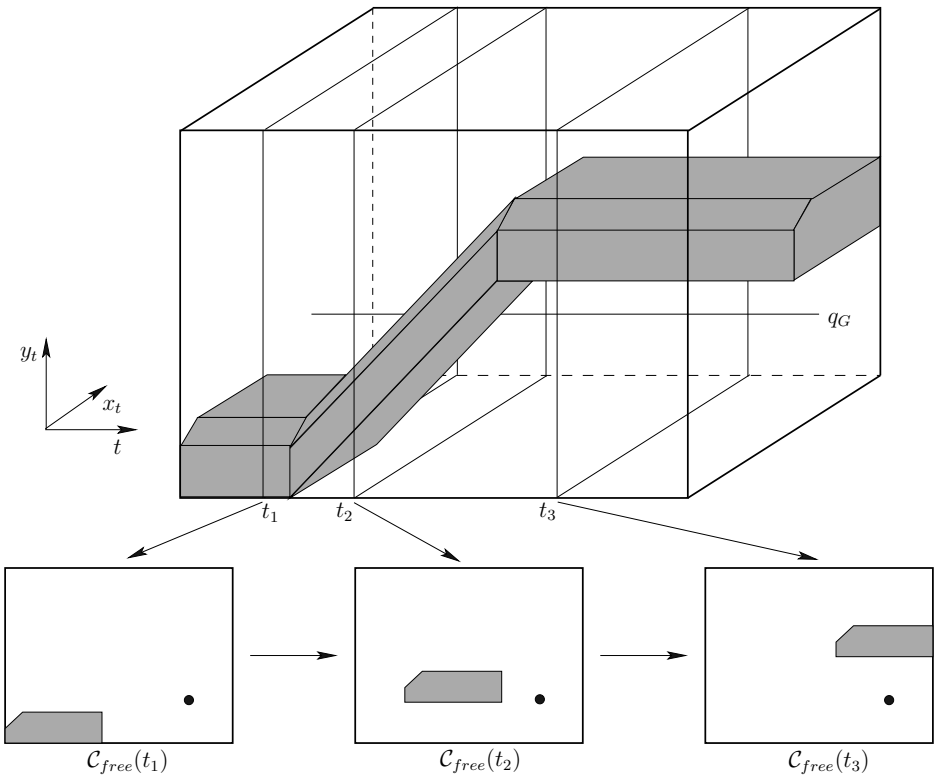


Figura 5.13: Exemplo da representação do espaço-tempo do movimento de um obstáculo planar. Imagem extraída de [11].

- o espaço de estado X é o produto cartesiano $X = \mathcal{C} \times T$, e denota o *espaço de configuração-tempo*;

- o estado $x \in X$ denotado como $x = (q, t)$ é formado por uma configuração $\mathbf{q}$ no tempo t;

- Uma região obstáculo, X_{obs}, definida no espaço configuração-tempo é definido como:

$$X_{obs} = \{(q, t) \in X \mid \mathcal{A}(q) \cap \mathcal{O}(t) \neq \emptyset\} \tag{5.6}$$

- Então, $X_{free} = X / X_{obs}$ é uma região configuração-tempo livre de colisões. Para um determinado $t \in T$ um corte do X_{obs} e X_{free} são obtidas as condições $\mathcal{C}_{obs}(t)$ e $\mathcal{C}_{free}(t)$ (Ver Fig. 5.13) utilizadas no caso estático, e formalizado como:

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O}(t) \neq \emptyset\} \tag{5.7}$$

- o estado $x_I \in X_{free}$ é designado como estado inicial, restrito como $x_I = (q_I, 0)$ para $q_I \in \mathcal{C}_{free}(0)$, em outras palavras, no tempo inicial o robô não possui colisões.

- o subconjunto $X_G \subset X_{free}$ é designado como região meta. Para o caso esta-

cionário da configuração meta, sendo $q_G \in \mathcal{C}$ então $X_G = \{(q_G, t) \in X_{free} \,|\, t \in T\}$;

- em conseqüência, um algoritmo deve calcular um caminho contínuo, monotônico no tempo, isto é designado como $\tau[0,1] \rightarrow X_{free}$, de maneira que $\tau(0) = x_I$ e $\tau(1) \in X_G$.

A solução para este tipo de problema dinâmico pode ser resolvido modificando alguns algoritmos do caso estático de planejamento do movimento. Esta formulação geral não possui outras restrições para τ, diante disso, o modelo do movimento do robô permite acelerações infinitas sem limitar a velocidade.

A descrição anterior pertence ao caso onde o comportamento do obstáculo é *totalmente previsível*. Outros casos onde é possível inferir os estados do obstáculo no tempo são os denominados: de *incerteza restringida* e *incerteza probabilística*. A Fig. 5.14 mostra uma representação simples dos tipos de problemas de planejamento dinâmico com noção do obstáculo no tempo. Existem estratégias para solucionar cada problema, e um estudo destas estratégias, utilizando a matemática anteriormente explicada, pode ser encontrado no livro de LaValle [11].

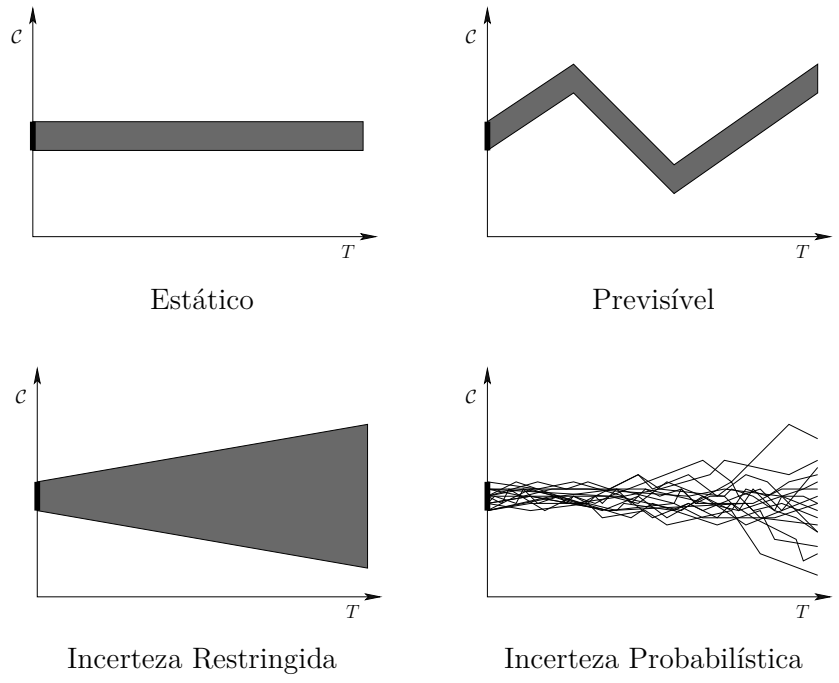


Figura 5.14: Representação planar da projeção do estado de um obstáculo no tempo para o problema de planejamento de movimento dinâmico.

Embora usando o conceito de espaço de configuração-tempo, seja possível usar alguns métodos de planejamento de movimento estático para o caso dinâmico, isto não acontece quando o estado do ambiente não pode ser previsto. Por exemplo, quando os obstáculos se movimentam de maneira aleatória ou mudam de forma.

Outro caso é quando um robô móvel está em um ambiente inexplorado, pois estado do ambiente pode parecer dinâmico porque um obstáculo estático é revelado.

Na seguinte seção é apresentada uma estratégia que lida com o *problema dinâmico não previsível*. A estratégia é baseada em amostragem, planejando as rotas de maneira dinâmica para se ajustar aos câmbios no espaço de busca. O algoritmo exibido é de tipo geral e pode ser utilizado em diversos problemas no âmbito do *Path Planning*.

5.2.3 Algoritmo DRRT-Con

Nos anos recentes, sobretudo após 2002, os algoritmos baseados na amostragem têm sido alvo dos pesquisadores para desenvolver estratégias de planejamento de rotas dinâmicas. Três algoritmos são de destaque: ERRT[167], DRRT[168] e MP-RRT[169]. A primeira formulação foi o ERRT, na qual, quando um obstáculo se movimenta é detectada uma colisão com a árvore existente e parte dela é cortada. Logo, um novo crescimento aparece dependendo de três condições de probabilidade. Infelizmente, a solução apresentada em [167] carece de detalhes na implementação e de resultados experimentais.

No ano 2006 aparece o DRRT, que trabalha crescendo uma árvore de $\mathbf{q}_{start}$ até $\mathbf{q}_{goal}$. A raiz da árvore não muda no tempo. Em geral a estratégia tenta cortar galhos menores e mais longe da raiz. Quando nova informação relativa ao espaço de configuração é recebida, o algoritmo remove os ramos inválidos da árvore e cresce os restantes focando-se com um valor de probabilidade, que empiricamente é ajustado em 0,4. Em resultados experimentais o DRRT supera em sobremaneira ao ERRT.

Paralelamente no ano 2007 é apresentado o MP-RRT. Esta estratégia mantém uma floresta F com sub-árvores desconectados no $\mathcal{C}_{free}$, mas essas não estão conectadas com à árvore principal T que contem a raiz $\mathbf{q}_{start}$. Usando uma dada probabilidade, o algoritmo tenta conectar a árvore principal T com um novo estado aleatório ou com uma raiz de F. Quando se apresenta colisão os galhos de T e F são cortados. Experimentalmente, o MP-RRT supera modestamente ao DRRT.

Subseqüentemente, foram apresentadas iniciativas distintas que misturam o algoritmo básico com outras técnicas. Por exemplo, em [170] aborda-se o problema de planejamento de trajetória dinâmica combinando técnicas simples e um algoritmo probabilístico multi-etapa, reconhecendo quando a busca local está presa e subseqüentemente reiniciando o planejador RRT. Em [171] usam o conceito de espaço de configuração-tempo e o filtro de Kalman para predizer estados dos obstáculos. Por outro lado, em [172] estabelecem para um robô móvel as heurísticas para determinar os obstáculos desconhecidos baseados na informação do rango de visibilidade dos sensores combinados com o planejamento dinâmico de rotas. E em [173] abordam

um modelo de aprendizagem de padrões, combinando a flexibilidade dos processos Gaussianos com a exploração do RRT.

Em geral, estes algoritmos são concebidos para ambientes dos robôs móveis integrando restrições não holonômicas como em [174] e podem incluir aspectos dinâmicos do robô como é apresentado em [175]. De conformidade com o desenvolvimento de novos algoritmos, espera-se que estas estratégias ajudem em outras aplicações, como é indicado em [176] para o transporte de células biológicas, em [177] para o andar autônomo bípede, para robôs de reabilitação e em ambientes não-estruturados [178], e em algumas abordagens em circuitos elétricos como em [179].

Neste trabalho foi desenvolvido o algoritmo chamado *Dynamic RRT-Connect* o simplesmente DRRT-Con, cujas principais características que o diferenciam dos outros algoritmos são:

1. Usa duas árvores de exploração com raízes $\mathbf{q}_{start}$ e $\mathbf{q}_{goal}$.

2. Verificação de colisão só na rota atual, em presença de colisão a rota é cortada de maneira bidirecional.

3. Tem capacidade de lidar com alvo móvel ($\mathbf{q}_{goal}$).

4. A raiz da árvore do robô ($\mathbf{q}_{start}$) é dinâmica, muda de posição no espaço de busca.

5. Uso da rapidez da exploração bidirecional proporcionada pelo RRT-Connect para realizar novas conexões entre as árvores.

6. Tendência de execução de rotas curtas dinamicamente usando pós-processamento.

7. Não requer ajuste de algum parâmetro de probabilidade para a busca da solução dinâmica.

O DRRT-Con usa estrategicamente o processo conectivo do RRT-Connnect, e sua filosofia pode ser resumida da seguinte maneira: primeiramente, é calculada uma rota entre as condições inicial e final usando o tradicional RRT-Connect (ver Fig. 2.19). Se o ambiente não possui mudança alguma, o robô é deslocado seguindo a rota calculada. Neste ponto a rota é cortada e recalculada com um algoritmo de pós-processamento usando a os dados da rota antiga. Quando aparece mudança no ambiente e uma colisão com a rota atual, a rota é cortada só onde ocorre a colisão. Logo é recalculada a rota com os dois fragmentos da rota antiga usando o RRT-Connect e o algoritmo de pós-processamento, posteriormente o robô se move na nova rota. A Fig. 5.15 mostra em pseudocódigo essa lógica do DRRT-Con.

Algoritmo 6: DRRT-Con(q_{start}, q_{goal})

```
1 Path←RRTconnect(q_start,q_goal);
2 Path←PostProcess(Path);
3 q_next ← q_start + ΔPath;
4 while q_next ≠ q_goal do
5     if NoEnvironmentChanges() then
6         UpdatePath_by_Robot(q_next,Path);
7     else
8         if PathCollsion() then
9             UpdatePath_by_Obstacle(q_next,Path);
```

Figura 5.15: Algoritmo genérico DRRT-Con.

Algoritmo 7: UpdatePath_by_Robot(q_{next},Path)

```
1 Path←CutPath(q_next,Path);
2 Path←PostProcess(Path);
3 q_actual ← q_next;
4 q_next ← q_actual + ΔPath;
```

Figura 5.16: Pseudocódigo da função *UpdatePath_by_Robot* do DRRT-Con.

Algoritmo 8: UpdatePath_by_Obstacle(q_{next},Path)

```
1 Path←CutPath(q_next,Path);
2 CutPoint_1←SearchPoint(q_actual, Path);
3 CutPoint_2←SearchPoint(Path, q_next);
4 T_a ←CutPath(CutPoint_1, Path);
5 T_b ←CutPath(Path, CutPoint_2);
6 Path←RRTconnect(T_a,T_b);
7 Path←PostProcess(Path);
8 q_actual ← q_next;
9 q_next ← q_actual + ΔPath;
```

Figura 5.17: Pseudocódigo da função *UpdatePath_by_Obstacle* do DRRT-Con.

A formalização das funções *UpdatePath_by_Robot* e *UpdatePath_by_Obstacle* são apresentadas como pseudocódigo na Fig. 5.16 e Fig. 5.17 respectivamente.

Em contrapartida, a Fig. 5.18 sintetiza visualmente o funcionamento do algoritmo DRRT-Con. Em uma dada iteração, uma rota livre de colisões é caracterizada por três pontos: a meta $\mathbf{q}_{goal}$, a situação atual do robô $\mathbf{q}_{actual}$ e sua próxima posição $\mathbf{q}_{next}$. Enquanto o robô se desloca de $\mathbf{q}_{actual}$ até $\mathbf{q}_{next}$, é testado o estado do ambiente com a função *NoEnvironmentChanges*. Esta função é executada no espaço de trabalho $\mathcal{W}$ e seu funcionamento depende da geometria do ambiente.

Quando não houver nenhuma colisão é ativada a função *UpdatePath_by_Robot*. A parte superior-direita da Fig. 5.18 resume esta função, estabelecendo uma rota curta entre $\mathbf{q}_{next}$ até $\mathbf{q}_{goal}$ e logo entregando as novas condições para a próxima iteração. Por outro lado, se é detectada apenas colisão na rota atual é ativada a função *UpdatePath_by_Obstacle*. Essa situação é o que diferença o DRRT-Con dos outros algoritmos, pois a verificação e feita só na rota e não na árvore que gerou a rota, reduzindo o número de chamadas de colisão que afetam os algoritmos de planejamento dinâmico baseados em amostragem.

Na Fig. 5.18, depois que é detectada uma colisão, os blocos apresentados na parte inferior representam a função *UpdatePath_by_Obstacle*, primeiramente são detectados os pontos de corte na rota, sua busca é bidirecional fragmentado a rota em duas partes, as quais são inseridas no algoritmo RRT-Connect como duas árvores que precisam de se conectar. Logo depois de conectar os segmentos de rota, a nova rota ingressa ao algoritmo de pós-processamento e finalmente estabelece as novas condições para a próxima iteração.

Um fator que influencia o desempenho desta estratégia é o algoritmo de pós-processamento. Uma descrição e análise dessas técnicas são mostradas nas seguintes seções.

O algoritmo DRRT-Con compartilha as siglas do algoritmo DRRT apresentado em [168], mas os conceitos são diferentes. Uma vantagem do DRRT é a conexão feita de maneira inversa, na qual o crescimento do RRT vai no sentido da configuração desejada para a configuração atual robô; isto de maneira similar ao sucesso do funcionamento do algoritmo D* para navegação com robôs móveis em ambientes desconhecidos. No caso do DRRT-Con essa condição e feita com a exploração bidirecional do RRT-Connect. Essa mesma característica permite ao DRRT-Con atingir um alvo móvel.

Um alvo móvel para o DRRT-Con é quando acontece uma desvinculação de $\mathbf{q}_{goal}$ da rota, e o restabelecimento da conexão é feita mais uma vez com RRT-Connect na etapa de verificação do ambiente e sem realizar pós-processamento. Na maioria dos casos, o deslocamento de $\mathbf{q}_{goal}$ em relação à rota é um valor pequeno, o que significa que a nova conexão pode ser efetuada rapidamente. Em aplicações reais,

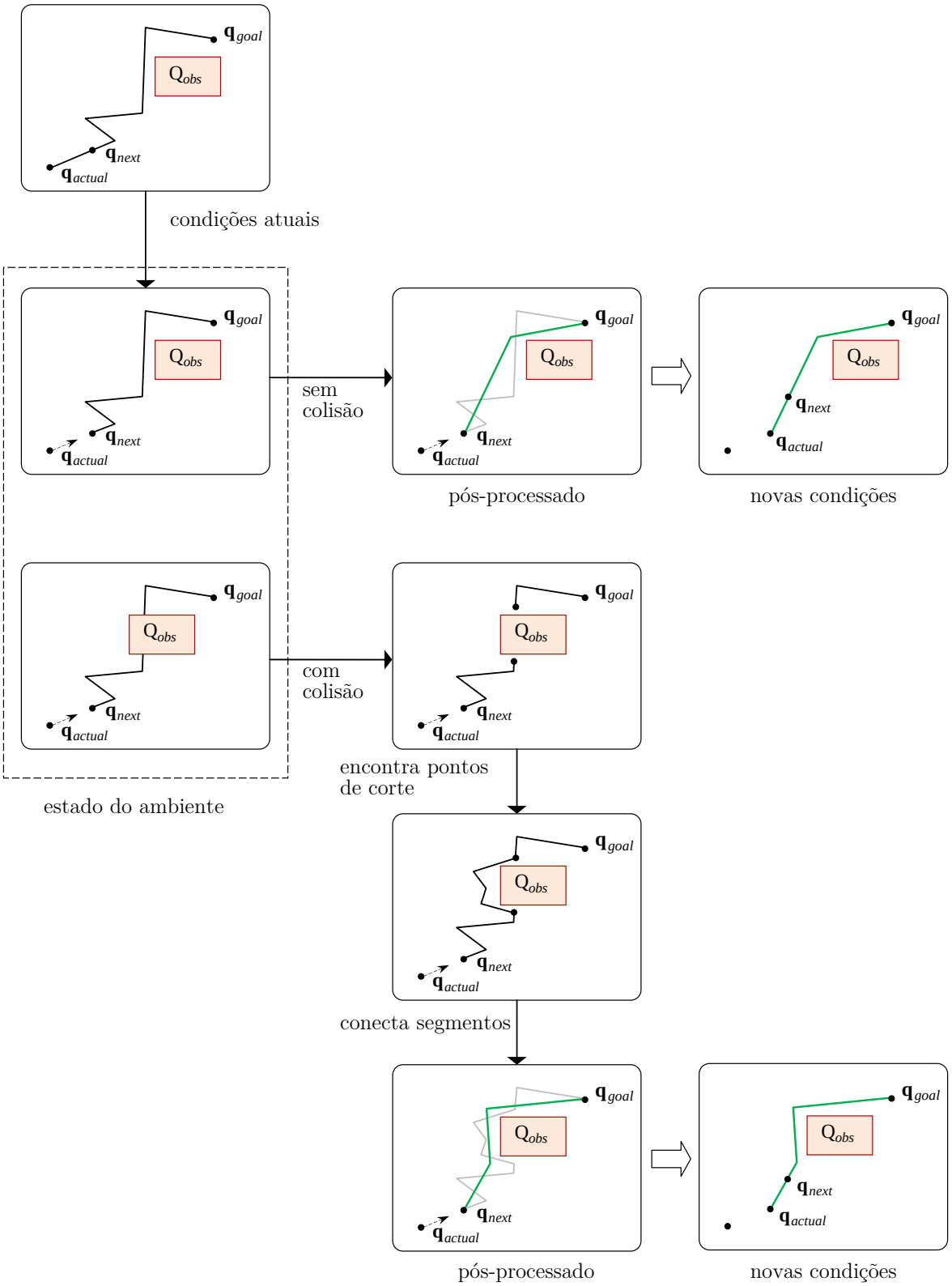


Figura 5.18: Representação da estratégia DRRT-Con.

o algoritmo RRT-Connect que trabalha dentro do DRRT-Con deve ser uma versão modificada do original RRT-Connect em dois aspectos: o primeiro é a finalização da busca por limite de tempo e não por um numero de iterações; o segundo aspecto é a formatação das entradas, pois na versão nova as entradas podem ser também árvores $\mathcal{T}$ e não só configurações $\mathbf{q}$.

5.2.4 Testes com DRRT-Con

Para ilustrar o funcionamento e capacidades do DRRT-Con, a seguir apresentam-se diversos testes realizados no MATLAB. Os resultados são obtidos em *tempo real de simulação*. Neste caso o termo *tempo real de simulação* refere-se que existe uma duração fixa do passo de iteração, fazendo que o algoritmo RRT-Connect interno do DRRT-Con responda nesse tempo determinado. Em conseqüência disso, a geração de rota dinâmicas e seu pós-processamento é limitado no tempo como poderia acontecer em uma aplicação prática.

Teste em espaço bidimensional simples. Este é um exemplo que indica como funciona o DRRT-Con. No primeiro bloco na esquerda da Fig. 5.19 mostra-se um ambiente bidimensional com dois obstáculos, cuja distribuição e forma estabelecem condições de mínimos locais para os tradicionais métodos de geração de rotas baseados em gradiente. O robô em sua configuração inicial $\mathbf{q}_{start}$ deve chegar até seu alvo $\mathbf{q}_{goal}$ quando o obstáculo superior em forma de letra "L"se desloca no sentido esquerdo. O segundo bloco da Fig. 5.19 mostra a inicialização do algoritmo DRRT-Con, onde é obtida uma rota livre de colisões usando o RRT-Connect (representado pela linha cinza) e uma rota mais curta obtida com o algoritmo de pós-processamento (representado pela linha verde). Esse segundo bloco é o resultado dos dois primeiros passos do algoritmo geral DRRT-Con mostrado na Fig. 5.15.

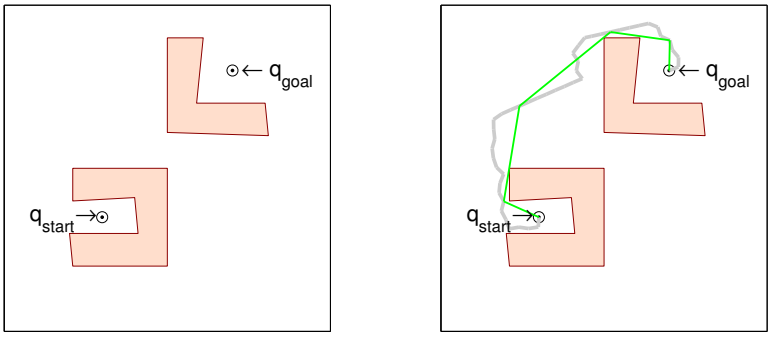


Figura 5.19: Exemplo do funcionamento do DRRT-Con, apresenta-se a preparação antes da primeira iteração.

Uma vez estabelecida uma rota inicial, o DRRT-Con executa o planejamento de rotas dinâmicas, do passo três até o passo nove do algoritmo geral RRT-Con. A

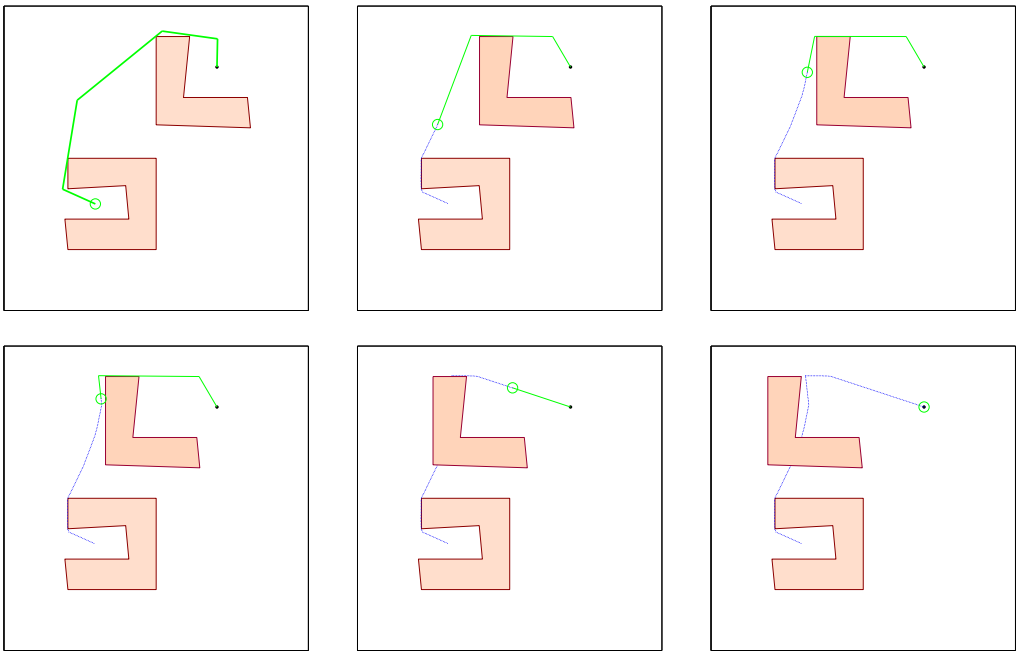
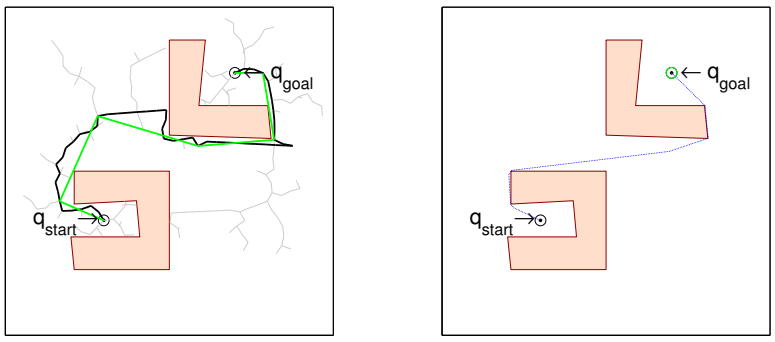


Figura 5.20: Exemplo do funcionamento do DRRT-Con, apresenta-se situações intermediarias quando o obstáculo móvel superior se movimenta para a esquerda.

Fig. 5.20 apresenta o resultado do planejamento dinâmico em seqüência de seis blocos ilustrando estados intermediários, começando pela condição inicial no bloco esquerdo superior, até a chegada do robô (círculo verde) no alvo, no bloco inferior direito. Observe-se que em cada estado é apresentada uma rota de pós-processamento que pode ser interpretado como "o futuro imediato" do movimento (linha verde). Ela não é ótima em seu estado atual, mas com vários deslocamentos do robô (seqüência de pontos azuis), a rota dinâmica tende ser mais curta possível dependendo da rota inicial obtida na Fig. 5.19.

Esta condição de obtenção de rotas curtas dinamicamente usando o DRRT-Con



(a) (b)

Figura 5.21: Comparação das rotas geradas em ambiente estático. (a) Com RRT-Connect mais pós-processamento. (b)Planejamento de rota dinâmico com DRRT-Con.

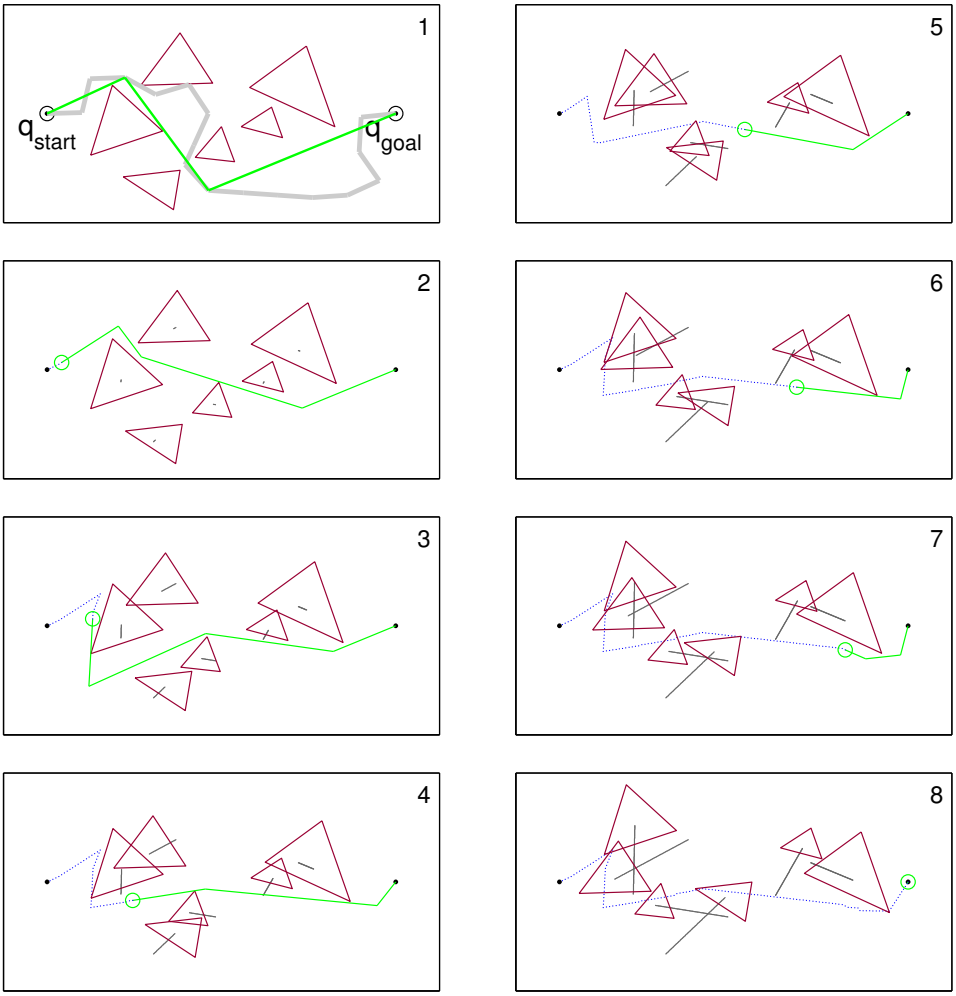
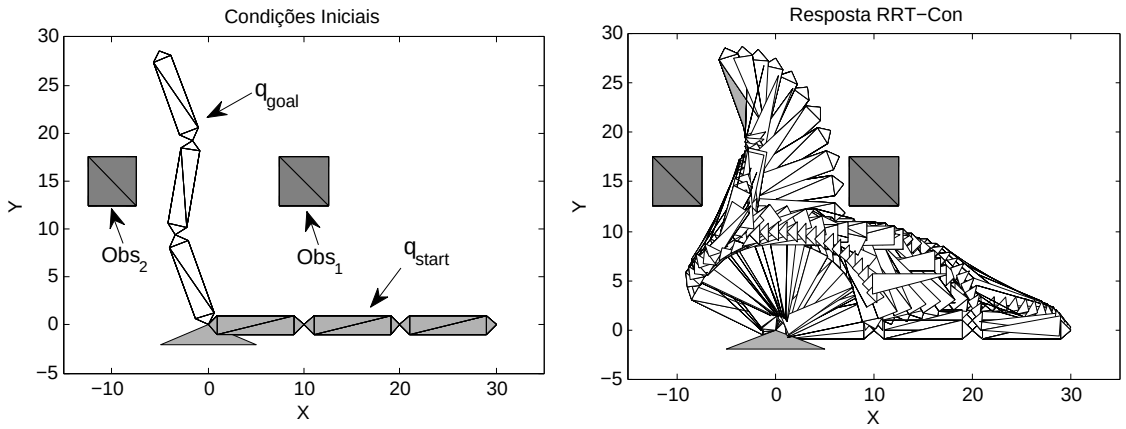


Figura 5.22: Resultados do DRRT-Con para um robô pontual num ambiente com seis obstáculos que se movimentam.

pode-se apreciar melhor quando o ambiente não sofre mudança. Por exemplo, na Fig. 5.21(a) apresenta-se em linhas cinza os galhos da exploração RRT-Connect, em preto a rota obtida com RRT-Connect e em verde a rota depois do pós-processamento, já na Fig. 5.21(b) é mostrada a rota curta em azul obtida com o DRRT-Con depois do planejamento de rotas dinâmicas.

Teste em espaço bidimensional com múltiplos obstáculos. Outro exemplo que mostra o desempenho do algoritmo DRRT-Con em ambientes dinâmicos é apresentado na Fig. 5.22 onde o espaço de busca contém seis obstáculos triangulares que se movimentam em direções aleatórias. Nesta figura é apresentado oito blocos intermediários, onde o bloco com número um coincide com a primeira etapa do algoritmo. A primeira rota livre de colisões é indicada em cinza e seu resultado do pós-processamento em verde. A partir dessa rota o DRRT-Con procede com o planejamento de rotas dinâmicas desde o bloco número dois até oito. Nes-



(a) (b)

Figura 5.23: Resultados do DRRT-Con para um robô puntual num ambiente com seis obstáculos que se movimentam.

ses blocos o caminho dos obstáculos é indicado com linhas cinza; mais uma vez, a rota dinâmica do robô é indicada com pontos azuis e seus estados estimados pelo pós-processamento com linhas verdes.

Pode-se observar que entre os blocos um e dois, a rota verde muda pelo pós-processamento melhorando a rota em direção ao alvo. Entre o bloco dois e três, é apreciável como o algoritmo recalcula a rota. Isto acontece por que os dois primeiros obstáculos da esquerda fecham o caminho que foi estabelecido no começo. Finalmente, do bloco quatro até oito, o robô se movimenta interativamente com as mudanças de posição dos obstáculos tendendo chegar com a rota mais curta ao alvo.

Teste com robô serial. Nos exemplos anteriores o robô é considerado como um ponto, portanto o espaço de configuração (ou busca) é coincidente com o espaço de trabalho. No caso de robôs seriais a dimensão do espaço de configuração $\mathcal{Q}$ é diferente à dimensão do espaço de trabalho $\mathcal{W} \in \{\mathbb{R}^2, \mathbb{R}^3\}$. A Fig. 5.23(a) apresenta um robô planar 3-GDL em presença de dois obstáculos Obs_1 e Obs_2. Conseqüentemente o espaço de busca pode ser representado como $\mathcal{Q} \in \mathbb{R}^3$, segundo a seção 2.4.2.

No caso estático, a solução do planejamento dos movimentos desde uma configuração inicial $\mathbf{q}_{start}$ até $\mathbf{q}_{goal}$ é resolvida aplicando o algoritmo RRT-Connect, ver a Fig. 5.23(b). Em um ambiente dinâmico, onde os dois obstáculos se movimentam no sentido do eixo x, o RRT-Connect não é aplicável, pois é possível que no percorrido dos movimentos do robô, seus elos colidiram com Obs_2. Neste caso dinâmico o DRRT-Con consegue obter uma resposta viável.

A Fig. 5.24 mostra um resumo de como o DRRT-Con é aplicado ao caso de movimento dos dois obstáculos da Fig. 5.23. No *step*=6 da Fig. 5.24 o manipulador

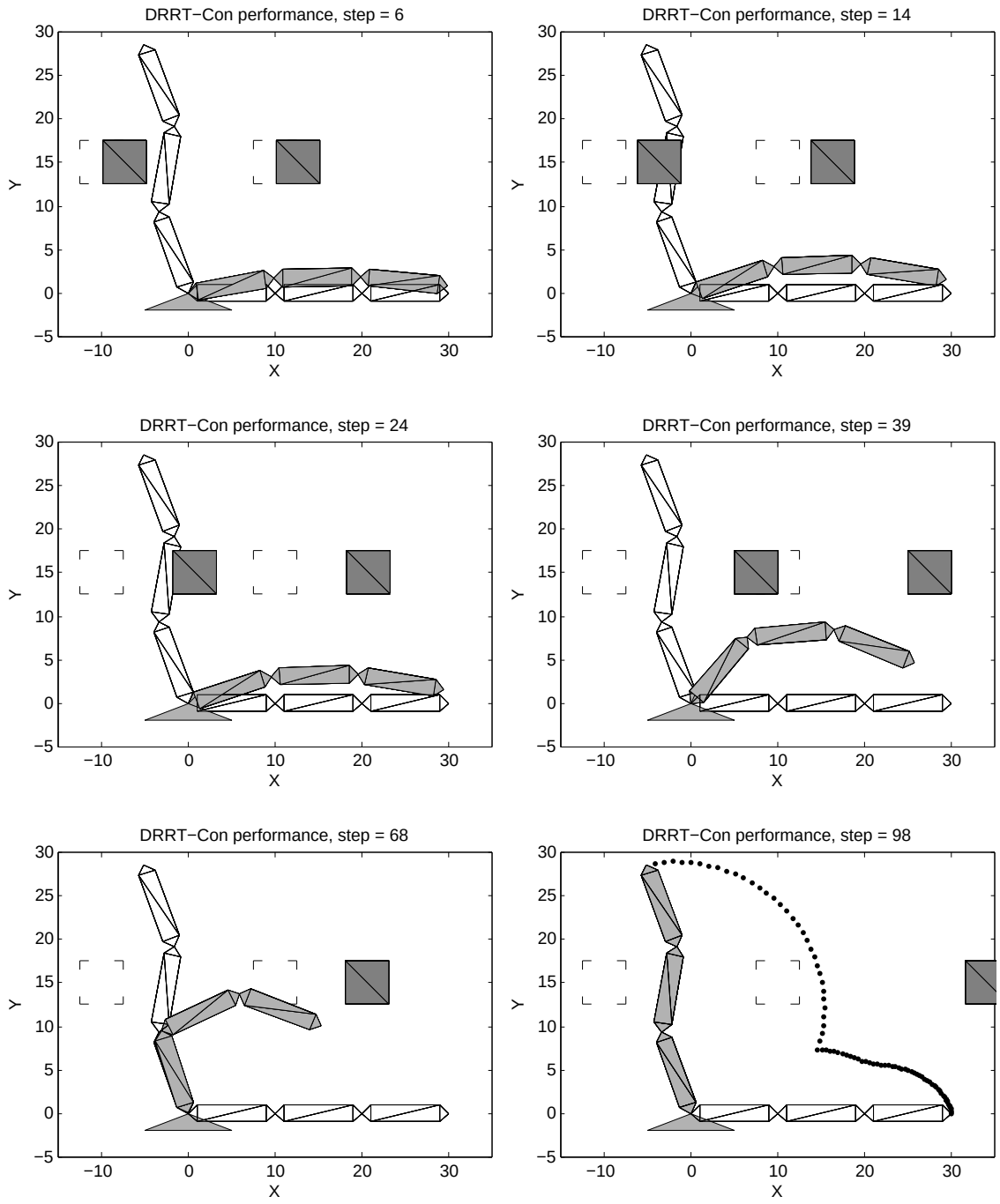


Figura 5.24: Resultados do DRRT-Con para um robô Serial.

planar se movimenta com a rota obtida pelo planejador interno RRT-Con, como se tratasse do caso estático. Já no *step*=14 o DRRT-Con realiza uma parada do robô aguardando até que exista uma rota viável desde a configuração atual até a $\mathbf{q}_{goal}$. Convém lembrar que como o Obs_2 está na situação de colisão com a configuração $\mathbf{q}_{goal}$, não existe uma solução direta da configuração atual até $\mathbf{q}_{goal}$. O algoritmo continua retendo o robô até que exista uma solução viável como no *step*=39. Logo no *step*=68 mostra que como o Obs_1 não está em sua posição inicial e em conseqüência disso, o robô pode se movimentar livremente por seu espaço de trabalho. Finalmente o *step*=86 mostra com pontos a rota percorrida pelo EF como solução do planejamento do movimento no ambiente dinâmico não previsível.

A partir das simulações desta seção pode-se obter algumas características especiais do planejamento dos movimentos com amostragem usando o DRRT-Con:

- O algoritmo recalcula o trajeto desde $\mathbf{q}_{next}$ até $\mathbf{q}_{goal}$ enquanto o robô se movimenta de $\mathbf{q}_{actual}$ até $\mathbf{q}_{next}$. Desta maneira é possível interceptar as futuras condições de colisão. O algoritmo tem uma natureza preventiva e não preditiva. O DRRT-Con não predisse as condições futuras dos obstáculos.

- O calculo do afastamento de $\mathbf{q}_{actual}$ até $\mathbf{q}_{next}$ é inerente ao problema estabelecido. Na presença de muitos obstáculos que se movimentam rapidamente, o robô conseqüentemente deve se movimentar igualmente rápido, portanto, a distância entre $\mathbf{q}_{actual}$ até $\mathbf{q}_{next}$ deve ser ajustada em proporção as condições geométricas do problema e as velocidades do sistema.

- O DRRT-Con é projetado para ambientes onde não é previsível o movimento dos obstáculos. Mas, para ambientes restringidos onde se pode calcular a direção ou velocidade dos obstáculos, o DRRT-Con pode ser adaptado para responder às mudanças preditivas, modificando o comportamento do robô quando se movimenta de $\mathbf{q}_{actual}$ até $\mathbf{q}_{next}$ e gerando um novo $\mathbf{q}_{next}$ de acordo com as predições obtidas do ambiente.

- As simulações com robôs seriais mostram um nível de complexidade diferente para o caso de ambientes imprevisíveis. Por exemplo, se algum obstáculo permanece ocupando parte do espaço reservado para a configuração final do manipulador (colisão no estado $\mathbf{q}_{goal}$), o algoritmo DRRT-con faz que o robô fique parado até que a configuração final esteja disponível. Se o manipulador possui multiplicidade da cinemática inversa para a configuração final, podem-se estabelecer novas sub-rotinas no algoritmo de planejamento de rotas que formulem novas soluções. Como foi discutido nas seções anteriores, o cálculo da cinemática inversa estendida não é uma questão trivial e as soluções obtidas dependem das restrições reais do sistema robótico.

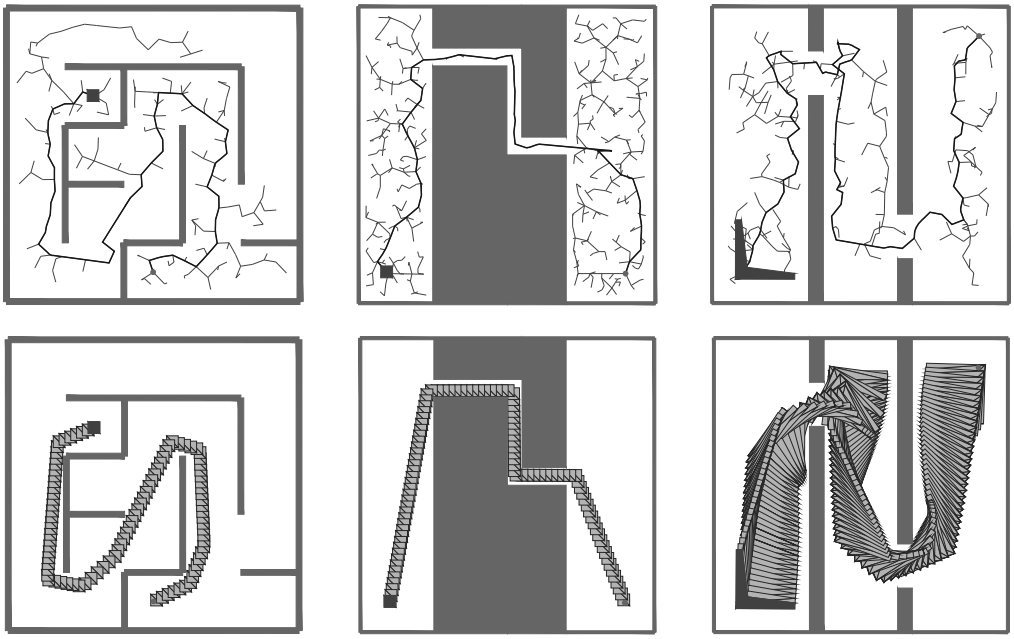


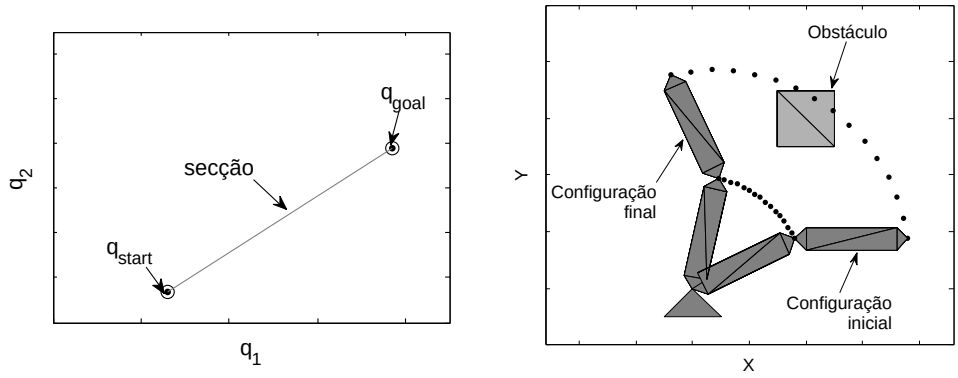
Figura 5.25: Resultados usando RRT-Con, os blocos inferiores são as rotas curtas desejadas a partir da solução obtida nos blocos superiores. Imagem modificada de [111].

5.3 Pós-processamento Para Obtenção de Rotas Curtas

Os *Algoritmos Baseados em Amostragem*, como o PRM e RRT, geram rotas livres de colisões, sem embargo essas rotas apresentam zigue-zagues que produz movimentos desnecessários. A Fig. 5.25 ilustra três exemplos de rotas obtidas com RRT nos blocos superiores e suas correspondentes rotas curtas desejadas mostradas nos blocos inferiores.

Para alcançar as rotas curtas desejadas é preciso aplicar algoritmos de pós-processamento, em conseqüência, o objetivo desta terceira parte do capítulo é apresentar diversas táticas de pós-processamento. Sua correta interpretação e escolha ajuda à aplicação bem sucedida das estratégias de *Planejamento dos Movimentos* na robótica.

Antes de apresentar as estratégias de pós-processamento convém lembrar que uma rota é projetada no espaço de configuração $\mathcal{Q}$. Ela é composta por pontos e seções. Por exemplo, para um robô planar 2-GDL da Fig. 5.26, a Fig. 5.26(a) mostra uma rota com dois pontos e uma seção no $\mathcal{Q}$, mas os movimentos do robô no $\mathcal{W}$ na Fig. 5.26(b) indicam que existe colisão com um obstáculo, portanto a rota apresentada na Fig. 5.26(a) não é válida. Em outras palavras, os pontos $\mathbf{q}_{start}$ e $\mathbf{q}_{goal}$ são válidos, pois as configurações inicial e final no $\mathcal{W}$ não têm colisões, no entanto, a seção entre os dois pontos no $\mathcal{Q}$ não é válida.



(a) $\mathcal{Q}$ (b) $\mathcal{W}$

Figura 5.26: Representação de rota não válida para um robô planar 2-GDL.(a) No espaço de configuração. (b) No espaço de trabalho.

O conceito de uma rota válida no $\mathcal{Q}$ é facilmente extrapolada a espaços de configuração multidimensionais, como acontece para um manipulador de 6-GDL onde $\mathcal{Q} \in \mathbb{R}^6$. Para simplificar a descrição das estratégias de pós-processamento a seguir é usado o caso de um robô ponto, onde os dois espaços $\mathcal{W}$ e $\mathcal{Q}$ são bidimensionais e coincidentes.

5.3.1 Pós-processamento por Corte Seqüencial

A primeira idéia do pós-processamento sempre é a redução da rota gerada por um planejador probabilístico removendo as partes desnecessárias. A estratégia de corte seqüencial realiza uma exploração da rota obtida para cortar aquelas seções e pontos da rota que ficam afastadas da configuração de destino.

A Fig. 5.27 ilustra o principio de funcionamento do pós-processamento por corte seqüencial. Em (a) se apresenta a condição inicial de exemplo num ambiente bidimensional, onde estão presentes dois obstáculos poligonais em cinza $\mathcal{Q}_{obs1}$ e $\mathcal{Q}_{obs2}$. Em (b) é representada a solução obtida por qualquer estratégia baseada em amostragem, onde essa rota está composta por sete seções e oito pontos incluindo $\mathbf{q}_{start}$ e $\mathbf{q}_{goal}$.

Em cada iteração do algoritmo de pós-processamento por corte seqüencial é usado um ponto da rota como um pivô de comprovação. Em (c) o algoritmo inicia com o pivô $i = 1$, onde se formulam cortes seqüenciais do pivô até os pontos seguintes, começando desde o último até o ponto $i + 1$, cada corte é uma seção que se deve validar. Neste caso, o corte 1-3 é o primeiro que está livre de colisão, então, a seção 1-2-3 é substituída pelo corte 1-3, essa condição é ilustrada em (d) da Fig. 5.27.

Dado que o último corte foi realizado no ponto número 3, esse ponto é o novo pivô $i = 3$, condição (e) da Fig. 5.27. O processo de corte seqüencial é repetido até que o último ponto de corte sem colisão seja o ponto final 8.

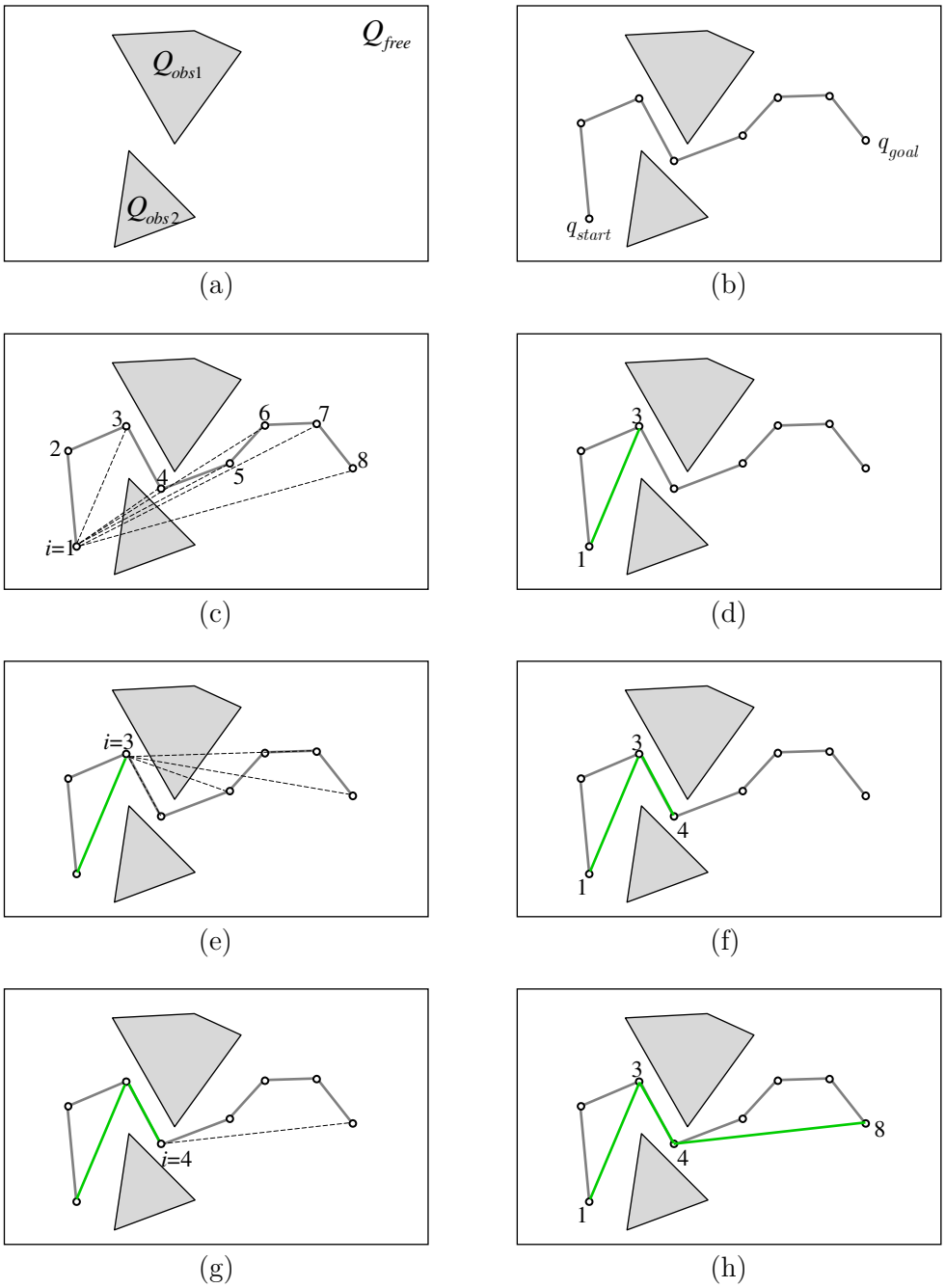


Figura 5.27: Representação da operação de pós-processamento sequencial. (a) condições iniciais. (b) rota obtida com RRT-Con. (h) a rota de pós-processamento final 1-3-4-8.

Ao final, a rota de pós-processamento é composta pelos cortes seqüenciais livres de colisão, como é apresentado na condição (h) pelos pontos 1-4-8.

A rota de pós-processamento pode ser aprimorada dividindo as secções originais com pontos intermediários, por conseqüência, cada ponto pivô duplicará seu número de cortes, resultando em um incremento do tempo de computação. Para obter uma rota de pós-processamento quase-ótima, o numero de pontos intermediários da rota original aumenta até o infinito, portanto o número de cortes aumenta da mesma maneira, fazendo que o tempo de cálculo tenda ao infinito.

O pós-processamento também pode ser realizado na ordem inversa, com o primeiro pivô $i = 8$ e com cortes seqüenciais também na ordem inversa, no entanto, não existe evidencia que estas variações melhorem a qualidade da rota ou o tempo de processamento.

Outra característica deste pós-processamento é qualidade da rota obtida, pois ela no consegue chegar a ser a rota ótima ou rota mais curta possível, pois não é usado um critério de otimalidade no algoritmo. No entanto, as soluções obtidas são rápidas em comparação com outras estratégias, tal como é mostrado nas próximas secções.

5.3.2 Pós-processamento por Corte Aleatório

Nesta estratégia o número de pontos não é um fator predominante na obtenção da resposta. Os cortes são feitos escolhendo dos pontos aleatórios sobre toda a rota.

Por exemplo, na Fig. 5.28 o algoritmo começa com a condição (a) da rota obtida por um planejador probabilístico, logo na primeira iteração em (b) são selecionados dos pontos de corte s_1 e s_2. A seção entre os dois pontos deve ser validada. Se essa seção não é valida, ela é esquecida e o algoritmo continua com a próxima iteração.

Na condição (c), o algoritmo depois de algumas iterações encontra dois pontos de corte s_1 e s_2 que produzem uma seção livre de colisões. Neste caso as secções originais da rota entre s_1 e s_2 são substituídas pelo corte, como é ilustrado em (d). Portanto, as próximas iterações usarão essa seção para estabelecer novos pontos de corte, ver condição (e). O pós-processamento por cortes aleatórios continua sua busca por uma nova seção livre de colisões, como em (f), para reduzir a rota de maneira iterativa. A parada do algoritmo é realizada por número máximo de iterações ou por um limite de tempo.

Com a estratégia por cortes aleatórios é possível alcançar uma rota ótima no caso do tempo infinito, pois novas secções são estabelecidas a partir de uma condição anterior. A idéia de cortes aleatórios sugere uma melhoria constante, no entanto, o tempo requerido para obter uma rota de boa qualidade depende da forma e do comprimento da rota. Em geral cortes aleatórios longos precisam de tempo de

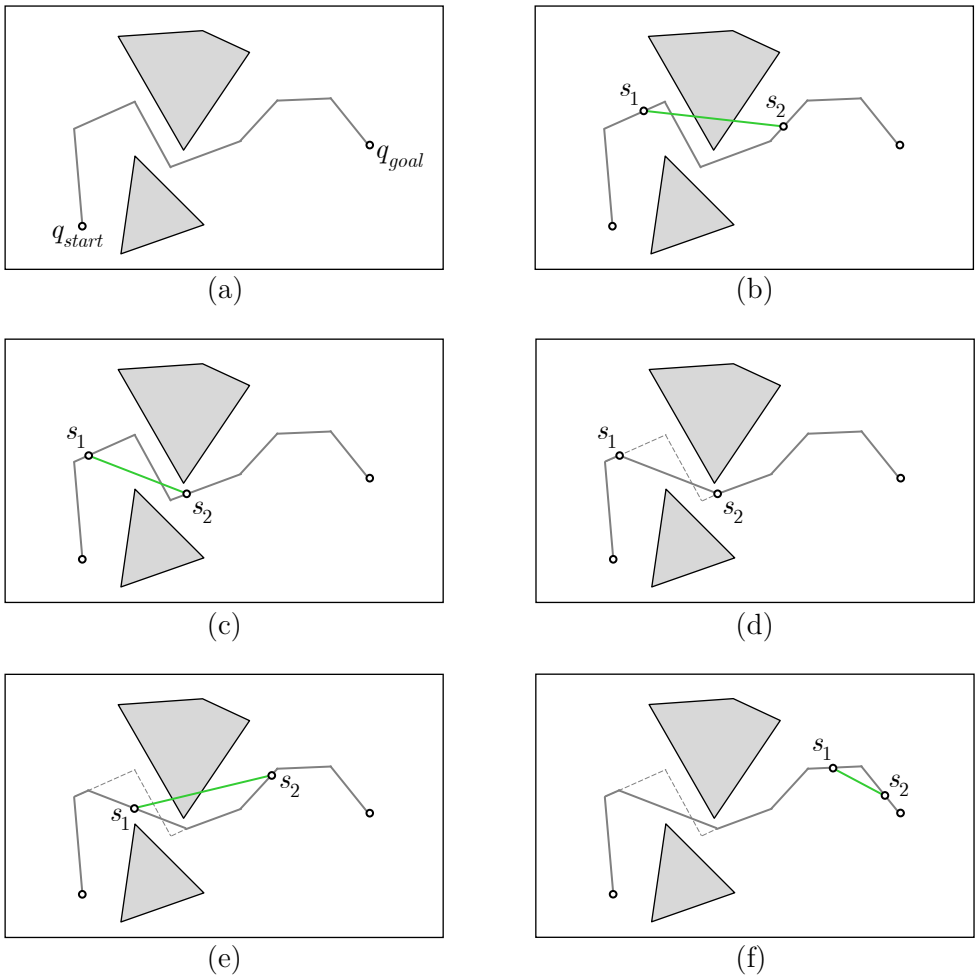


Figura 5.28: Representação da operação de pós-processamento por corte aleatório. (a) condição inicial. (b) corte aleatório não válido. (c) corte válido. (d) reposição da rota.

computo maior no módulo de detecção de colisão operando no $\mathcal{W}$.

5.3.3 Pós-processamento por Retração

O algoritmo tem como base um *Processo de Busca por Divisão* ou PBD. O PBD é uma busca melhorada do pós-processamento por corte seqüencial. A Fig. 5.29 ilustra o principio do funcionamento do PBD, onde uma rota curva está restrita entre o intervalo s_1 e s_2, o PBD calcula o ponto médio m. Parte superior da Fig. 5.29. Se o corte entre o pivô i e o ponto m é válido, é gerado um novo intervalo de busca onde o novo $s_1 = m$, condição mostrada na parte inferior-direita da Fig. 5.29. No caso contrario, onde o corte entre o pivô i e o ponto m não é válido, o novo intervalo está definido por $s_2 = m$. O processo PBD se repete até que s_1 e s_2 sejam consecutivos.

O algoritmo de pós-processamento por retração foi desenvolvido nesta tese, e ele usa três fases de maneira consecutiva repetindo o PBD em cada fase para obter um ponto em particular da seguinte maneira:

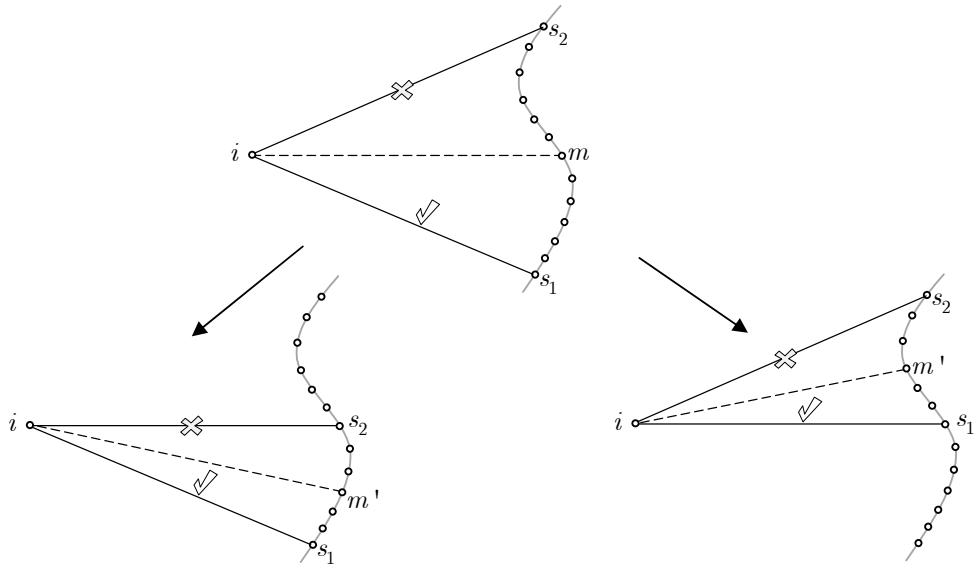


Figura 5.29: Representação do Processo de Busca por Divisão ou PBD.

Fase 1 (obter j): calcular o ponto j mais próximo ao pivô i, que produz um corte válido.

Fase 2 (obter k): afinar a busca para obter um ponto k que está mais longe ao ponto j, que produz um corte válido.

Fase 3 (obter l). Retração ao pivô para encontrar o ponto l, que produz um corte válido.

A Fig. 5.30 mostra o funcionamento da Fase 1. No inicio, o intervalo de busca é $[s_1, s_2]$, correspondente ao intervalo $(\mathbf{q}_{start}, \mathbf{q}_{goal}]$. Após aplicar repetidamente o PBD e esgotar os pontos da rota original e definido o ponto j, a seção da rota original i-j é substituída pelo corte i-j encontrado pelo PBD, como mostra a condição (d) da mesma figura.

A Fase 2 para o mesmo ambiente é representado na Fig. 5.31. O objetivo desta fase é encontrar um ponto k que produz um corte válido o mais afastado de j, portanto é aplicado iterativamente o PBD selecionando como intervalo inicial $[s_1, s_2] = [j, j+1]$ e realizando uma discretização da seção original entre j e $j+1$, ver condição (a) da Fig. 5.31. Uma vez encontrado o ponto k, a seção original da rota i-k é substituída pelo corte obtido com PBD. Observa-se que depois de usar de forma iterativa o PBD, o ponto final s_2 pode ser diferente do s_2 inicial, como é mostrado pela condição (b) da mesma figura.

A Fig. 5.32 ilustra a Fase 3, onde ocorre a retração. O objetivo é descobrir o ponto l mais perto de i usando iterativamente o PBD com intervalo inicial $[s_1, s_2] = [i, k]$. O pivô do PBD muda ao último ponto s_2 da Fase 2 (Ponto i' na Fig. 5.32). Uma vez encontrado o ponto l, a seção original da rota i-k-i' é substituída pelo seção i-l-i'.

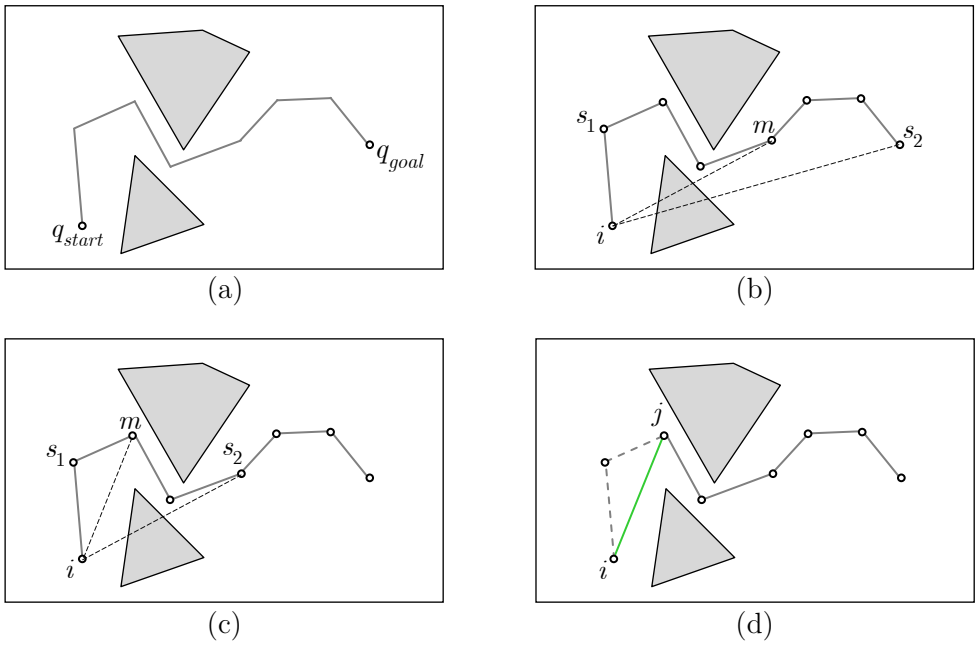


Figura 5.30: Representação da Fase 1 do pós-processamento por retração.

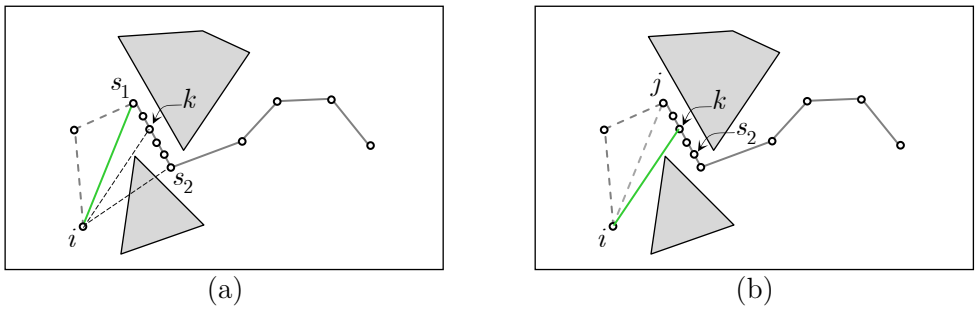


Figura 5.31: Representação da Fase 2 do pós-processamento por retração.

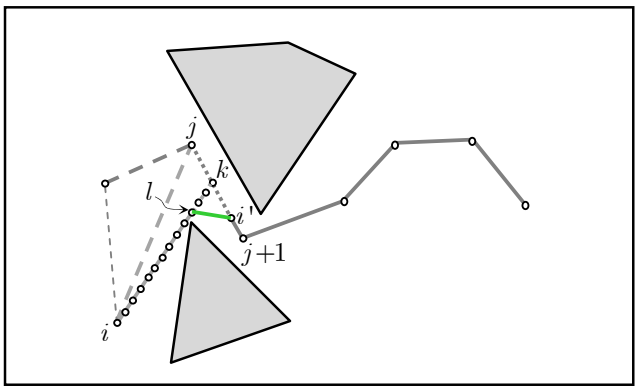


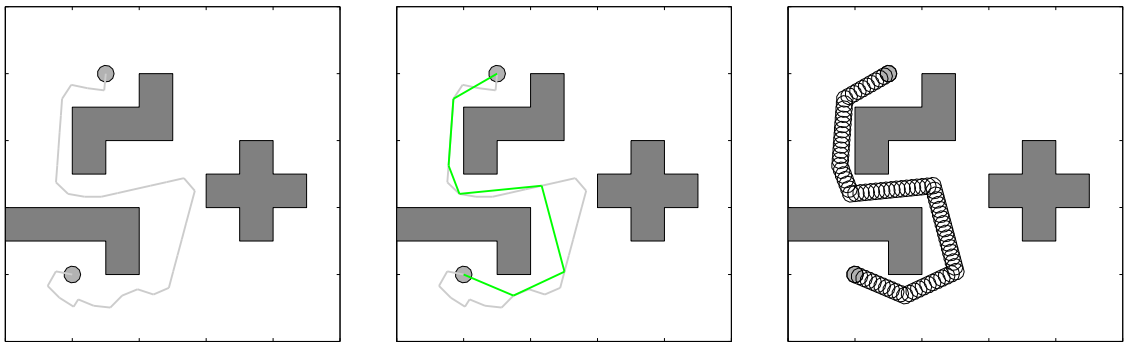
Figura 5.32: Representação da Fase 3 do pós-processamento por retração.

As Figuras 5.30, 5.31 e 5.32 representam a primeira iteração do algoritmo de pós-processamento por retração. Para a próxima iteração o pivô i muda ao ponto l, e as três fases se repetem até chegar ao final da rota original.

O algoritmo de Pós-processamento por Retração é original desta tese. Uma característica desta estratégia é a obtenção de rotas curtas próximas à rota mais curta possível partindo de uma rota obtida com planejadores probabilísticos. A qualidade da rota depende da discretização das seções usadas nas fases 2 e 3. A próxima seção apresenta uma comparação entre os três métodos de pós-processamento usando ambientes simulados em MATLAB.

5.3.4 Comparação Entre os Métodos

A Fig 5.33(a) mostra um ambiente de teste com um robô circular em presença de obstáculos poligonais, também é apresentada a rota obtida pelo planejador probabilístico RRT-Connect. Essa figura é a primeira etapa comum para todos os métodos de pós-processamento testados.



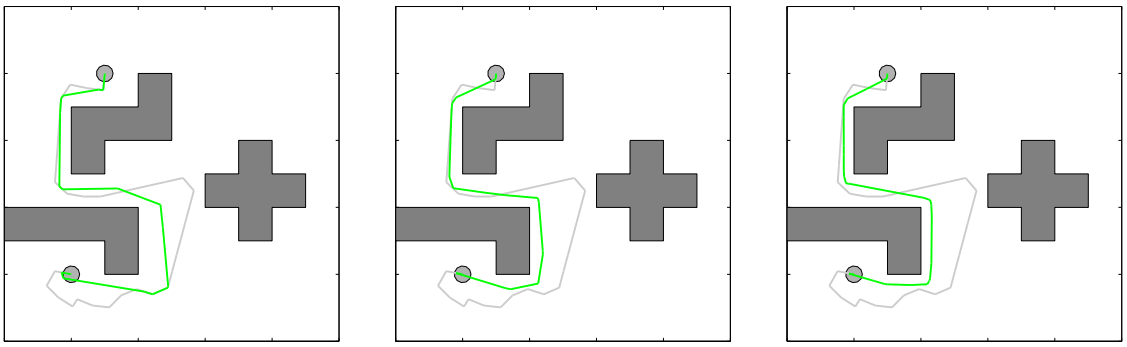
(a) (b) (c)

Figura 5.33: Pós-processamento.(a) Rota original RRT-Connect. (b) Em linha verda, a rota obtida com pós-processamento por corte seqüencial. (c) movimentos intermediários.

Após da obtenção da rota livre de colisões, na Fig 5.33(b) é observada em verde o resultado do pós-processamento por corte seqüencial. Percebe-se que a rota não é ótima, no entanto, ela permite menor número de movimentos que a rota inicial. Do mesmo modo, na Fig 5.33(c) pode se observar uma série de movimentos intermediários do robô para esse mesmo pós-processamento.

A qualidade da rota original é dependente do parâmetro de crescimento ϵ da árvore de exploração (Ver Fig 2.16), portanto o tempo de processamento também muda de acordo ao ϵ usado. Para o caso particular da Fig 5.33(a), o tempo de computação do pós-processamento por corte seqüencial é de 0,94seg.

Em relação ao pós-processamento por cortes aleatórios, a qualidade da rota é dependente do numero de iterações realizadas no pós-processamento. Na Fig 5.34

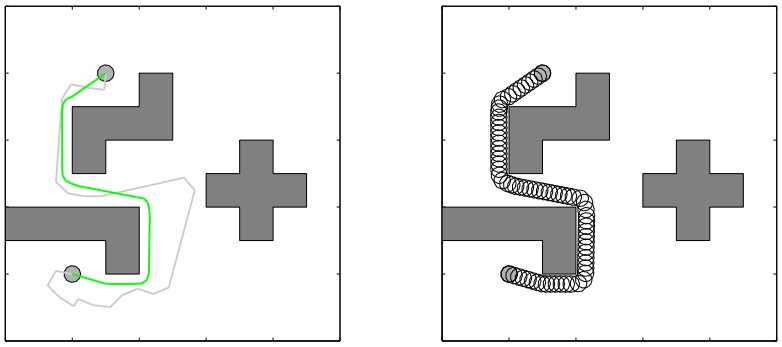


(a) $n = 100$, $t = 1,07seg$ (b) $n = 300$, $t = 2,17seg$ (c) $n = 800$, $t = 3,97seg$

Figura 5.34: Resultados de pós-processamento por corte aleatório.

mostra três resultados aplicando o método. Na condição (a) após de $n = 100$ iterações, a rota obtida está longe de ser ótima, além disso o tempo de cálculo é similar ao obtido no pós-processamento por corte seqüencial. Já na condição (b) e (c) é óbvio um avanço na obtenção de uma rota o mais curta possível, entretanto, o tempo de cálculo aumenta até quase o duplo.

O terceiro método testado é o pós-processamento por retração, os resultados obtidos são apresentados na Fig. 5.35. A condição (a) ilustra em linha verde a rota de pós-processamento, a qual está perto da rota mais curta possível, em (b) pode ser observados os movimentos intermediários do robô na execução da rota obtida. Como foi expresso anteriormente, o tempo de computo e a qualidade da rota de pós-processamento depende da discretização da rota original em pontos e seções. Para o resultado obtido na Fig. 5.35 o algoritmo de pós-processamento por retração na Fase 2 utiliza o mesmo comprimento ϵ usado na obtenção da rota original RRT-Connect. Para a Fase 3, foi empregado um fator $\epsilon/5$.



(a) (b)

Figura 5.35: Resultados de pós-processamento por retração.

O tempo de pós-processamento por retração foi de 2,14seg. Esse tempo é maior que o obtido com o método por corte seqüencial, mas, o método por retração emprega menor tempo que o pós-processamento por cortes aleatórios para obter rotas com

características similares.

Outra distinção entre os dois últimos métodos é a capacidade do pós-processamento por retração de fornecer sempre o mesmo tempo de cálculo, diferentemente do método de cortes aleatórios que depende de um gerador de números randômicos para obter os pontos de corte. Conseqüentemente, sendo que o planejador probabilístico também depende de gerador de números randômicos, o tempo total de cálculo sofre muitas variações para solucionar um mesmo problema, pois deve-se somar o tempo de obtenção de uma rota livre de colisões e o tempo de pós-processamento por cortes aleatórios.

5.4 Conclusões Referentes ao Estudo De Caso

O movimento autônomo de braços robóticos em ambientes não-estruturados envolve diversos desafios, por exemplo: como configurar o manipulador para uma desejada posição do efetuador final quando existem obstáculos no ambiente real; outra das preocupações principais é determinar se é possível movimentar os elos do manipulador sem produzir colisões em um ambiente dinâmico. O capítulo concentra-se no estudo destes dos questionamentos.

Com relação ao primeiro item de estudo, a seção 5.1 mostra como é possível resolver o caso da cinemática inversa estendida para robôs seriais usando o conceito de espaço de configuração e os métodos baseados em amostragem. Apesar de que as estratégias entregam resultados satisfatórios, o tempo de computação pode ser alto. A busca de uma solução da cinemática inversa em presença de obstáculos está fortemente limitada pelo número de verificações de detecção de colisões.

Embora a posição atual do EF e a desejada estejam próximas, nem sempre se tem uma solução rápida. Por exemplo, ao tentar tomar um copo que fica acima da mesa quando o EF do manipulador está diretamente baixo dela; neste caso a rota indica que a seqüência de movimentos dos elos deve-se adaptar, se deslocando ao redor à geometria da mesa e dos objetos restantes da cena.

Dado o exposto, atualmente uma aplicação *on-line* destas estratégias para solucionar a cinemática inversa estendida não é pratica usando MATLAB e V-Collide. Mas, existem outras bibliotecas para detecção de colisões, linguagens de programação e estratégias de processamento em paralelo que permitiriam melhorar os tempos de computação e viabilizar aplicações reais.

A outra preocupação constante é a variação do ambiente, quer dizer, o problema de *Path Planning* estendido; onde os obstáculos se movimentam ou variam de forma. A seção 5.2 ilustra como lidar com ambientes dinâmicos de duas maneiras. Uma solução é usar uma iteração entre os métodos planejadores de rotas e os sistemas de cálculo da dinâmica do manipulador, realizando ajustes iterativamente conforme ao

estado real do ambiente (Ver Fig. 5.12). Uma segunda solução geral é proposta neste trabalho, usando o algoritmo RRT-Connect de maneira iterativa (DRRT-Con), pois quando existe uma mudança do ambiente o planejador RRT-Connect estabelece uma nova rota livre de colisões até atingir o alvo.

Esta técnica de geração de rotas em ambientes dinâmicos é apropriada para quando não são previsíveis os estados futuros dos obstáculos. O algoritmo DRRT-Con estabelece que o próximo passo dado pelo robô esteja livre de colisões. Isto implica que o sucesso do DRRT-Con depende da velocidade de amostragem do deslocamento do robô.

Uma analise para robôs seriais mostra que apesar dos bons resultados obtidos com técnicas preventivas tipo DRRT-Con, os elos dos manipuladores se movimentam com diferentes velocidades absolutas, em conseqüência um planejamento no $\mathcal{Q}$ desconsiderando a dinâmica envolvida dos elos no $\mathcal{W}$ produz paradas instantâneas o qual às vezes é impossível de realizar com robôs reais sem danificar os atuadores. Consequentemente, a solução ao problema dinâmico é a mesma, usar baixas velocidades de operação, tal como é utilizado nas estratégias reativas baseadas no gradiente.

Por outro lado, para obter rotas de boa qualidade com os métodos de Planejamento Probabilístico é preciso usar um algoritmo de pós-processamento. A seção 5.3 apresenta de maneira formal duas metodologias existentes de pós-processamento, e que até o momento de desenvolvimento desta tese, foram negligenciados na literatura robótica. Além disso, na mesma seção é apresentado um novo algoritmo de pós-processamento que consegue obter rotas curtas próximas as rotas ótimas, como as obtidas usando o algoritmo RRT* proposto por Karaman e Frazzoli em [180, 181].

Capítulo 6

Formulação de Ferramenta Para Auxilio à Teleoperação

Sendo o objetivo deste trabalho apresentar estratégias que facilitem a operação de braços robóticos em ambientes não-estruturados, nos dois últimos capítulos foram estudadas as tecnologias de Realidade Aumentada na robótica e a análise de técnicas de geração de rotas livres de colisões em ambientes não tradicionais das células de manufatura. O presente capítulo ilustra como integrar estas dois tecnologias para o estudo de caso de manipuladores robóticos submarinos operados de maneira remota, estabelecendo uma nova arquitetura de controle compartilhado.

O capítulo está constituído por três partes: a primeira introduz conceitos de *Human-In-The-Loop* (HITL) para o controle compartilhado de braços robóticos submarinos; a segunda mostra uma aplicação em software baseada em HITL e desenvolvida tendo em vista as dificuldades da teleoperação expostas no Capítulo 2, às condições de singularidade do Manipulador TITAN-4 analisadas no Capítulo 3, à perda de sensação de profundidade estudada no Capítulo 4, e ao contexto do planejamento autônomo de movimentos de robôs seriais estabelecido no Capítulo 5; já a última e terceira parte deste Capítulo apresenta uma nova estratégia de geração de rotas que permite aos pilotos humanos acompanhar os movimentos gerados automaticamente, os quais são baseados nos conceitos introduzidos no Capítulo 5. (Ver Fig. 6.1).

6.1 Contexto do Controle de Manipuladores

A tendência da autonomia das arquiteturas telerobóticas é obter um sistema supervisionado (designado como *Supervisory control* na Fig. 1.3). Idealmente, o planejamento automático de trajetórias em robôs manipuladores pode ser resumido na Fig. 6.2. Onde a entrada do *Planejador de Trajetórias* é a tarefa desejada que pode

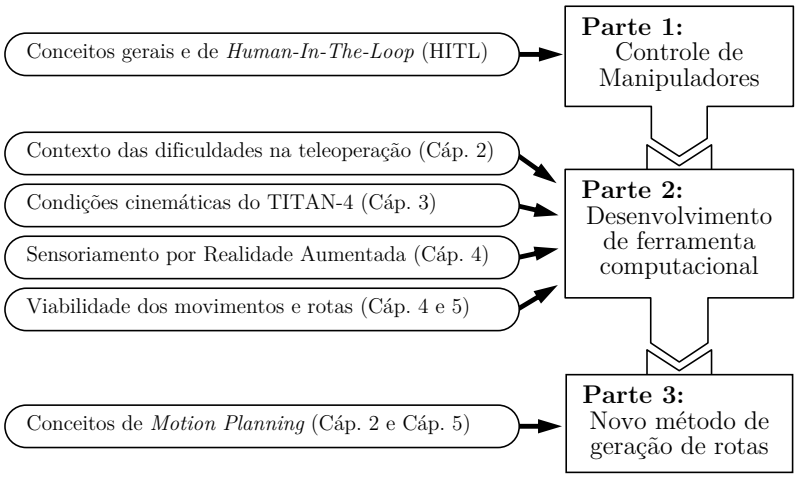


Figura 6.1: Esquema para entendimento do capítulo 6. Na esquerda são indicados os requisitos e na direita a descrição das partes do capítulo.

ser indicada em palavras (operação de alto nível), por exemplo: "tirar um copo da mesa"; e a saída, são as trajetórias: perfis de posição, velocidade e aceleração nas juntas (operações de baixo nível).

Um *Planejador ideal de trajetórias* deve considerar as condições cinemáticas, dinâmicas e do ambiente antes de agir no manipulador. As considerações cinemáticas referem-se à seqüência de movimentos do manipulador em seu espaço de operação, sem incluir velocidades e acelerações, isto é chamado de *Planejamento de Rotas*. Nas considerações dinâmicas as inércias afetam as forças exercidas sobre os elos em virtude as velocidades dos mesmos. E as considerações do ambiente incluem a presença de obstáculos e a maneira como o ambiente atua no braço robótico, por exemplo, as forças exercidas por correntezas.

Como foi apresentado no Capítulo 2, a conjunção de problemas cinemáticos e dinâmicos em ambientes não-estruturados não permite obter uma formulação matemática de um *Planejador ideal de trajetórias*. Uma solução viável, tal como foi analisado na seção 5.2, é utilizar *Planejadores de Rotas Livres de Colisões*, que permitem a abordagem de condições cinemáticas de maneira conjunta com o estado

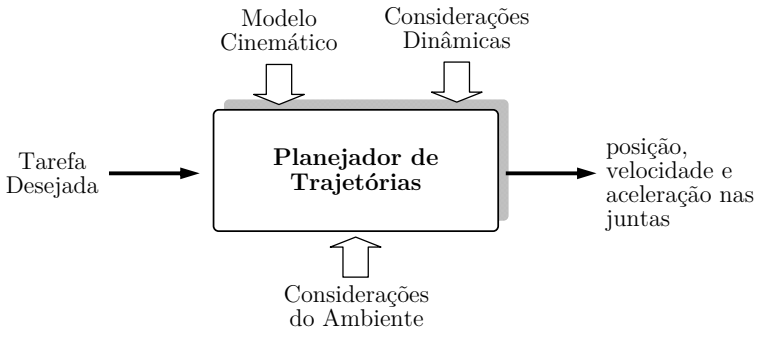


Figura 6.2: Considerações para um planejador de trajetórias.

dos obstáculos no ambiente; logo, deve-se conferir com a modelagem de sistemas dinâmicos se a *rota* pode ser usada para gerar uma *trajetória real*.

6.1.1 Controle Automático Ideal

A Fig. 6.3 mostra o diagrama de blocos do *sistema de controle* ideal de um manipulador robótico. Uma vez que a tarefa desejada é analisada, interpretada e traduzida em uma serie de posições, velocidades e acelerações de junta pelo bloco *Planejador de Trajetórias*, o bloco de *Sistema de Controle* é encarregado de calcular os valores de torque de controle sobre o *Manipulador* real. No ambiente é possível medir as posições e velocidades reais das juntas, daí, o sistema *sensor* tem a função de encaminhar os valores reais ao bloco *Sistema de Controle* para realizar os ajustes minimizando um sinal de erro.

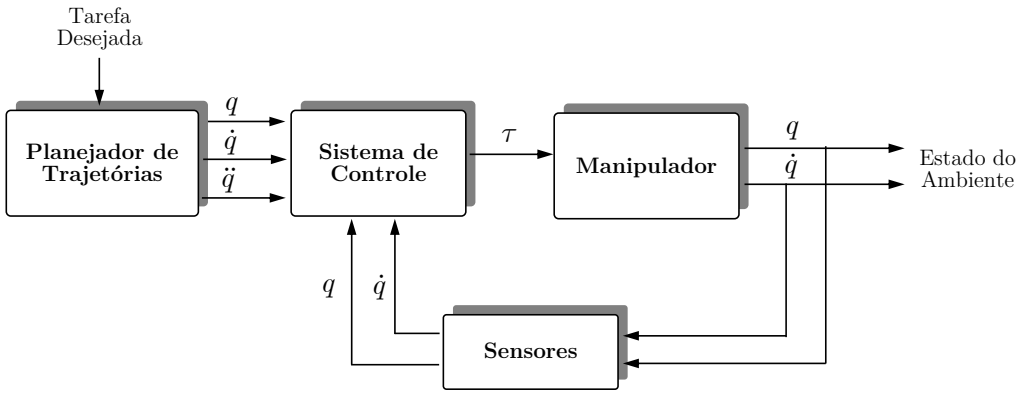


Figura 6.3: Diagrama de controle automático ideal de manipuladores.

Na Fig. 6.3 o sinal de controle representa os valores das variáveis nas juntas, esse tipo de diagrama é chamado de *Controle no Espaço das Juntas*. Também existe um diagrama similar que usa como sinal de controle a posição e orientação do efetuador final, esse esquema é chamado de *Controle no Espaço Operacional*. A Fig. 6.3 é apresentada como Controle Automático Ideal para facilitar a compressão no tema de geração de trajetórias.

6.1.2 Controle Remoto Real

Os movimentos dos manipuladores industriais tradicionais são programados de maneira *off-line*. Por exemplo, usando simuladores que replicam o ambiente e condições nas células de manufatura. Por outro lado os robôs usados em ambientes não-estruturados são controlados de maneira remota *on-line* por pilotos humanos com habilidades cognitivas na resolução de problemas complexos em frente de telas ou projeções das imagens do ambiente real onde o robô opera. Uma dificuldade para robôs que operam fora de ambientes controlados é conseguir realizar movimentos

totalmente autônomos, devido em maior parte pelas tarefas complexas apresentadas nos ambientes dinâmicos e que exigem processamento de dados o mais ágil possível.

O esquema que ilustra de modo geral como é controlado um manipulador submarino real é apresentado na Fig. 6.4. O termo técnico teleoperação refere-se quando um piloto humano controla os movimentos de maneira remota do robô, onde os sinais são enviados desde uma estação de controle da embarcação na superfície a uma profundidade determinada de trabalho do manipulador. O ser *Humano* conhece a tarefa desejada e transmite suas ordens através de uma *Interface ao robô*, e um *Sistema de controle* permite que o sistema *ROV/Manipulador* se movimente. O sistema real *ROV/Manipulador* possui *Sensores* básicos nos atuadores do robô que impedem avarias e danos nos elos do manipulador, por exemplo, interruptores de fim de curso que estabelecem o ângulo máximo e mínimo de avanço das juntas.

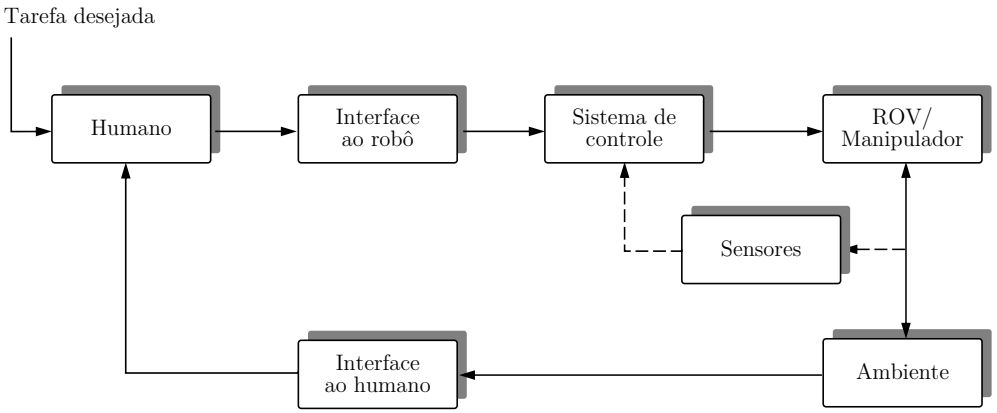


Figura 6.4: Diagrama de controle para o caso de um manipulador submarino.

O ambiente é um fator relevante na teleoperação, por exemplo, sendo necessário estabelecer o estado dos obstáculos e alvos de manipulação a um determinado referencial. A fonte de informação do estado do ambiente é em maior parte subministrada pelas imagens transmitidas ao piloto pelas câmeras a bordo do *ROV/Manipulador*, e transmitidas por uma *Interface ao humano* (Fig. 6.4). O piloto é quem atualmente realiza a operação de julgamento de posição e rotação dos objetos da cena real, dessa maneira todas as decisões são executadas no bloco *Humano*.

6.1.3 Abordagem de Controle por HITL

Nos sistemas que precisam da colaboração entre homem e robô, o problema natural é equilibrar o poder entre ambas as partes. Os sistemas "*Human-In-The-Loop*" ou HITL, têm o potencial para executar tarefas complexas em ambientes não estruturados, compartilhando as habilidades cognitivas humanas com as ferramentas e comportamentos autônomos das máquinas.

Os sistemas HITL não são novos e vários trabalhos têm sido publicados, desde a identificação de causas potenciais de falhas humanas [182, 183], onde a título de exemplo podemos citar: o uso de cadeiras de rodas robóticas [184–186], a simulação de sistemas multicorpos e controle de múltiplos robôs [187, 188] até os estudos das vantagens do uso em manipuladores em [189–191].

A Fig. 6.5 ilustra um diagrama de controle por HITL para o estudo de caso. Percebe-se que de maneira semelhante ao tradicional controle remoto, os sinais de controle gerados no bloco *Humano* são interpretados na *Interface ao robô* e transmitidas ao *Sistema de Controle*; que atua sobre o *ROV/Manipulador* real. Quando o manipulador se movimenta, ou quando muda o *Ambiente*, um *Sistema de sensoriamento* deve encaminhar informações do estado dos objetos reais ao *Sistema de controle* e à *Interface ao Humano*.

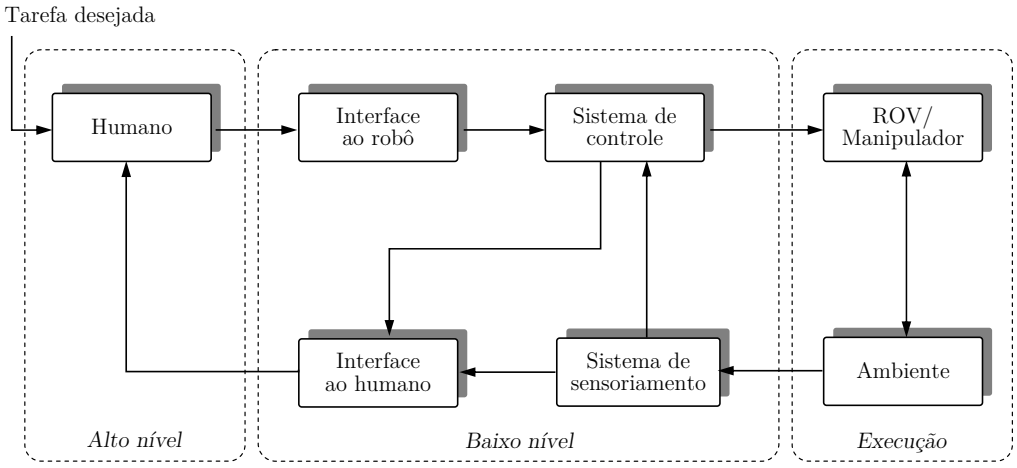


Figura 6.5: Diagrama de controle em HITL para caso de manipulador submarino.

As informações fornecidas pelo *Sistema de sensoriamento* realimentam o *Sistema de controle*, que efetua decisões de *baixo nível*, que podem ser de dois tipos: a) *Decisões Preventivas*: são decisões baseadas em cálculos cinemáticos e impossibilitam o movimento do robô, por exemplo: não permitir auto-colisões ou movimentos dos elos que gerem condições adversas à tarefa desejada; b) *Decisões Preditivas*: mostram ao humano os movimentos futuros ou condições do cenário atual de outro ponto de vista. Por exemplo: apresentar dados de distâncias ou sólidos virtuais usando a interface ao humano (Ver seção 4.4.1).

Todas as informações subministradas pela *Interface ao humano* contribuem para tomada de decisões de *Alto nível* do piloto, por exemplo, a melhor posição e orientação do efetuador final em uma operação de agarre. No entanto, algumas operações não poderão ser executadas no *Ambiente*, isto de acordo com os cálculos do *Sistema de controle*, essas restrições são apresentadas ao piloto diretamente pela *Interface ao humano*. Na próxima parte deste Capítulo é apresentada uma ferramenta computacional de ajuda na teleoperação baseada nesta estratégia HITL.

6.2 Desenvolvimento da Ferramenta Computacional

Nesta segunda parte do capítulo serão expostos diversos aspectos do aplicativo desenvolvido para auxilio à teleoperação de braços robóticos em ambientes não-estruturados. Estes aspectos contemplam: o esquema HITL; explicação do ambiente do aplicativo em janelas; como se estabelece a teoria robótica e AR em C++; as interações entre os linguajes C++ e MATLAB; parte dos conceitos de geometria computacional que ajudam nas decisões automáticas; exemplos de funcionamento do aplicativo; e finalmente são proporcionadas as considerações no uso *on-line*.

6.2.1 Estratégia HITL no Aplicativo

A Figura 6.6 mostra em blocos a estratégia HITL proposta para o estudo de caso e esta estratégia segue a disposição da Fig. 6.5. No entanto, uma diferença com o esquema da Fig. 6.5 é o uso de um Manipulador Virtual, o qual permite a visualização antecipada dos movimentos do manipulador. Sua representação em OpenGL corresponde às mesmas dimensões de sua contraparte do manipulador real e consegue interagir com o ambiente real usando marcadores AR.

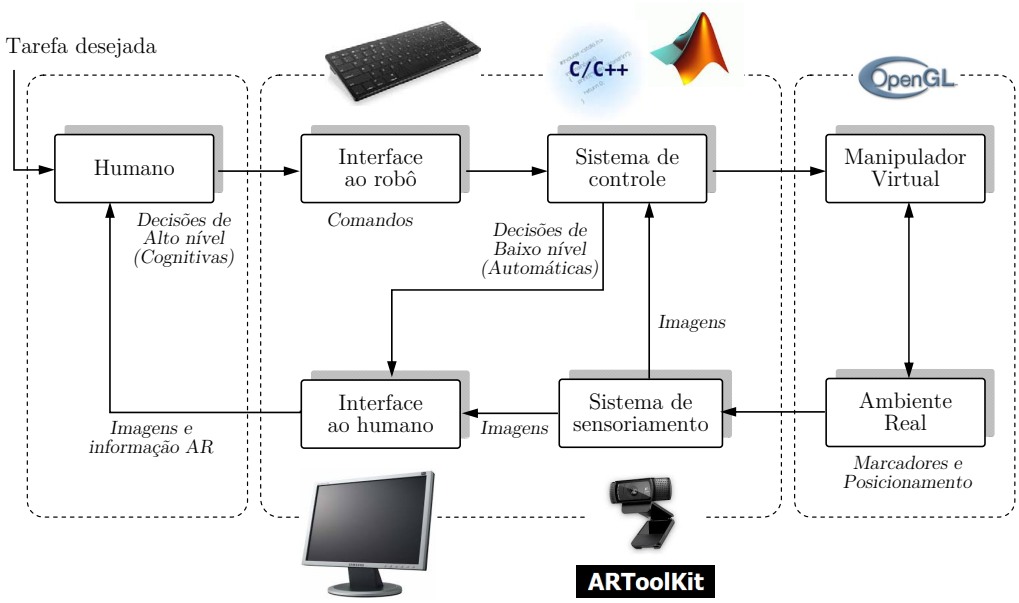


Figura 6.6: Esquema de controle HITL do aplicativo que mistura AR e geração de rotas automáticas.

O *Sistema de sensoriamento* é constituído por uma câmera de alta resolução, onde as imagens são lidas no módulo de *Sistema de controle* com as bibliotecas ARToolKit. As saídas do *Sistema de controle* são decisões de Baixo nível que alimentam as informações visuais da *Interface ao humano*. Uma vez que estas informações são

interpretadas pelo *Humano* usando suas habilidades cognitivas, comandos de teclado são inseridos no *Sistema de controle* para movimentar o *Manipulador virtual*.

6.2.2 Descrição do Aplicativo

Normalmente as imagens transmitidas aos pilotos de ROV não contém informações do ambiente diferente à percepção visual humana. Por exemplo, a foto da Fig.6.7 mostra dois objetos que o humano pode interpretar como uma estrutura de madeira "*quadro*" e uma "*caixa*". Com o uso de marcadores AR é possível dar a mesma informação para um sistema computacional, essa é a ideia de como o software é desenvolvido, e como ele usa as informações do ambiente real para dar predições do comportamento cinemático do manipulador TITAN-4.

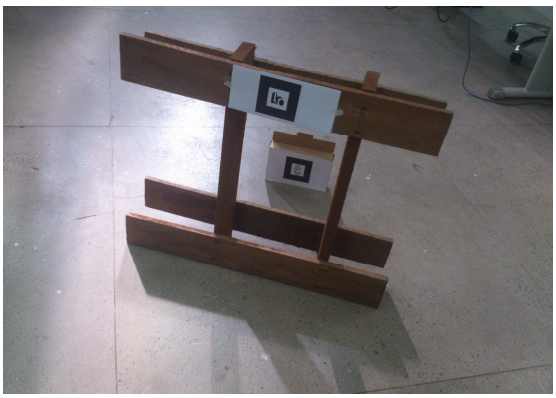


Figura 6.7: Foto de um ambiente real obtida com celular. O *quadro* é um obstáculo em oclusão para atingir a *caixa*

O software foi desenvolvido em C++ funcionando no sistema operacional Microsoft Windows, e sua arquitetura está baseada na implementação modular das bibliotecas indicadas na Fig. 4.16. A Fig. 6.8 apresenta uma captura do ambiente visual do software. As quatro janelas que compõem a interface ao humano são:

1. *Janela de Vídeo/OpenGL (C/C++).* Nesta janela são apresentadas ao usuário as imagens obtidas da câmera e são inseridos os modelos virtuais e informações relevantes da configuração do manipulador virtual. Percebe-se que são virtualizados e apresentados os mesmos dos objetos identificados na foto da Fig. 6.7.

2. *Janela de Comando (MS-Windows).* Usando o teclado, o usuário pode interagir com a inteligência computacional. Esta janela apresenta em menus as interações com o humano.

3. *Janela Gráfica MATLAB.* Nesta janela se apresenta uma visão panorâmica do que acontece na cena real, bem como se consegue apresentar os movimentos automáticos livres de colisões, antes de ser executados na Janela de vídeo.

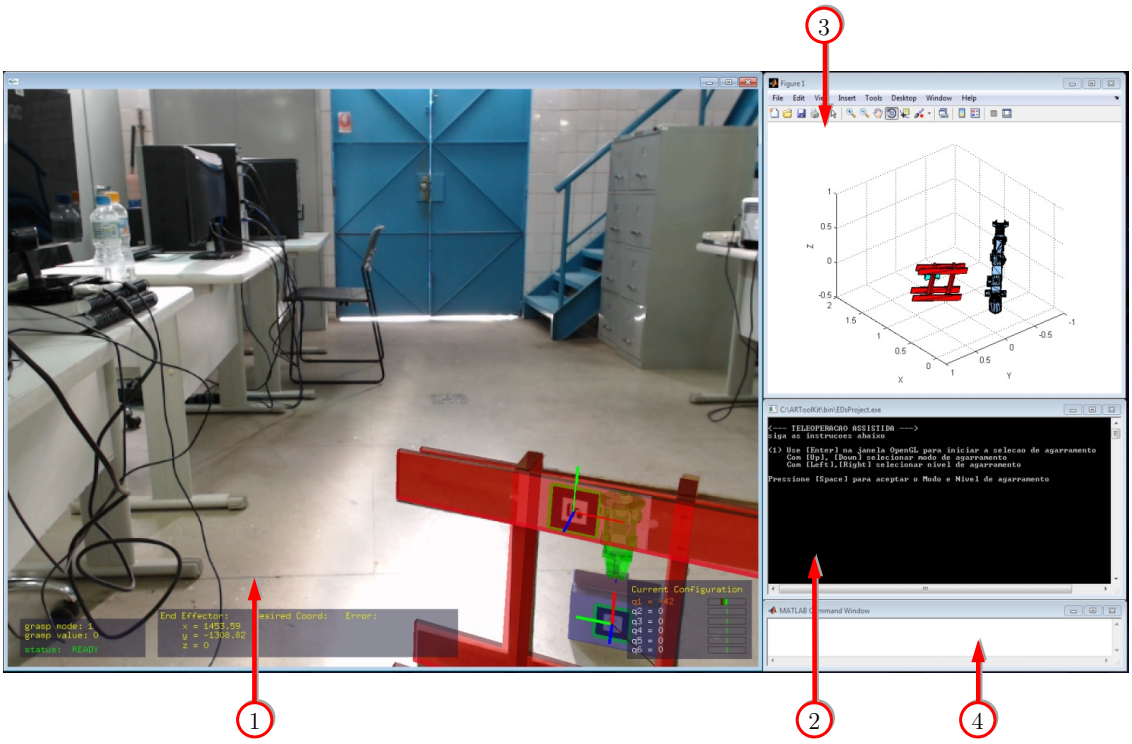


Figura 6.8: Captura do ambiente visual do software.

4. *Janela de Comandos MATLAB*. Esta janela é utilizada exclusivamente para interação avançada com os algoritmos de geração de rotas e modificação das visualizações na Janela Gráfica MATLAB.

Com o fluxograma da Fig. 6.9 pode ser resumida a interação entre as Decisões Cognitivas e as Decisões Automáticas usando o software proposto. O processo começa com a detecção de um (objeto) sólido real que é Alvo do efetuador final para realizar uma tarefa tipo *Pick-And-Place*. Quando é detectado o Alvo na cena, o usuário pode eleger entre uma MANOBRA ASSISTIDA e uma TELEOPERAÇÃO ASSISTIDA.

Uma MANOBRA ASSITIDA se refere ao movimento do Manipulador virtual de acordo com os comandos de seleção de junta e seus respectivos valores. Neste caso a visualização do manipulador na Janela de MATLAB permite perceber rapidamente a configuração atual dos elos do braço robótico, quando o Manipulador não aparece visível na *Janela de Vídeo/OpenGL* (Ver Fig. 3.8).

Quando o usuário seleciona a TELEOPERAÇÃO ASSISTIDA, diversas decisões automáticas são acionadas para auxiliar as decisões do humano. A primeira é o cálculo da cinemática inversa para detectar se o Alvo está ou não no espaço destro do robô. Originalmente esta habilidade é outorgada pela intuição do piloto de ROV, com o software essa tarefa é automática e é mostrada sempre que o Alvo é reconhecido.

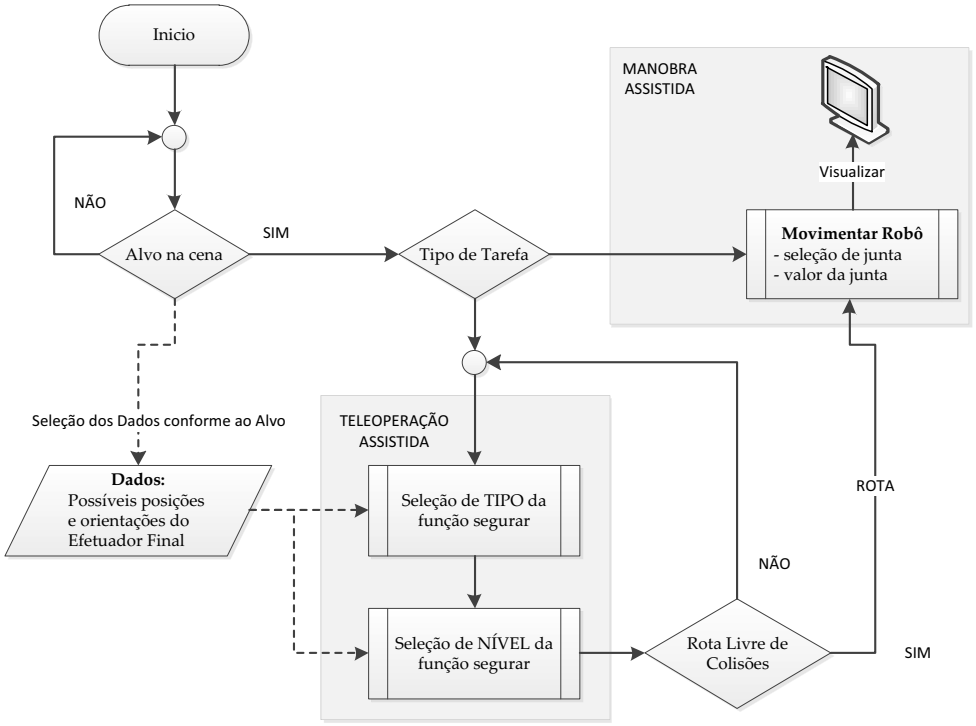


Figura 6.9: Fluxograma da interação humano-máquina.

Na Fig. 6.9 com a TELEOPERAÇÃO ASSISTIDA o usuário deve realizar dois processos: a Seleção de TIPO e a Seleção de NÍVEL. Nestes processos são determinados a posição e a rotação desejada do efetuador final em relação com o Alvo. A Fig. 6.10 ilustra como software, com o usuário seleciona o TIPO e ou NÍVEL. A representação do efetuador final aparece em cada TIPO e NÍVEL, seu cor verde indica que o braço consegue atingir o Alvo. Certamente o TIPO e NÍVEL dependem da geometria do Alvo (Ver Fig. 4.21). Estes dados são pré-carregados de maneira automática quando é reconhecido o Alvo, essa condição é ilustrada na Fig. 6.9 com linhas tracejadas.

Uma vez que o humano decide a posição e orientação final do efetuador final usando TIPO e NÍVEL da função segurar da Fig. 6.9, o software calcula se existe uma serie de movimentos livre de colisões (ROTA) entre a configuração atual e a configuração correspondente à posição e orientação final do EF. Quando é calculada uma rota válida usando o algoritmo RRT-Connect, a *Janela Gráfica MATLAB* apresenta uma animação dos movimentos que o humano deve seguir para movimentar o braço robótico desde a condição atual até a desejada.

Se o humano aceita a ROTA sugerida, ele pode continuar movimentando o manipulador na *Janela de Video/OpenGL* usando uma MANOBRA ASSISTIDA. A Fig. 6.11 apresenta um exemplo do desempenho do software quando o usuário realiza uma MANOBRA ASSISTIDA.

Pode-se ver na Fig. 6.11 que a *Janela de Vídeo/OpenGL* indica como é executado

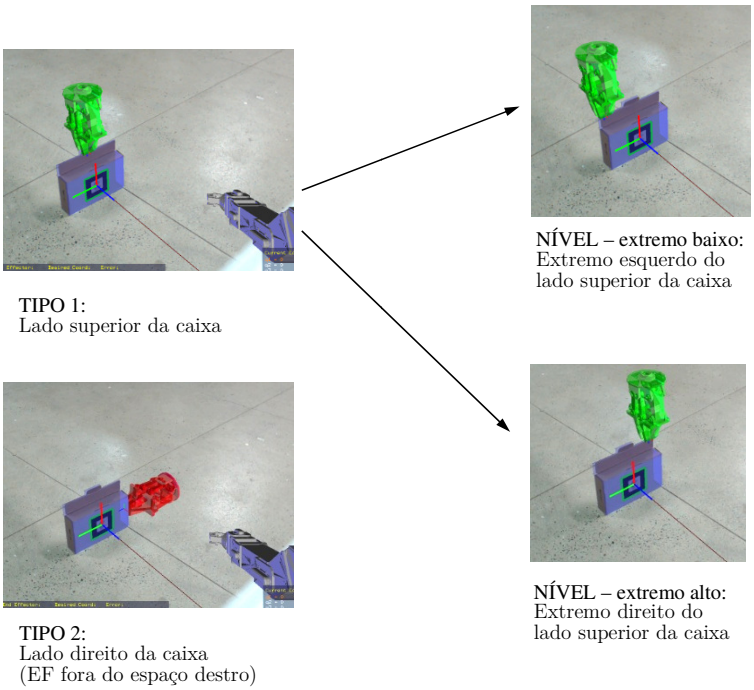


Figura 6.10: Seleção da posição do efetuador final no Alvo usando TIPO e NÍVEl. Na condição TIPO 2, o NÍVEL atual com cor vermelha indica que o braço não consegue segurar a caixa.

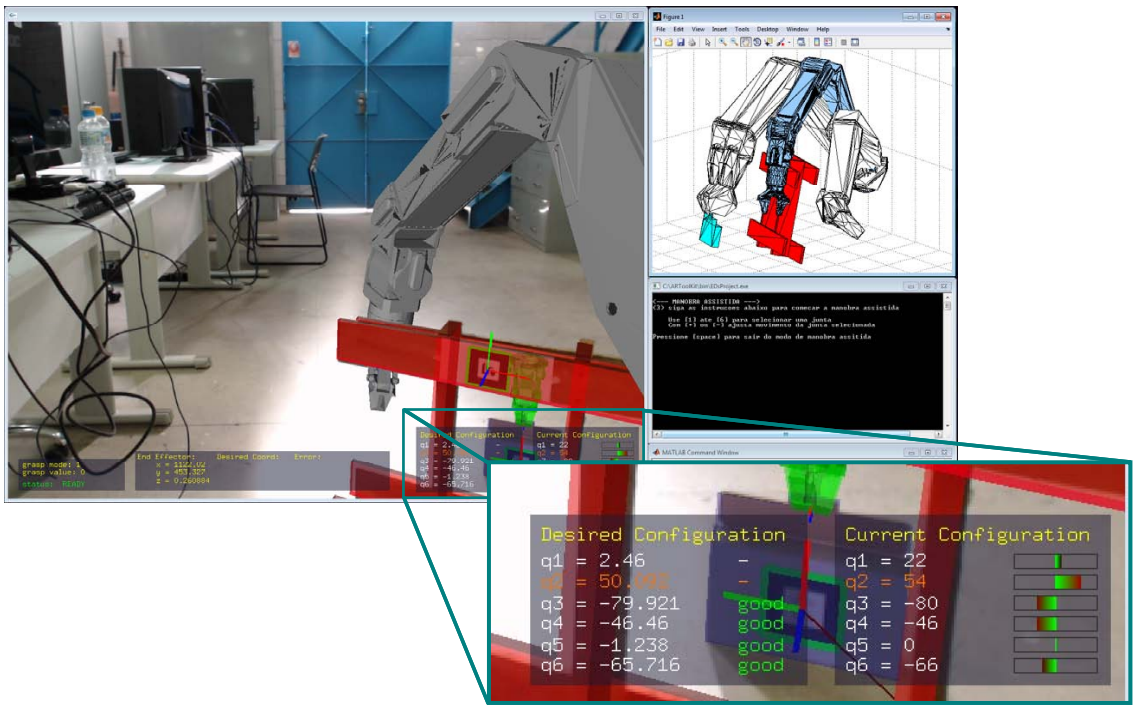


Figura 6.11: Captura de uma MANOBRA ASSISTIDA para manipulação em obstrução e uso de *God's-Eye View* na *Janela de MATLAB*.

o movimento do robô do ponto de vista da câmera. Como existe uma obstrução pelo *quadro*, a *Janela de MATLAB* torna-se uma ajuda valiosa mudando o ponto de visão e apresentando configurações intermediarias para atingir o Alvo *caixa*. Esse tipo de visão é chamada de *God's-Eye View* e acrescenta a percepção do piloto na cena real [192, 193].

Outra característica do software no modo de MANOBRA ASSISTIDA são as informações visuais de texto apresentadas na parte inferior-direita na *Janela de Vídeo/OpenGL*. Um exemplo disto aparece na Fig. 6.11, onde é indicada as variações entre a configuração atual e a desejada, além disso, pode-se ver com as barras de progresso a condição de cada junta antes de atingir os ângulos máximos das mesmas.

6.2.3 Geração Automática das Rotas e a Interação com MATLAB

Uma característica do software desenvolvido é a capacidade de comunicação entre C++ e MATLAB, isto permite aproveitar as vantagens dos comandos e *toolbox* prontos de MATLAB facilitando as futuras pesquisas assim como testes de novos algoritmos de controle. A interação entre C++ e MATLAB é apresentada na Fig. 6.12.

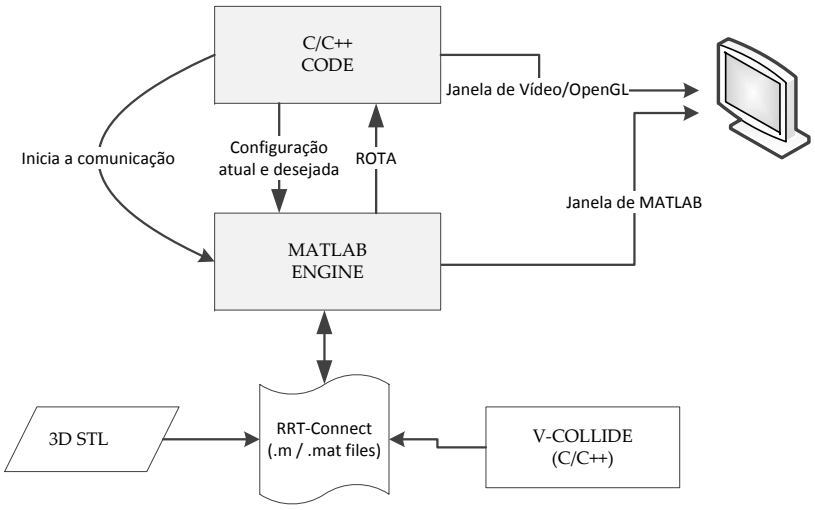


Figura 6.12: Resumo da iteração C/C++ e MATLAB para a geração de rotas livres de colisões.

Desde o aplicativo realizado em C++ e possível iniciar somente o motor de comandos de MATLAB ENGINE sem precisar de todas as funcionalidades visuais. Assim que em funcionamento o motor de comandos, o algoritmo RRT-Connect programado em MATLAB calcula a rota livre de colisões usando as bibliotecas V-COLLIDE e os modelos 3D do manipulador em formato STL (*STereoLithography*).

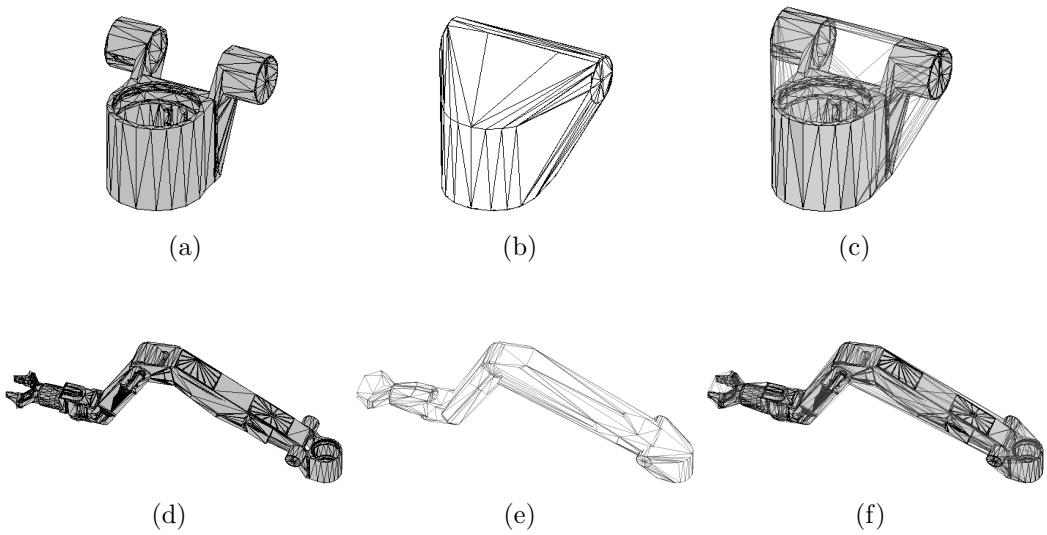


Figura 6.13: Simplificação de vértices para diminuir o tempo na detecção de colisões.

Para melhorar a qualidade da rota é empregado o pós-processamento por corte sequencial estudado na seção 5.3.1. Os resultados finais são apresentados em animação criando uma *Janela Gráfica de MATLAB*, qual exibe todas as funcionalidades embarcadas, tais como zoom, rotação e aparência. Os dados da rota obtida também são usados no código C++ para que o usuário acompanhe a rota movimentando o manipulador virtual em OpenGL usando uma MANOBRA ASSISTIDA.

Um dos aspectos que influencia fortemente no tempo total de resposta é a detecção de colisões, para reduzir seu efeito é preciso usar conceitos de geometria computacional. Neste trabalho são utilizadas as bibliotecas V-COLLIDE para determinar se dois sólidos, triangularmente facetados, estão em condição de colisão. Com um número menor de triângulos para cada sólido, V-COLLIDE aumenta sua velocidade de resposta [157]. Isto pode ser obtido encontrando-se o casco convexo (*Convex Hull*) dos vértices do sólido original.

A Fig. 6.13 apresenta como no MATLAB podem ser obtidas representações simplificadas dos sólidos originais. A Fig. 6.13(a) mostra a representação original do primeiro elo do manipulador. Usando triangulação do casco convexo a Fig. 6.13(b) apresenta uma representação simplificada com um número menor de vértices. A título de comparação, a Fig. 6.13(c) ilustra ambas representações do mesmo sólido, e analogicamente as Fig. 6.13(d), 6.13(e) e 6.13(f) mostram as representações do manipulador completo.

6.2.4 Considerações do Desempenho

O software mostra como vincular a Realidade Aumentada e a geração de rotas livres de colisões visando assim estabelecer uma nova estratégia teleoperação baseada em

Tabela 6.1: Identificação das dificuldades e respetivas soluções estabelecidas pelo software.

Problemática	Solução proposta
Perda da sensação de profundidade na visão monocular. Execução de tarefas de *inserir no furo*.	Cálculo da posição do EF e dos objetos virtualizados em relação à câmera usando os dados da AR. (Capítulo 4).
Estabelecer se efetuador final consegue atingir um alvo, garantindo que o objeto de manipulação esteja dentro do espaço destro do robô.	Uso de dados fornecidos pela AR junto com um rápido algoritmo de cinemática inversa. (Capítulo 3 e 4).
Desconhecimento da configuração atual do manipulador e sua posição relativa ao ambiente.	Uso de uma janela especializada (MATLAB) que mostra uma panorâmica em terceira pessoa. (Capítulo 6).
Encontrar uma posição e orientação válida do EF em relação ao alvo de manipulação.	Base de dados com diferentes Tipos e Níveis da função segurar. Uso de cinemática inversa e visualização em AR. (Capítulo 4 e 6).
Desconforto na execução de uma tarefa *Pick-And-Place*. Múltiplas tentativas para movimentar de maneira certa o manipulador, gerando perda de tempo.	Sistema inteligente que calcula uma serie de movimentos livre de colisões (*Rota*) entre a configuração atual e a desejada. (Capítulo 4 e 5).

HITL. A Tabela 6.1 mostra um resumo de como a ferramenta computacional propõe resolver as dificuldades da teleoperação encontradas no processo de pesquisa.

Outro problema encontrado após do desenvolvimento do software é a maneira como os humanos conseguem acompanhar as rotas geradas automaticamente. Para resolver esses tipos de dificuldades a terceira parte deste capítulo apresenta o contexto deste tipo de problema e uma metodologia de solução baseada num novo algoritmo de geração de rotas.

Na teleoperação o tempo de atraso entre à ordem e a execução do manipulador é uma preocupação constante. Para sistemas totalmente automatizados existem estratégias de controle por laço fechado para minimizar o efeito do atraso. Para a estratégia de teleoperação HITL, o atraso é percebido pelo humano entre o momento de digitar um comando e as imagens exibidas na tela, conseqüentemente é um esquema de controle por laço aberto. No caso do software desenvolvido o tempo de atraso é regulado pelo conjunto das bibliotecas ARToolKit com o tempo entre a captura de uma imagem e a superposição dos sólidos virtuais no vídeo após cálculos cinemáticos do manipulador.

Em relação ao ambiente experimental, ele é constituído por:

- **Hardware:**
  - Processador Intel i5 QuadCore 2.66GHz
  - 8 GB de memória DDR3

- Placa de vídeo ATI Radeon HD 5400
- Câmera vídeo USB PC, Logitech C920 – Full HD de 1080p

- **Software:**
  - Windows 7 Professional Versão 2009
  - Microsoft Visual Studio C/C++ 2010
  - ARToolKit 2.72 - 2006
  - MathWorks MATLAB R2013b

Para os experimentos deste trabalho foi ajustado uma resolução de captura de 640×480 pixels e uma taxa de amostragem de 30 quadros por segundos (fps). Os diversos resultados com o software mostraram que não há atraso de tempo apreciável quando é realizada uma MANOBRA ASSISTIDA, incluindo o uso da segunda janela de exibição (MATLAB). No entanto, no momento da geração de rotas automáticas, MATLAB precisa de alguns segundos para calcular a solução do algoritmo RRT-Connect e seu pós-processado. A Fig. 6.14 apresenta uma captura da *Janela de comando* na obtenção de uma rota automática para o exemplo indicado na Fig. 6.11.

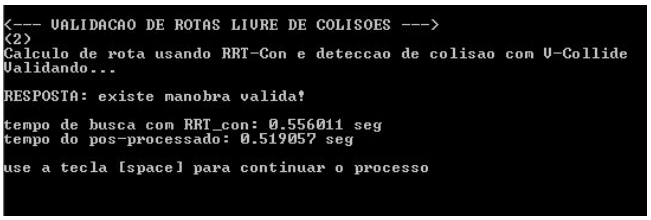


Figura 6.14: Exemplo de resposta na *Janela de comando* para uma geração de rota automática.

Outro aspecto que influencia o desempenho do software é a detecção dos marcadores, as condições de luz, o ângulo e suas dimensões (Ver capítulo 4). Os experimentos contaram com uma combinação aleatória de diversas fontes de luz artificial e natural. Os melhores resultados foram obtidos com a câmera de vídeo em alta definição (HD) e foco automático mesmo em distâncias de apenas 20 cm. Mais um exemplo do desempenho nestas condições é apresentado na Fig. 6.15 onde se ilustra os resultados para uma MANOBRA ASSISTIDA com o problema típico de perda da sensação de profundidade, onde os marcadores AR são padrões de 8 cm com posições e orientações diferentes em relação a câmera. Pode ser constatado o posicionamento final do manipulador com três tipos de ajudas visuais: visualização em primeira pessoa na *Janela de Vídeo/OpenGL*, com dados numéricos na mesma janela e a visualização de panorâmica em terceira pessoa na *Janela Gráfica MATLAB*.

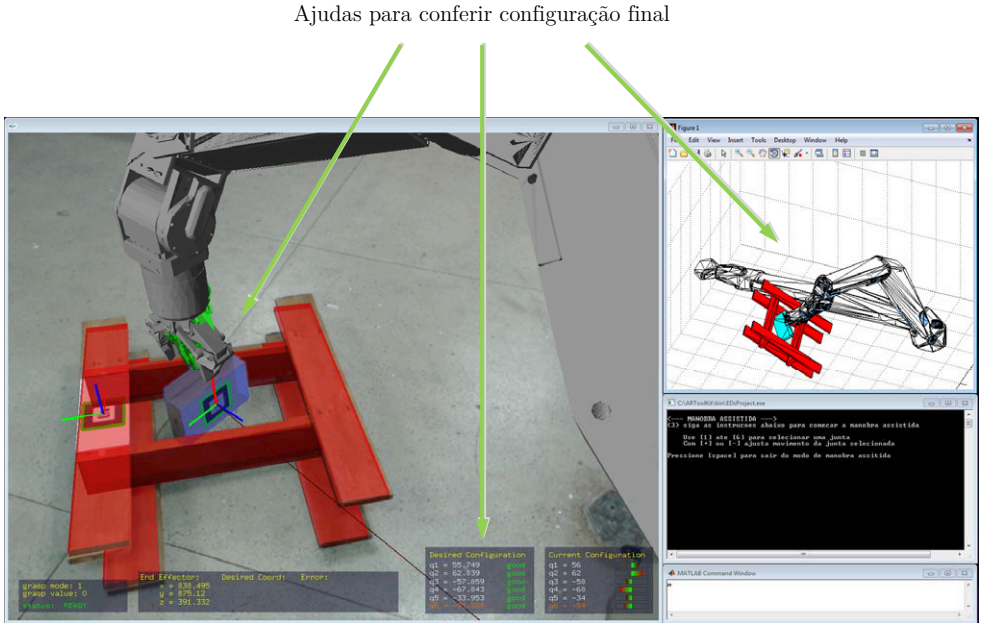


Figura 6.15: Captura de manipulação típica de perda da sensação de profundidade monocular.

Faz-se necessário mencionar que o software desenvolvido é de caráter acadêmico e sua finalidade é demonstrar as capacidades das tecnologias envolvidas. Diversas questões devem ser resolvidas para aplicação industrial. Uma delas é a confiabilidade na detecção de marcadores AR. Para isto alguns estudos apontam a necessidade de se melhorar a tecnologia de realidade aumentada usando múltiplos marcadores fiduciais [194, 195], marcadores naturais [196, 197] e visão estéreo [198, 199]. Por outro lado, para o estudo de caso, tem-se alguns avanços no uso de marcadores AR em ambientes submarinos como em [200–202].

6.3 Planejamento de Movimentos por Junta Independente

Esta parte do Capítulo apresenta uma nova estratégia de *Path Planning*, baseada na exploração rápida e incremental do espaço de configuração para obter rotas livres de colisões que podem ser facilmente acompanhadas por pilotos humanos. As rotas obtidas são uma sequência de movimentos com uma junta de cada vez. Ao contrário das estratégias RRT (seção 2.4.6 e 2.4.7), o método proposto utiliza um crescimento incremental na direção de um único grau de liberdade de cada vez. Uma simples heurística é utilizada para concretizar uma estrutura em forma de árvore, a qual garante a obtenção de soluções semelhantes ao RRT, mas, sem a necessidade de parâmetros de crescimento ou ajuste de parâmetros probabilísticos algum.

A organização desta parte é a seguinte: a seção 6.3.1 tipifica o problema de *Path*

(a) Schilling Robotics - Rate Hand Controller

(b) Schilling Robotics - Master Controller

Figura 6.16: Tipos de interface de controle dos manipuladores submarinos. Imagens obtidas de [13].

Planning no caso da teleoperação; a seção 6.3.2 e 6.3.3 explica a abordagem proposta para versões básicas e bidirecionais do novo algoritmo; a seção 6.3.4 apresenta os conceitos de otimização; a seção 6.3.5 ilustra um método de pós-processamento para conseguir rotas de boa qualidade; a seção 6.3.6 apresenta os resultados experimentais com simulações de casos de referência; finalmente, a seção 6.3.7 expõe as principais conclusões e perspectivas.

6.3.1 Planejamento de rotas e teleoperação

Normalmente num sistema teleoperado o piloto possui uma interface adequada para guiar os movimentos do robô numa sequência desejada. Em geral estas interfaces podem ser joysticks ou dispositivos hápticos como os usados na cirurgia robótica. No estudo de caso, o controle dos manipuladores submarinos é realizado com dispositivos que atuam diretamente em cada atuador de maneira simples (*Rate Control*) ou usando uma réplica pequena do manipulador real onde suas configurações são correspondentes (*Position Control*).

A Fig. 6.16 mostra dois tipos de interfaces dos manipuladores submarinos. O *Rate Control* é a forma mais simples de controle, combinando todas as funções em um joystick com botões. O *Rate Control* é uma interface econômica, robusta e pode ser usada com uma mão, não entanto, precisa-se de treinamento para obter o movimento contínuo de múltiplas juntas. Paralelamente, com o *Position Control* se controlam várias juntas simultaneamente resultando em movimentos precisos e contínuos do manipulador. Obviamente o *Position Control* é um sistema mais complexo porque a posição das juntas tem de ser detectada e realimentada para que o controlador mestre envie sinais adequados da posição às válvulas dos atuadores. Consequentemente surge a necessidade de recalibrar periodicamente o *Position Control* para obter um

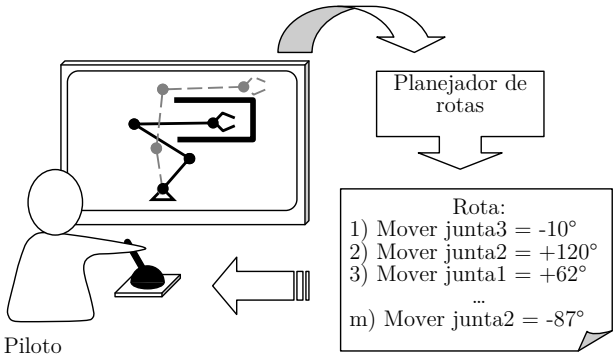


Figura 6.17: Esquema que combina a geração de rotas e a teleoperação.

bom desempenho da teleoperação.

Certamente, o treinamento e a experiência do humano desempenha um papel importante na teleoperação, onde é necessário deduzir uma sequência de movimentos do robô sem produzir colisões nos ambientes não-estruturados. Como foi exibido em capítulos anteriores, existem algoritmos capazes de resolver este problema. No entanto, combinar o julgamento humano e as rotas automáticas livres de colisões é um tema pouco estudado. Um exemplo que mostra como o planejamento automático ajuda na direção de robôs móveis é apresentado em [203] e o uso destes planejadores como sistema de treinamento dos pilotos humanos da estação espacial internacional é ilustrado em [204].

A principal dificuldade no acompanhamento de rotas automáticas é a coordenação simultânea de múltiplos graus de liberdade. Isto é, para o humano movimentar simultaneamente os 2-GDL de um robô planar é um exercício relativamente fácil, mas, tentar movimentar um robô cobra com múltiplos graus de liberdade é uma tarefa complexa. Consequentemente, o desafio é obter uma rota que seja fácil de seguir por o humano.

Uma solução natural é encontrar a lista mais curta possível de movimentos independentes sem produzir colisões. Esta ideia torna-se um tipo de problema discreto de *Path Plannning*. Existem estratégias para solucionar de maneira exata o problema discreto de planejamento de rotas para espaços de configuração com baixa dimensionalidade, contudo, no caso de manipuladores com n-GDL>3 o problema é intratável [11].

A Fig. 6.17 ilustra o contexto para misturar teleoperação e planejamento de rotas automáticas, onde uma lista de movimentos é apresentada ao piloto para controlar o manipulador de maneira remota.

A primeira ideia para conseguir essa lista de movimentos é transformar uma rota obtida com um planejador probabilístico em uma rota discreta. Esta abordagem cria um novo problema de *path planning*. Por exemplo, a Fig. 6.18 mostra uma simples rota em $\mathcal{Q}^2$ composta por uma borda e dois vértices q_1 e q_2, o que corresponde às duas

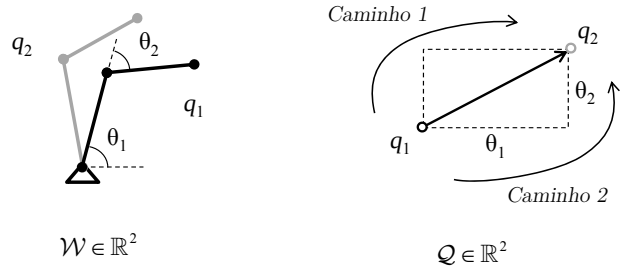
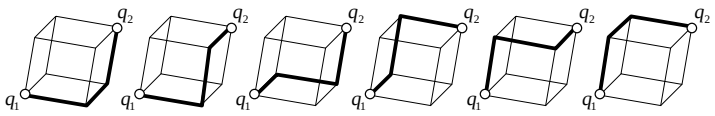


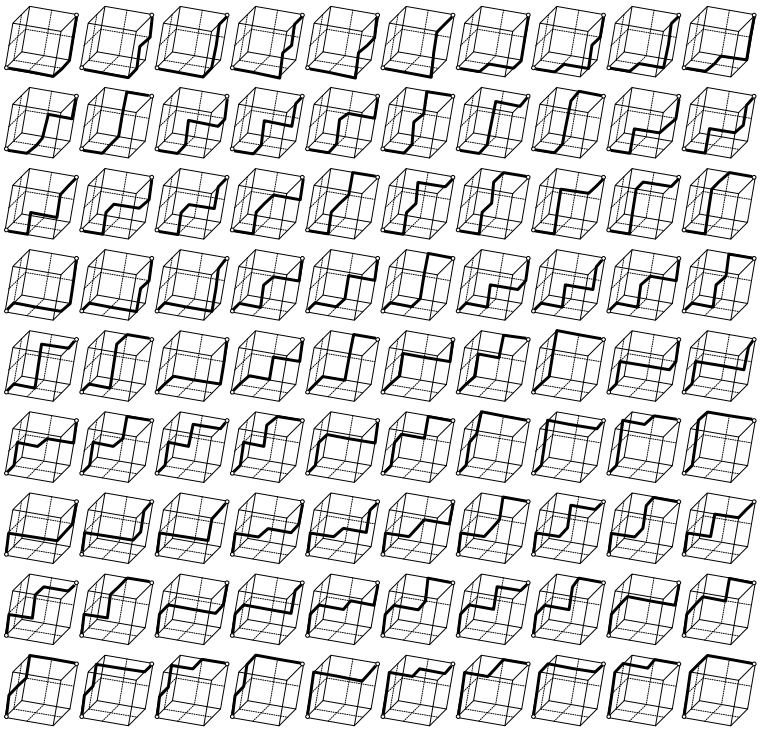
Figura 6.18: Discretização de uma simples rota com uma única borda.

configurações do robô planar de 2-GDL no espaço de trabalho $\mathcal{W}$. Ao discretizar essa rota, uma borda produz dois novos caminhos: $\{\theta_2, \theta_1\}$ e $\{\theta_1, \theta_2\}$. No entanto, na obtenção de melhor resolução para evadir obstáculos e consecução de rotas de melhor qualidade, o espaço de busca disponível pode ser divido em uma grade. É claro que novas rotas serão estabelecidas.

Por exemplo, no caso de $q = [\theta_1, \theta_2, \theta_3]^T \in \mathcal{Q}^3$ uma simples rota de uma borda, tem-se seis possíveis caminhos $n! = 3! = 6$, ver Fig. 6.19(a). Entretanto, com uma grade de $2 \times 2 \times 2$ o número de rotas obtidas são $(2n)!/(2!)^n = 90$, ver Fig. 6.19(b).



(a) grade de $1 \times 1 \times 1$: 6 rotas



(b) grade de $2 \times 2 \times 2$: 90 rotas

Figura 6.19: Discretização de uma simples borda em $\mathcal{Q} \in \mathbb{R}^3$.

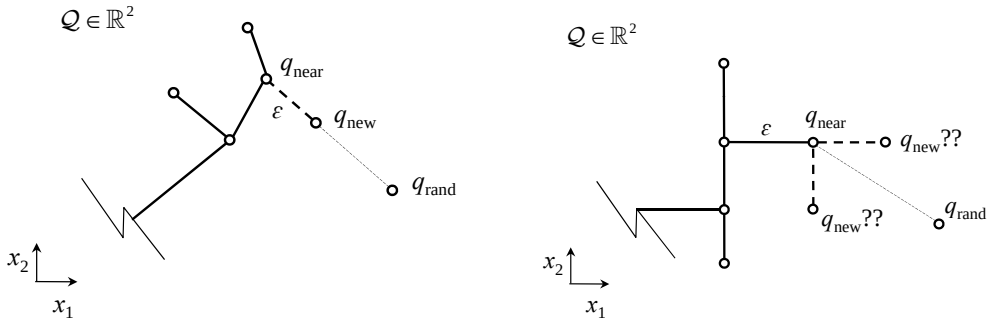
Para sistemas multidimensionais o numero de soluções obtidas com a discretização é grande e aumenta em consideração do valor da grade e uma rota original composta de múltiplas bordas e vértices. O número de soluções pode ser calculado com:

$$\text{Rotas} = m!\frac{(n \cdot l)!}{(l!)^n} \tag{6.1}$$

Onde: n e a dimensão do espaço de busca, l o valor homogêneo da grade e m é o número de bordas da rota original.

Por outro lado, em lugar de tentar adaptar uma rota automática aos critérios de discretização, uma alternativa viável é usar um método para obter diretamente a rota discretizada. As Árvores Aleatórias para Exploração Rápida RRT são capazes de obter soluções em ambientes multidimensionais e complexos, no entanto, seu crescimento incremental por amostragem não é projetado para estes tipos de problemas discretos.

A Fig. 6.20(a) ilustra o crescimento normal RRT onde uma amostra q_{rand} produz um novo vértice q_{new} com um comprimento ϵ desde q_{near}, que é o vértice mais próximo da árvore até q_{rand}. A estrutura de árvore desejada é mostrada na Fig. 6.20(b), não obstante, o sistema de amostras não determina como deve crescer a árvore.

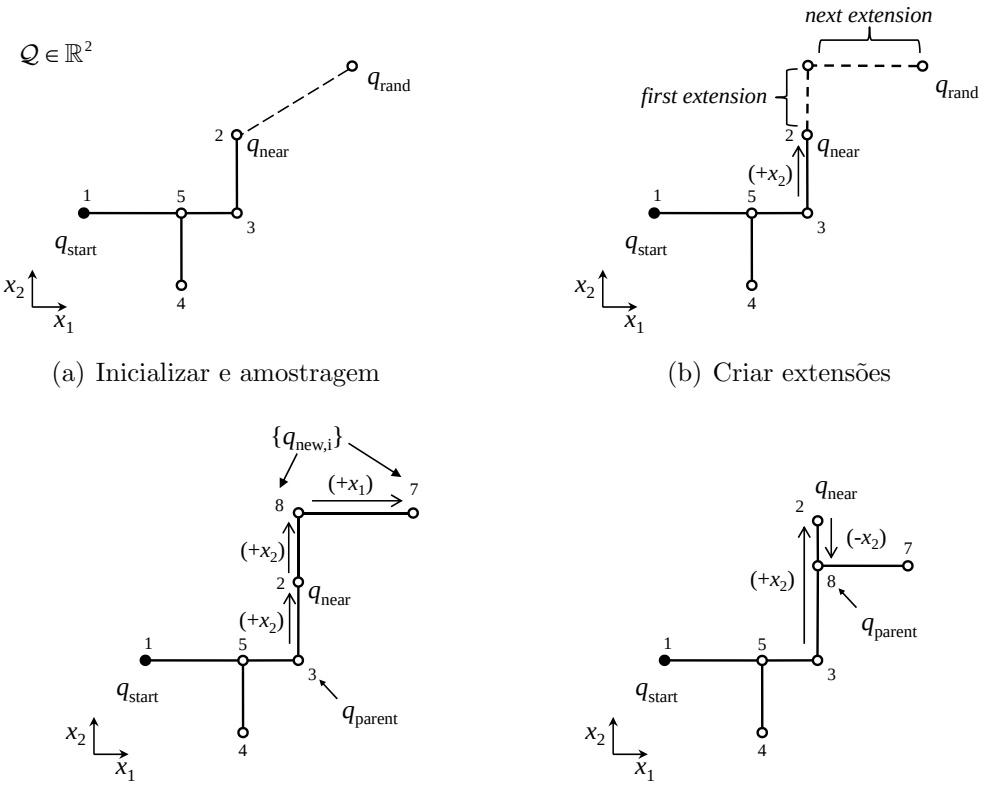


(a) RRT (b) desejado por eixo

Figura 6.20: Tipos de crescimento incremental para exploração rápida.

6.3.2 Algoritmo C-RDT

Para dar solução ao questionamento anterior, esta seção introduz o algoritmo Planejador baseado na Combinação de extensões para Árvores Densas de Exploração Rápida ou C-RDT (*Combinational-based Random Dense Tree planner*). O algoritmo explora o espaço de busca construindo uma árvore combinando extensões que são guiados pelos eixos do espaço de busca. A estratégia proposta tem um comportamento similar aos *Rapidly Exploring Dense Trees* (RDTs) descritos em [11] devido à maneira como é realizada a cobertura do espaço de busca.



(a) Inicializar e amostragem

(b) Criar extensões

(c) Validação

(d) Reescrever pais

Figura 6.21: Descrição bidimensional da estratégia C-RDT

Alem disto, Algoritmo C-RDT explora a informação encontrada em vértices sobre o crescimento passado da árvore para determinar extensões subsequentes em direção a uma configuração aleatória. Esta estratégia não requer um rota predefinida, qualquer parâmetro de crescimento, ajuste de parâmetros probabilísticos ou abordagem de grade. Para facilitar a compreensão, a seguir é explicado como funciona o algoritmo C-RDT para um espaço de busca bidimensional.

- **Inicializar e amostragem.** O C-RDT começa com uma configuração inicial q_{start} como raiz da árvore T. Para cada iteração, o processo de extensão usa uma amostra q_{rand} dentro do $\mathcal{Q}$. Em seguida, calcula-se o vértice $q_{near} \in T$ mais próximo de q_{rand} (ver Fig. 6.21(a)).

- **Criar extensões.** A árvore tende crescer a partir de q_{near} para q_{rand} usando uma sequência de ramais paralelos aos eixos x_1 e x_2. A primeira extensão é selecionada pela direção do vetor q_{parent} até q_{near}. No exemplo da Fig. 6.21(b), O processo de conexão entre q_{near} e q_{rand}, pode-se extrair a direção de $q_{parent} = q_3$ até $q_{near} = q_2$, isto é $d_{3,2} = +x_2$. Portanto, a primeira extensão é a componente x_2 do vetor $q_{rand} - q_{near}$. Consequentemente, a segunda extensão é a componente x_1 do mesmo vetor $q_{rand} - q_{near}$. No caso dos espaços de busca

multidimensionais $\mathcal{Q}^n$, uma vez que a primeira extensão é definida, as outras $n-1$ extensões são selecionadas de maneira aleatória.

- **Validação e novos galhos.** Se as extensões não produzem colisões no espaço de trabalho, ambas as bordas e vértices são adicionados à árvore. No exemplo da Fig. 6.21(c) o conjunto de vértices $\{q_{new,1}, q_{new,2}\} = \{8, 7\}$ as suas direções de crescimento $\{+x_2, +x_1\}$ são adicionadas à estrutura da árvore. No caso multidimensional, a validação e adição de novos galhos podem ser realizados em sequência começando em q_{near}. De modo que, se q_{rand} não é válido, algumas extensões podem ser adicionadas à árvore.

- **Reescrever informação dos pais.** Nos casos em que a primeira extensão vai no sentido oposto do crescimento imediatamente anterior do vértice q_{near}, é preciso reescrever as informações do pai de q_{near}. A Fig. 6.21(d) mostra um exemplo onde a primeira extensão tem a direção $d_{2,8} = -x_2$ contrária de $d_{3,2} = +x_2$. Portanto, o novo pai de q_{near} é precisamente o novo vértice $q_{new,1} = q_8$; e o pai de q_8 é o anterior pai de q_{near}, isto é o vértice q_3. Em espaços de busca Euclidianos, a direção entre dois vértices $\{q_a, q_b\}$ pode ser calculada com a Eq. (6.2).

$$d_{a,b} = \frac{q_b - q_a}{\|q_b - q_a\|} \tag{6.2}$$

A descrição formal em pseudocódigo do C-RDT é apresentado nas Fig. 6.22 e Fig. 6.23. O algoritmo Fig. 6.22 tem a mesma formulação de qualquer estratégia RRT, a diferença está na operação EXTEND indicada na Fig. 6.23. Com NEAREST_NEIGHBOR é calculado o vértice mais próximo de T para a amostra q_{rand}. Se q_{near} é o vértice raiz, não existe informação do crescimento anterior, Logo, com RANDOM_DIR é designado uma direção aleatória d_{near} paralela algum dos eixos do espaço de busca.

SORT_COORDINATES é responsável pela geração de novas extensões utilizando os componentes do vetor q'. SORT_COORDINATES também estabelece a primeira

Algoritmo 9: BUILD_C-RDT(q_{init})

```
1 T.init(q_init);
2 for k = 1 to K do
3 |   q_rand ← RANDOM_CONFIG();
4 |   EXTEND(T, q_rand)
5 return T
```

Figura 6.22: Algoritmo básico C-RDT.

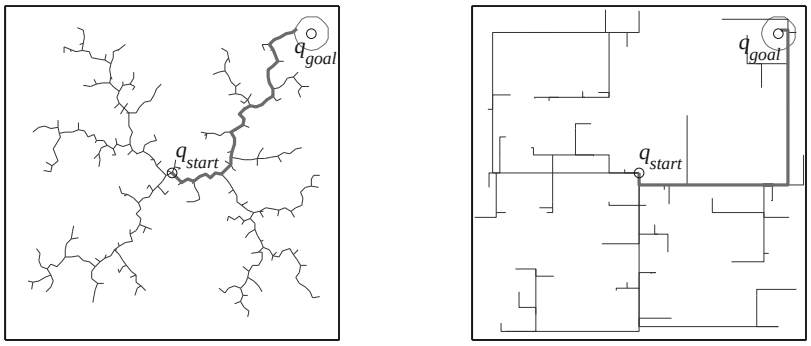
Algoritmo 10: EXTEND(T, q_{rand})

```
1  q_near, d_near ← NEAREST_NEIGHBOR(q_rand, T);
2  if q_near = q_init then
3  |  d_near ← RANDOM_DIR();
4  q' ← q_rand − q_near;
5  {q_new,i} ← SORT_COORDINATES(q', d_near);
6  if VALID({q_new,i}) then
7  |  T.add_vertex({q_new,i});
8  |  T.add_edge({q_new,i});
9  |  if SIGN{q_new,1 + d_near} = −1 then
10 |  |  REWRITE_PARENT(q_near, T);
11 |  return Advanced;
12 else
13 |  return Trapped;
```

Figura 6.23: Função EXTEND do algoritmo C-RDT.

extensão que corresponde à direção do d_{near}, as seguintes extensões são dispostas aleatoriamente. Quando o conjunto de extensões $\{q_{new,i}\}$ são válidas, os vértices e as bordas são adicionadas à estrutura da árvore. Quando o crescimento anterior e a primeira extensão tem sentido diferente, a informação do vértice pai de q_{near} é reescrita, excetuando quando q_{near} é o vértice raiz.

Do mesmo modo que acontece com o RRT básico, o C-RDT encontra uma rota solução quando algum dos vértices da árvore T está suficientemente perto de uma configuração meta q_{goal}. A Fig. 6.24 ilustra o comportamento similar dos algoritmos básicos RRT e C-RDT.



(a) RRT básico (b) C-RDT básico

Figura 6.24: Exemplo do crescimento em um espaço de busca bidimensional livre de obstáculos.

6.3.3 Algoritmo bidirecional - BiCRDT

As abordagens bidirecionais de busca para planejadores incrementais proporcionam melhor precisão da rota entre as duas configurações e diminui o tempo geral de convergência [111]. A Fig. 6.25 apresenta a abordagem bidirecional do C-RDT, nomeado de BiCRDT.

Algoritmo 11: BiCRDT(q_{init}, q_{goal})

```
1 T_a.init(q_init); T_b.init(q_goal);
2 for k = 1 to K do
3     q_rand ← RANDOM_CONFIG();
4     if not EXTEND(T_a,q_rand) = Trapped then
5         if EXTEND(T_b,q_near) = Reached then
6             return PATH(T_b,T_b);
7     SWAP(T_a,T_b);
8 return Failure
```

Figura 6.25: Algoritmo BiCRDT.

Do mesmo modo que a estratégia RRT-Connect, o BiCRDT usa duas árvores enraizadas nas configurações de início e meta. A cada iteração a árvore T_a cresce conforme à estratégia C-RDT básica, enquanto a árvore T_b tende se conectar com T_a usando a mesma função EXTEND mostrada na Fig. 6.23. A busca termina quando as árvores conseguem se conectar. Caso contrario, na próxima iteração os comportamento das árvores são trocados utilizando a função SWAP.

6.3.4 Otimização no planejamento de rotas

Apesar dos algoritmos baseados em amostragem logram obter rotas no $\mathcal{Q}$ que refletem os movimentos livre de colisões no $\mathcal{W}$, essas rotas não são ótimas (Ver Fig. 5.25). Frequentemente as rotas resultantes são compostas por uma sequência de segmentos (bordas ou linhas retas) no espaço livre de colisão $\mathcal{Q}_{free}$, e sua distribuição fica com zigue-zagues produzindo movimentos desnecessários no espaço de trabalho $\mathcal{W}$.

Tradicionalmente o conceito de otimalidade é referido aos espaços Euclidianos como o caminho mais curto. Para espaços de busca $\mathbb{R}^2$ é possível imaginar algoritmos de busca incremental usando o *Paradigma Continuo de Dijkstra* que conseguem obter o caminho mais curto. Infelizmente, tais métodos exatos não podem ser generalizados para $\mathbb{R}^3$.

Para os espaços de configuração multidimensionais como os usados para braços robóticos, é difícil conceituar um critério adequado de otimização, em especial

quando as representações de orientação e posição estão embarcadas no mesmo espaço $\mathcal{C} \in \mathbb{R}^n$ [11].

A busca de soluções ótimas também levanta questões interessantes sobre da transformação de um problema contínuo como um problema discreto. Consideremos o caso de dois pontos no plano, em que o caminho mais curto é uma linha diagonal. Podem existir dois caminhos discretos com custos iguais: um dos caminhos tem a forma de escada com vários vértices; o outro consiste de um único passo com um vértice. Por exemplo, na Fig. 6.19 todas as rotas obtidas por discretização têm o mesmo custo. Um critério de otimização para selecionar o melhor caminho é o número de vértices ou *transições* da rota discreta livre de colisões. Em vista disso, uma rota com o menor número de *transições* produz um reduzido número de paradas e movimentos do manipulador. No estudo de caso, donde é requerido mover uma junta por vez, a rota obtida com o menor número de *transições* produz uma lista com o menor número de instruções, que é o objetivo desejado.

O número mínimo de *transições* é associado com as configurações inicial e final, junto com a dimensão do espaço de busca. Considere-se a função de alteração σ definida pela Eq. 6.3; sua aplicação para duas configurações $q_a = [a_1, a_2, \ldots a_n]$ e $q_b = [b_1, b_2, \ldots b_n]$ num espaço n-dimensional, é expressada pela Eq. 6.4.

$$\sigma(a,b) = \begin{cases} 1, & a-b>1 \quad \vee \quad a-b<1 \\ 0, & a-b=0 \end{cases} \tag{6.3}$$

$$\sigma(q_a, q_b) = \{\sigma(a_1, b_1), \sigma(a_2, b_2), \ldots \sigma(a_n, b_n)\} \tag{6.4}$$

Assim, tem-se o número de alterações c entre duas configurações q_a e q_b é determinado por:

$$c = \sum_{i=1}^{n} \sigma(a_i, b_i) \tag{6.5}$$

Partindo de um ambiente sem obstáculos, um robô consegue se deslocar desde q_a até q_b com um número mínimo de transições igual a c. Desta maneira, no melhor dos casos para o problema de *Path Planning* discreto, uma rota ótima não pode ter um número de transições inferior a c.

Assim como as estratégias RRTs, os C-RDTs geram rotas de baixa qualidade. Na próxima seção é apresentado um algoritmo simples de pós-processamento que usa como entrada a rota obtida por os métodos C-RDTs e cujo resultado é uma nova rota com custo que tenta aproximar-se ao critério ideal de otimização c.

6.3.5 Pós-processamento

Nesta seção apresenta-se uma nova técnica de pós-processamento que consegue reduzir o número de *transições* e uma diminuição do comprimento da rota. O método proposto obtêm atalhos livres de colisão de maneira iterativa entre os vértices da rota de entrada, de modo similar ao pós-processamento por corte sequencial exibido na seção 5.3.

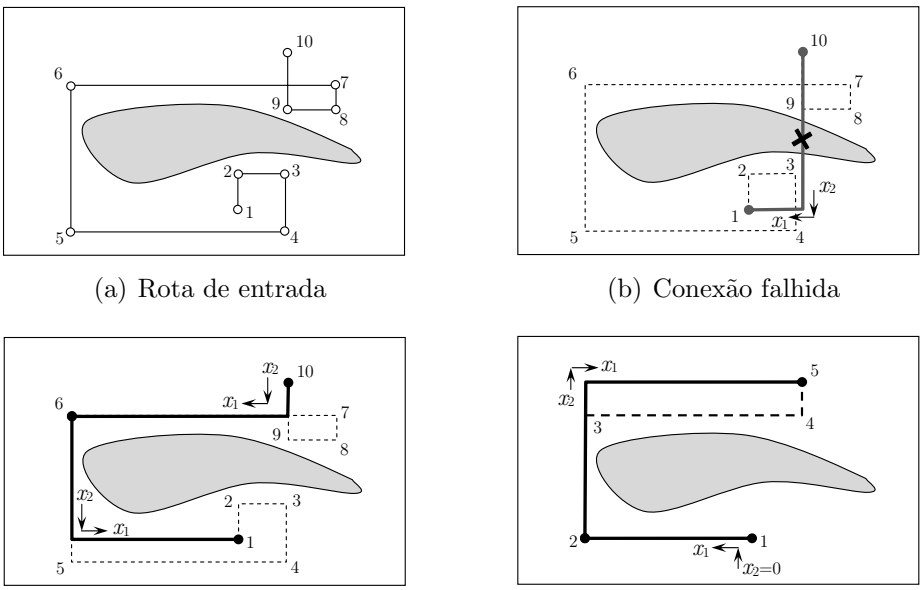
O método de pós-processamento pode ser resumido como segue para um espaço bidimensional:

- Dada uma rota de entrada, como a ilustrada na Fig. 6.26(a), a primeira ação é tentar conectar o último vértice com o primeiro. A função EXTEND poderia ser usada para obter essa conexão. Mas, considerando que nos robôs seriais a condição de posição e rotação de um elo j depende da condição do elo imediatamente anterior $j-1$, a função EXTEND pode ser substituída por uma série ordenada de extensões por eixo começando com a última junta $j = n$ até a primeira $j = 1$. Convém lembrar que os eixos do espaço de configuração estão compostos por cada GDL.

- A Fig. 6.26(b) mostra um exemplo no $\mathcal{Q} \in \mathbb{R}^2$ onde a primeira extensão usa a última coordenada x_2 e extensão posterior é realizada na coordenada x_1. As extensões tentam conectar diretamente o vértice pivô $i = 10$ com o primeiro vértice de prova $k = 1$. A primeira extensão tem uma colisão com o obstáculo central; em consequência, toda a conexão falhou.

- O processo de ligação é repetido aumentando k até que exista uma conexão válida. Por exemplo, a Fig. 6.26(c) indica que depois de verificar as extensões entre $i = 10$ e os vértices $k = \{1, 2, 3, \ldots 9\}$, existe uma conexão válida quando $k = 6$. Em seguida, um novo vértice pivô é atribuído como $i = 6$. Logo o processo de conexão é repetido com $i = 6$ e $k = \{1, 2, \ldots 5\}$. Como há uma conexão entre $i = 6$ e o primeiro vértice $k = 1$, o processo é interrompido e a resposta de pós-processamento é gerada.

O pós-processamento explicado acima elimina laços desnecessários e reduz o comprimento do percurso. No entanto, a procura heurística unidirecional desde a meta ao inicio produz soluções subótimas para o caso do critério de *transições*. Para simplificar a soluções ainda mais, é usado pela segunda vez o método do pós-processamento trocando as duas configurações de modo que é revertida a direção dos cortes sequenciais. Não obstante que o segundo pós-processamento não garanta em todos os casos a obtenção das soluções ótimas, as rotas obtidas ficam perto das condições ótimas no critério das *transições*.



(a) Rota de entrada

(b) Conexão falhida

(c) Rota resposta

(d) Processo revertido

Figura 6.26: Pós-processamento de rota com extensões por eixo.

A Fig. 6.26(d) ilustra um exemplo da aplicação de um segundo pós-processamento, obtendo-se uma solução ótima. A rota de entrada é o percurso encontrado com o primeiro pós-processamento $\{1, 2, 3, 4, 5\}$. Neste caso primeira iteração tenta conectar o vértice $i = 1$ incrementalmente com o conjunto de vértices $k = \{5, 4, 3, 2\}$. O final do processo é estabelecido com a extensão entre o pivô $i = 2$ e o vértice $k = 5$. Como resultado a rota final é indicada na mesma figura pela linha contínua.

A representação formal de pós-processamento no pseudocódigo é ilustrado na Fig. 6.27. Na primeira linha deve-se adquirir o número de vértices da rota de entrada. Na linha dois, é inicializada a nova rota. As linhas três e quatro iniciam os vértices de pivô e de prova. Já na linha seis é criado um vetor temporário q' que abriga as magnitudes das extensões por eixo. Na linha sete, a função INVERT_COORDINATES usa as coordenadas do vetor q' para calcular o conjunto de extensões desde q_i para q_k. Quando as extensões são válidas, novos vértices são adicionados à nova rota de pós-processamento na linha nove. Em conseqüência disso, i e k são atualizados. Caso contrário, o processo continua com outro vértice de prova.

Como outras abordagens de pós-processamento, o algoritmo da Fig. 6.27 converge para uma solução subótima se for dada uma rota de entrada pobre. Isto é consequência da natureza exploratória probabilística do C-RDT. Tal como é explicado em [11], a principal preocupação dos métodos baseados em amostragem é *viabilidade* em oposição à *otimalidade*.

Algoritmo 12: Postprocessing(Path)

```
1  m ← #vertices of Path;
2  NewPath.init(Path(m));
3  i ← m "pivot vertex";
4  k ← 1 "test vertex";
5  while i ≠ 1 do
6      q' ← (Path(k)−Path(i));
7      {q_new,i} ← INVERT_COORDINATES(q');
8      if VALID({q_new,i}) then
9          NewPath.add({q_new,i});
10         i ← k; k ← 1;
11     else
12         k ← k + 1;
13 return NewPath
```

Figura 6.27: Algoritmo de pós-processamento para extensões por eixo.

6.3.6 Testes com C-DRTs

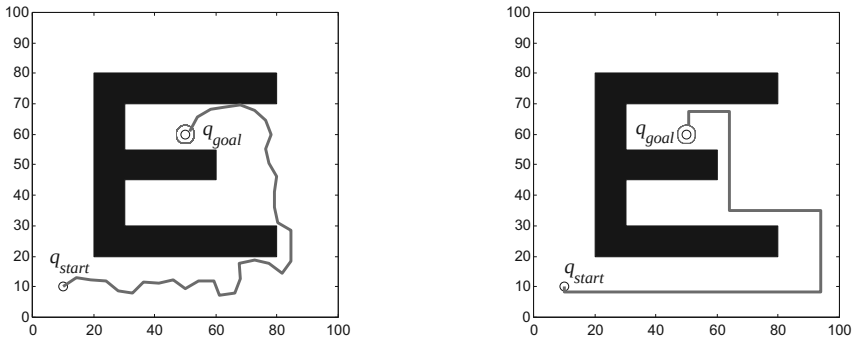
A fim de demonstrar o funcionamento das metodologias propostas, três cenários representativos são ilustrados com simulações. O primeiro apresenta o desempenho de C-RDT básico. O segundo mostra o caso de um robô planar de 3-GDL aplicando o BiCRDT e pós-processado. No terceiro cenário consta do sistema multidimensional do caso de estudo TITAN-4 obtendo-se rotas quase-ótimas. Para os testes se dispõe da programação no software MATLAB e da importação de arquivos de modelação 3D em formatação STL; para a detecção de colisões são usadas as bibliotecas V-COLLIDE disponíveis em C/C++.

Desempenho em espaço de busca 2D

A Fig. 6.28 indica um espaço de busca bidimensional com um obstáculo central, a letra "E". As condições inicial e final são representadas pelos pontos q_{start} e q_{goal} dentro do $\mathcal{Q}_{free}$ e onde foram testados as estrategias RRT e C-RDT. Na Fig. 6.28(a) mostra a típica resposta do algoritmo RRT-Básico, usando um parâmetro de crescimento $\delta = 1$. Paralelamente, a Fig. 6.28(b) apresenta a rota típica obtida com o C-RDT básico.

Ambas as estratégias foram avaliados 1000 vezes com 100% de sucesso. Os resultados estão resumidos na Tabela 6.2 e Tabela 6.3. Embora que o tempo médio do C-RDT é levemente melhor que o exibido pelo RRT-Basic, em termos gerais, neste caso não há uma diferença estatística significativa entre os critérios ponderados das duas estratégias.

Uma das principais fraquezas dos algoritmos RRT é a ultrapassagem de



(a) RRT Básico (b) C-RDT Básico

Figura 6.28: Típicas rotas obtidas num espaço bidimensional (ambiente "E").

obstáculos com passagens estreitas, até o ponto de estagnação se o parâmetro δ é grande com relação à cavidade da passagem estreita. Ainda que o C-RDT não use parâmetro de crescimento, os diversos testes realizados mostram que existe estagnação e nos casos de sucesso as rotas obtidas possuem múltiplas *transições*. Nesta situação o processo de expansão da árvore aumenta o número de vértices, o número de detecções de colisões e portanto o tempo de processamento. No entanto, se o ambiente apresenta passagens estreitas orientados nos eixos do $\mathcal{Q}$, o C-RDT apresenta desempenho similar aos tradicionais RRTs.

Tabela 6.2: Desempenho do RRT básico no ambiente "E"

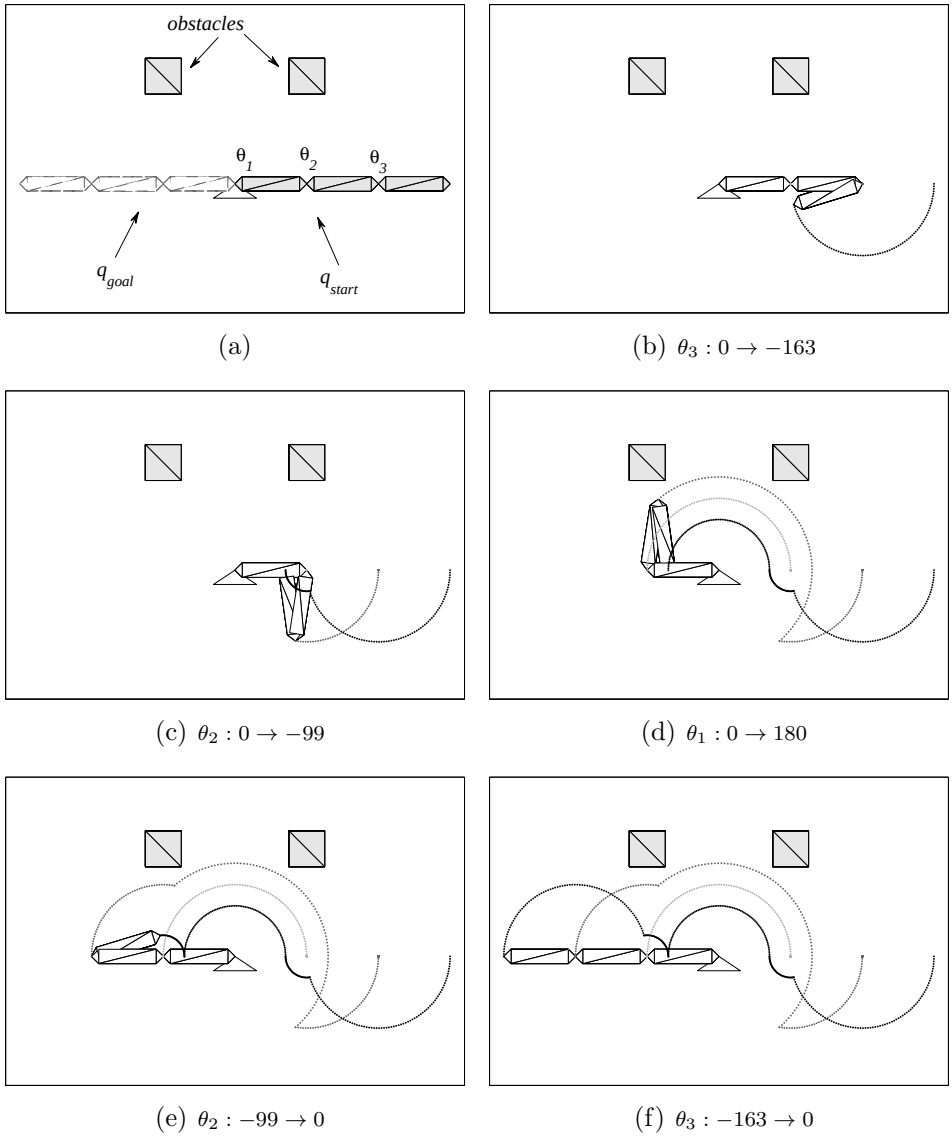
	Iterações	Vértices	Núm. Colisões	Tempo [s]
Média	604.40	422.07	2973.21	45.57
Mín.	106	47	484	2.59
Máx.	2000	1571	7361	99.98
Desv.	303.79	240.37	1137.67	30.87

Tabela 6.3: Desempenho do C-RDT básico no ambiente "E"

	Iterações	Vértices	Núm. Colisões	Tempo [s]
Média	232.47	383.63	2714.50	33.25
Mín.	2	5	261	1.02
Máx.	1897	3238	9561	98.85
Desv.	233.76	395.06	1461.48	22.76

Planejamento dos movimentos para robô planar de 3-GDL

A fim de demonstrar o desempenho do BiCRDT nos robôs seriais, a Fig. 6.29(a) exibe o problema de planejamento de rotas para um robô planar de 3-GDL, o qual precisa se deslocar da configuração $q_{start} = [0\ 0\ 0]^T$ para a configuração $q_{goal} = [180\ 0\ 0]^T$



(a) (b) $\theta_3 : 0 \rightarrow -163$

(c) $\theta_2 : 0 \rightarrow -99$ (d) $\theta_1 : 0 \rightarrow 180$

(e) $\theta_2 : -99 \rightarrow 0$ (f) $\theta_3 : -163 \rightarrow 0$

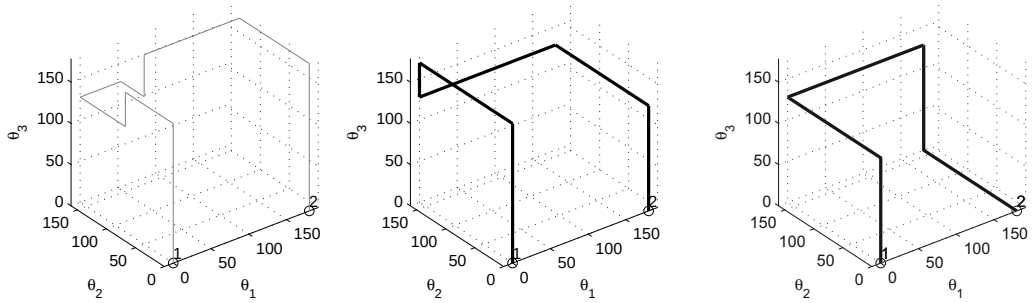
Figura 6.29: Transições obtidas para resolver o problema de planejamento dos movimentos do robô planar de 3-GDL.

usando uma junta por vez. O espaço de trabalho $\mathcal{W} \in \mathbb{R}^2$ contém dois obstáculos quadrados obstruindo os movimentos do robô sob as restrições cinemáticas (6.6).

$$\text{valores permitidos} = \begin{cases} -45 < \theta_1 < 270 \\ -180 < \theta_2 < 180 \\ -180 < \theta_3 < 180 \end{cases} \tag{6.6}$$

As cinco transições da Fig. 6.29(b) até Fig. 6.29(f) mostram o progresso da solução final depois de aplicar o algoritmo BiCRDT e duplo pós-processamento, assim como é indicado na seção 6.3.5.

É claro que os movimentos da solução final foram ilustrados no $\mathcal{W}$, mas, a execução do algoritmo BiCRDT, o crescimento das árvores e as expansões por eixo

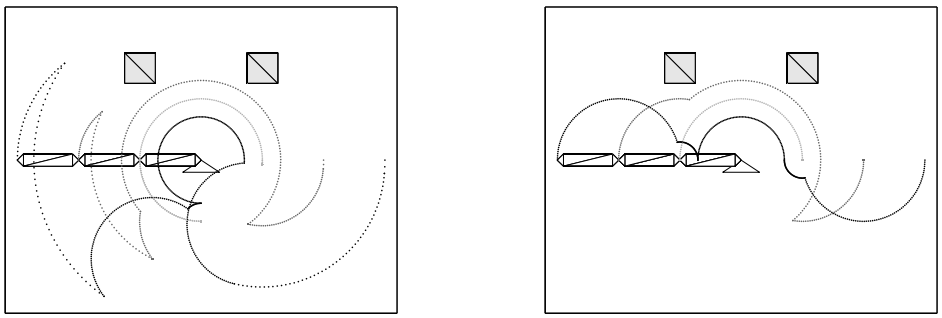


(a) BiCRDT (b) 1o pós-processamento (c) 2o pós-processamento

Figura 6.30: Rotas obtidas no tridimensional espaço de busca para o problema do robô de 3-GDL. Os números 1 e 2 indicam as configurações inicial q_{start} e final q_{goal}.

são realizadas no espaço de configuração $\mathcal{C}$. Bem como o espaço de busca pode ser representado num espaço Euclidiano $\mathcal{C} \in \mathbb{R}^3$ a Fig. 6.30 representa três estados do processo de busca para o problema em questão. Na Fig. 6.30(a) é apresentado a rota solução com dez *transições* que é obtida com o BiCRDT. O primeiro pós-processamento obtém uma rota de seis *transições* que são ilustradas na Fig. 6.30(b). Já a Fig. 6.30(c) mostra as cinco *transições* da rota final encontrada pelo pós-processamento revertido.

Com a representação da Fig. 6.30 é evidente que no caso de um ambiente sem obstáculos o número mínimo de transições para deslocar o robô da configuração q_{start} para a configuração q_{goal} é $c = 1$, tal como é previsto na Eq. 6.5. De modo que, se um obstáculo obstrui os movimentos do robô somente para uma junta, a rota ótima deve conter $c = 3$ *transições*. A Fig. 6.31 apresenta de outra maneira o efeito simplificador da estratégia de pós-processamento, onde é possível perceber os movimentos desnecessários obtidos pelo algoritmo BiCRDT.



(a) BiCRDT (b) 2o pós-processamento

Figura 6.31: Rotas obtidas para resolver o problema do robô planar de 3-GDL.

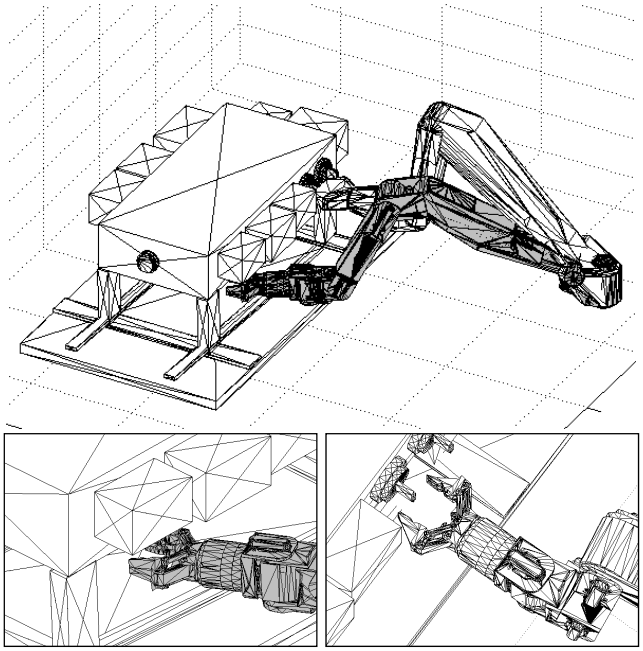


Figura 6.32: Problema de planejamento dos movimentos para manipulador de 6-GDL. *esquerda-abaixo*: condição inicial do efetuador. *Direita-abaixo*: condição final do efetuador.

Planejamento dos movimentos para manipulador de 6-GDL

Para verificar o desempenho da estratégia BiCRDT e seu pós-processamento em espaços de busca multidimensionais a Fig. 6.32 apresenta o caso de estudo da teleoperação do manipulador de 6-GDL TITAN-4. Neste caso deve-se movimentar o braço robótico desde uma configuração onde seu efetuador final encontra-se na parte interior-abaixo de uma Unidade de Distribuição Submarina ou SDU (*Subsea Distribution Unit*) até uma configuração final, onde a posição do efetuador final esteja em frente de um botão de controle. A descrição anterior se encaixa no problema apresentado no esquema da Fig. 6.17. As condições inicial e final estão expressadas em (6.7) e suas restrições cinemáticas em (6.8). (Mais informações no Apêndice A e B).

$$\text{Condições} = \begin{cases} q_{start} = [25\ 19\ \ -82\ 60\ 90\ 0]^T \\ q_{goal} = [10\ 41\ \ -122\ 80\ 80\ 0]^T \end{cases} \tag{6.7}$$

$$\text{Restrições} = \begin{cases} -120 < \theta_1 < 120 \\ -34.19 < \theta_2 < 85.81 \\ -164 < \theta_3 < 106 \\ -90 < \theta_4 < 90 \\ -90 < \theta_5 < 90 \\ -120 < \theta_6 < 120 \end{cases} \tag{6.8}$$

Assumindo que $\mathcal{C}$ tem uma representação num espaço Euclidiano $\mathcal{Q} \in \mathbb{R}^6$, o

comprimento de uma rota l é a soma das distâncias das secções retas e o número de *transições* é $tr = m - 1$, sendo m o número de vértices da rota.

Após aplicar o BiCRDT e as estrategias de pós-processamento, os resultados obtidos são apresentados em (6.9). A solução final com sua sequência de sete *transições* é ilustrada na Fig. 6.33 e Tabela 6.4. Em relação à ótimalidade, quando é usada a expressão (6.5) para as condições (6.7) o numero de alterações é $c = 5$, de maneira que na presença de um único obstáculo o número de *transições* ótimas é $tr > 5$, logo a rota obtida com sete transições ($tr_3 = 7$) é uma solução perto da ótima ideal.

$$\text{Soluções} = \begin{cases} tr_1 = 12, & l_1 = 16.06 & \text{BiCRDT} \\ tr_2 = 9, & l_2 = 7.53 & \text{1o pós-processamento} \\ tr_3 = 7, & l_3 = 3.16 & \text{2o pós-processamento} \end{cases} \tag{6.9}$$

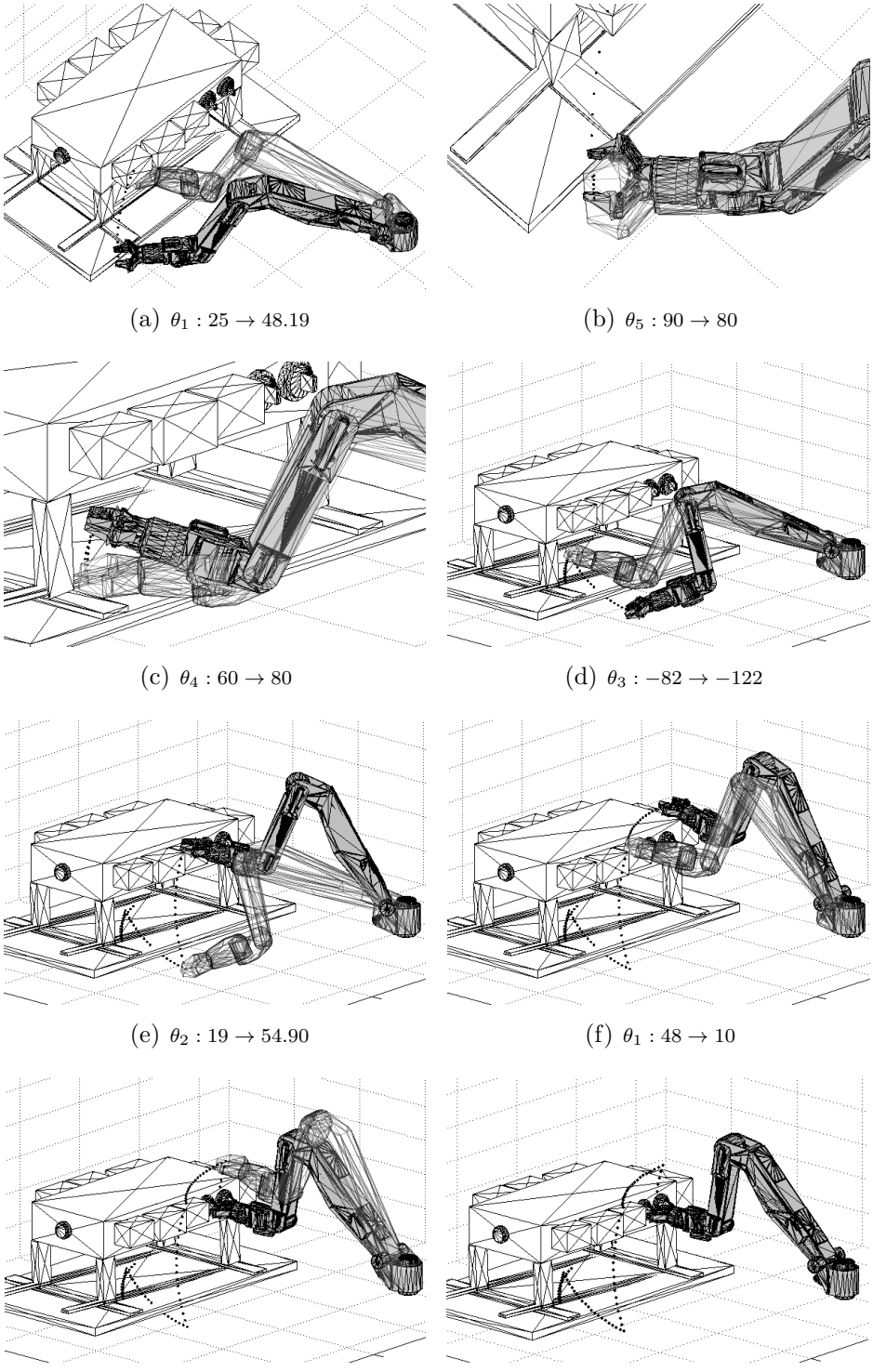
A Tabela 6.4 ressalta as *transições* usando uma junta por vez. Por exemplo, entre o primeiro vértice e o segundo, a transição é realizada desde 25 graus para 48,19 graus no θ_1. De maneira similar acontece na última transição entre 54,90 graus para 41 graus em θ_2. Note-se que não é requerido a rotação da última junta θ_6, isto indica a eficácia do método na redução de movimentos desnecessários.

Tabela 6.4: Rota obtida para o problema do manipulador de 6-GDL

vértice	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	25	19	-82	60	90	0
2	48,19	19	-82	60	90	0
3	48,19	19	-82	60	80	0
4	48,19	19	-82	80	80	0
5	48,19	19	-122	80	80	0
6	48,19	54,90	-122	80	80	0
7	10	54,90	-122	80	80	0
8	10	41	-122	80	80	0

Um exemplo que ilustra a simplificação do número de movimentos necessários para resolver um problema de *Path Planning* é apresentado na Fig. 6.34(a) e 6.34(b). Neste ambiente, o manipulador deve evadir a condição imposta pelos dois prismas e se movimentar para uma configuração final onde é requerido o número mínimo de alterações $c = 1$, em outras palavras, a diferença entre a configuração inicial e final recai no valor da primeira junta θ_1. O pensamento lógico do humano indica que o problema pode ser resolvido movimentando simplesmente as duas primeiras juntas na sequência $\theta_2 \rightarrow \theta_1 \rightarrow \theta_2$, logo a rota ótima tem um número de transições $t = 3$.

Uma comparação dos resultados obtidos com as estrategias RRT-connect e BiCRDT pode ser estabelecida com a Fig. 6.34(c) e 6.34(d), junto com a Tabela 6.5 e 6.6. As duas respostas possuem o mesmo número de vértices, no entanto, a solução com RRT-Connect movimenta todas as juntas para executar uma rota livre de co-



(a) $\theta_1 : 25 \rightarrow 48.19$

(b) $\theta_5 : 90 \rightarrow 80$

(c) $\theta_4 : 60 \rightarrow 80$

(d) $\theta_3 : -82 \rightarrow -122$

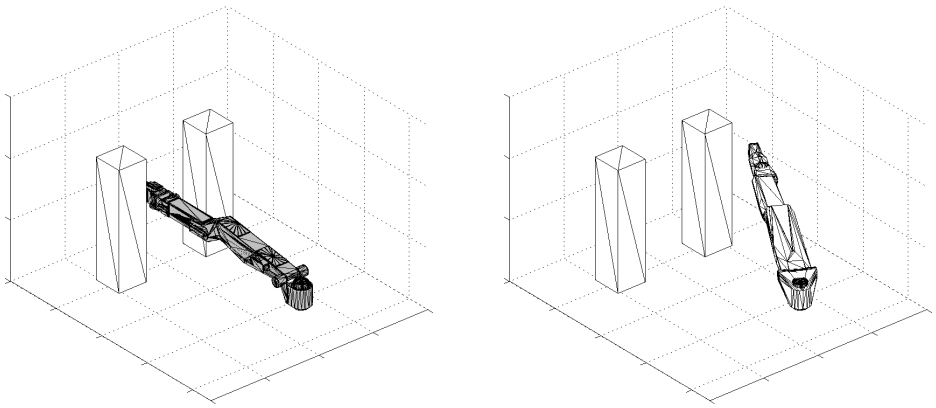
(e) $\theta_2 : 19 \rightarrow 54.90$

(f) $\theta_1 : 48 \rightarrow 10$

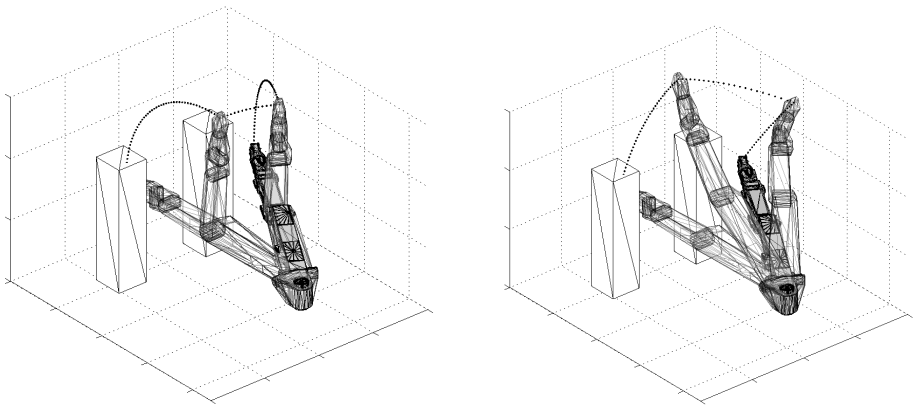
(g) $\theta_2 : 54.90 \rightarrow 41$

(h) Configuração final

Figura 6.33: Conjunto de transições para resolver o problema de planejamento de movimentos do manipulador de 6-GDL.



(a) Configuração inicial (b) configuração final



(c) Resposta BiCRDT (d) Resposta RRT-Connect

Figura 6.34: Desempenho dos algoritmos de planejamento dos movimentos para o manipulador de 6-GDL.

lisões (Ver Tabela 6.6). Por outro lado a estratégia BiCRDT produz uma rota ótima com $t = 3$, movimentando somente a primeira e terceira junta na sequência $\theta_3 \rightarrow \theta_1 \rightarrow \theta_3$ (Ver Tabela 6.5).

Tabela 6.5: Rota obtida pelo BiCRDT correspondente a Fig. 6.34(c).

vértice	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	90	0
2	0	0	95,13	0	90	0
3	-30	0	95,13	0	90	0
4	-30	0	0	0	90	0

Sem dúvida a rota obtida com a estratégia BiCRDT é mais fácil de acompanhar que a rota RRT-Connect e parece mais natural à lógica humana. No entanto, após as 100 vezes que o algoritmo BiCRDT obtém uma rota ótima, 95% das rotas tem a sequência $\theta_3 \rightarrow \theta_1 \rightarrow \theta_3$ da Fig. 6.34(c) e o restante 5% das rotas com a sequência do julgamento humano $\theta_2 \rightarrow \theta_1 \rightarrow \theta_2$. Isto indica que a tendência da estratégia BiCRDT é entregar uma sequência de movimentos com o menor deslocamento de

Tabela 6.6: Rota obtida pelo RRT-Connect correspondente a Fig. 6.34(d).

vértice	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0	0	0	0	90	0
2	-7,73	24,42	18,64	0,64	81,13	-69,05
3	-37,94	15,77	3,761	-0,70	70,10	-43,18
4	-30	0	0	0	90	0

elos.

6.3.7 Considerações e perspectivas

Nesta parte do capítulo foi estudado o problema de acompanhamento de rotas automáticas, expondo uma metodologia para resolver o problema estendido de *Path Planning* onde se precisa encontrar uma sequência de movimentos dos elos do manipulador usando uma junta de cada vez, sem produzir colisões com o ambiente.

A estratégia proposta para resolver esse tipo de problema é chamada de C-RDT e realiza uma exploração incremental no espaço de configuração a maneira de árvore com galhos distribuídos somente nas direções dos eixos do espaço de busca, logo uma simples estratégia heurística de pós-processamento garante a obtenção de rotas de boa qualidade. A versão bidirecional BiCRDT consegue maior precisão da resposta e rapidez na convergência à solução.

Os resultados indicam que o C-RDT fornece soluções rápidas como estratégias RRTS, mas não é necessário um parâmetro de crescimento ou ajustar de qualquer outro parâmetro probabilístico na busca da solução. Além disso, este método é facilmente escalável para espaços de busca multidimensionais como foi apresentado para o estudo de caso.

Apesar da rapidez do BiCRDT e simplicidade do algoritmo de pós-processamento, os diversos experimentos mostram que o tempo do duplo pós-processamento requere um tempo igual ou superior ao empregado pelo algoritmo de busca. O desafio comum para a maioria dos métodos baseados em amostragem é encontrar um método rápido de pós-processamento que garanta soluções ótimas ou quase-ótimas sem comprometer o tempo de computação total. No futuro, devido a natureza discreta das rotas obtidas com BiCRDT, pode pensar-se em desenvolver métodos de pós-processados baseados na programação dinâmica [11]. Outra abordagem futura é incluir funções de custo dentro da fase de exploração do algoritmo C-RDT para obter progressivamente caminhos de melhor qualidade, de maneira semelhante ao enfoque proposto recentemente com o algoritmo RRT* em [180, 181].

Para os manipuladores seriais com juntas rotativas, a estratégia C-RDT consegue entregar uma sequência de movimentos válida para operações tipo *Pick-and-Place*,

no entanto, esta estratégia não consegue proporcionar movimentos lineares do EF exigidas nas tarefas contínuas (como corte ou soldagem), pois com o movimento de uma junta cada vez, resultariam em infinitos pequenos movimentos do EF.

Os métodos C-RDTs foram originalmente idealizados para estabelecer uma arquitetura de controle compartilhado na teleoperação robótica. Não obstante, a redução no número de movimentos obtidos e o fato de deslocar uma junta de cada vez, faz com que as rotas obtidas tenham o potencial de simplificar o controle autônomo dos manipuladores. Isto é, se uma rota exige que se movimentem várias juntas simultaneamente, o controle deve determinar com cálculos dinâmicos iterativos se os atuadores reais conseguem cumprir com as velocidades que a rota demanda num tempo desejado (esse é o conceito de planejamento de trajetórias, ver Fig. 5.12). Por outro lado, uma rota que movimenta uma junta de cada vez, o controle usa cálculos dinâmicos simples, inclusive o cálculo dos perfis de aceleração e velocidade nas juntas podem ser feitos de maneira tradicional com polinômios ou curvas splines [8, 70].

Algumas áreas específicas que podem implantar com sucesso as estrategias aqui mencionadas como sistemas autônomos são: a coordenação de múltiplos braços robóticos, a desmontagem de peças, o controle autônomo dos robôs móveis de exploração, e o controle de manipuladores e robôs moveis usados na mineração.

Capítulo 7

Conclusões e Sugestões

No presente trabalho, foram indicadas estratégias modernas para facilitar a teleoperação de braços robóticos em ambientes não-estruturados. As estratégias estão baseadas na tecnologia da realidade aumentada e nos algoritmos de geração de rotas livres de colisões. Especificamente, o presente trabalho apresentou duas contribuições: a primeira é a formalização dos conceitos da realidade aumentada focada na robótica, apresentando a maneira de estruturar e validar em software as técnicas de auxilio visual que podem ser aplicadas em curto prazo. A segunda é o estabelecimento de novos algoritmos baseados em amostragem que tendem à operação semiautônoma dos manipuladores em aplicações futuras.

O caso de estudo foi a teleoperação de manipuladores que são montados nos ROVs tradicionais. Inicialmente o texto mostrou a complexidade do controle do sistema base-manipulador dos ROVs, onde seu entendimento pode motivar soluções para os problemas expostos e orientar o desenvolvimento de novas pesquisas na área da teleoperação submarina. Além disso, o texto expõe uma análise cinemática do braço robótico TITAN-4, o qual permitiu esclarecer as dificuldades da teleoperação que foram evidenciadas na aplicação da realidade aumentada.

Para o planejamento dos movimentos de um manipulador é necessário perceber o estado dos elementos presentes no ambiente real do robô, os testes realizados mostram que com a realidade aumentada se consegue obter posições e orientações de objetos em relação à câmera, e esses dados podem ser usados para antecipar uma operação do manipulador, mostrando ao usuário o resultado antes da execução real. O aplicativo computacional desenvolvido usa a mesma realidade aumentada para visualizar o resultado inserindo o modelo virtual do manipulador na cena real apresentada numa tela.

As implementações computacionais empregadas neste trabalho em relação com à realidade aumentada foram desenvolvidas em C++, pois esta ferramenta estabelece: a melhor velocidade de processamento, compatibilidade com as bibliotecas de manipulação gráfica e com o sinal de vídeo das câmeras reais. Os algoritmos de geração

de trajetórias foram desenvolvidos na plataforma MATLAB, o que reduziu bastante o esforço na programação focada na engenharia e no ambiente universitário.

O desenvolvimento da aplicação de realidade aumentada foi baseado nas bibliotecas originais ARToolKit por sua disponibilidade do código aberto. Além de explicar a interação dessas bibliotecas com a manipulação de arquivos gráficos 3D, foram apresentadas estratégias para determinar a precisão e exatidão dos dados obtidos com a realidade aumentada empregando-se marcadores de comprimento padrão. Esta metodologia pode ser aprimorada dependendo do tipo de ambiente, como o submarino, do sistema de reconhecimento visual e do equipamento real usado em aplicações reais específicas.

No conteúdo deste texto, a maioria dos pseudocódigos e esquemas de processo foram projetados para o caso de estudo, bem como para sua aplicação em outros contextos onde um manipulador deve operar num ambiente não-estruturado. Similarmente os algoritmos baseados no RRT, descritos no texto, são de caráter geral e podem ser aplicados em outras áreas do planejamento de movimento, tais como: robótica móvel, robótica paralela e sistemas multirrobôs.

Concluiu-se que, usando os algoritmos baseados em amostragem é possível solucionar o problema da cinemática inversa estendida, onde é requerido não somente obter uma configuração válida mais uma sequencia de movimentos livres de colisões para atingir essa configuração desconhecida. O algoritmo proposto consegue lidar satisfatoriamente com as singularidades e a multiplicidade pela redundância posicional do efetuador final. Entretanto, sua busca iterativa requer ajuste de parâmetros para garantir uma convergência entre posição e rotação de uma pose desejada. Outro aspecto observado é o longo tempo requerido para a convergência à resposta, o que inviabiliza sua aplicação de maneira *on-line*. Assim, como proposta para trabalhos futuros, procura-se a mudança do sistema de amostragem e o uso de técnicas de otimização numérica para melhorar os tempos de processamento.

Os resultados obtidos com o método proposto para o planejamento de movimentos em ambientes dinâmicos demonstram a viabilidade do planejamento de rotas dinâmicas baseadas no RRT, desde que o sistema de verificação de colisões seja eficiente e sejam inferidos os obstáculos no espaço de trabalho do robô. Este mesmo novo algoritmo aplicado em ambientes estáticos mostra a capacidade de obter rotas curtas e quase-ótimas sem precisar de métodos de pós-processamento, como é necessário com o original RRT. Em virtude das simulações realizadas com estes algoritmos, podem-se prever aplicações diretas na robótica móvel, como: a navegação autônoma dos chamados *drones* em ambientes dinâmicos desconhecidos, onde obstáculos são reconhecidos e esquecidos usando o range dos sensores abordo. Já para a aplicação nos robôs seriais, os experimentos mostram que os manipuladores sofrem paradas instantâneas na presença de uma colisão iminente, e isto restringe o uso destes

métodos para velocidades baixas de operação.

As estratégias e testes com o software desenvolvido neste trabalho indicam que é possível projetar um sistema robótico real que vincule realidade aumentada com o planejamento de movimentos autônomos, não obstante, é requerido vincular no futuro o modelo dinâmico do robô. Esse requerimento da modelagem dinâmica é próprio de cada aplicação real e precisa também projetar seu sistema de controle real. Uma área interessante de estudo é a abordagem das velocidades das juntas no conceito de espaço de configuração, o qual é até agora um tópico relativamente pouco explorado, e que promete o surgimento de novas maneiras de planejamento de movimento e controle destes sistemas robóticos.

Finalmente foi introduzido formalmente o contexto de como acompanhar as rotas automáticas pelo humano, o qual é um tema negligenciado ou pouco descrito na literatura robótica. Foi apresentado um novo algoritmo que consegue obter rotas livres de colisões que são facilmente seguidas por pilotos humanos no estudo de caso. Os testes mostram que é possível obter rotas quase-ótimas e com desempenho similar as atuais técnicas de geração de rotas baseadas em amostragem. Além disso, fica como tema de pesquisa futura conferir o melhor desempenho destas rotas automáticas, pois o controle por junta independente permite que manipuladores consigam executar os movimentos com cálculos dinâmicos reduzidos e sistemas de controle não iterativos. Consequentemente, no futuro os braços robóticos em ambientes não-estruturados poderiam executar de maneira *on-line* e autônoma os movimentos livres de colisões.

Referências Bibliográficas

[1] GUIZZO, E., DEYLE, T. "Robotics Trends for 2012, [The Future Is Robots]", *IEEE Robot. Automat. Mag. IEEE Robotics & Automation Magazine*, v. 19, n. 1, pp. 119–123.

[2] SICILIANO, B., KHATIB, O. *Springer handbook of robotics*. Berlin, Springer, 2008.

[3] ELVANDER, J., HAWKES, G. "ROVs and AUVs in support of marine renewable technologies". In: *Oceans, 2012*, pp. 1–6.

[4] WHITCOMB, L. L. "Underwater robotics: out of the research laboratory and into the field". In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, v. 1, pp. 709–716 vol.1.

[5] ASSOCIATION, I. M. C. *Code of Practice for the Safe and Efficient Operation of Remotely Operated Vehicles*. International Marine Contractors Association, 2009.

[6] DE LIMA VIEIRA COELHO, L., PAPARELLI, R. "A experiência do trabalhador offshore: o caso de operadores de ROV", *1er Seminário de Saúde do Trabalhador de Franca, Curitiba, Brazil*, v. 97, 2010. Disponível em: <http://www.proceedings.scielo.br/scielo.php?script=sci_arttext&pid=MSC0000000112010000100009&lng=en&nrm=iso>.

[7] JEE-HWAN, R., DONG-SOO, K., PAN-MOOK, L. "Control of underwater manipulators mounted on an ROV using base force information". In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, v. 4, pp. 3238–3243 vol.4, .

[8] SICILIANO, B. *Robotics : modelling, planning and control*. Advanced textbooks in control and signal processing. London, Springer, 2009.

[9] ANGELES, J. *Fundamentals of robotic mechanical systems : theory, methods, and algorithms*. New York ; Berlin, Springer, 2007.

[10] SPONG, M. W., HUTCHINSON, S., VIDYASAGAR, M. *Robot modeling and control*. Hoboken, NJ, John Wiley & Sons, 2006.

[11] LAVALLE, S. M. *Planning algorithms*. Cambridge; New York, Cambridge University Press, 2006.

[12] CHOSET, H. *Principles of robot motion : theory, algorithms, and implementation*. Intelligent robotics and autonomous agents. Cambridge, Mass., MIT Press, 2005.

[13] CHRIST, R. D., WERNLI, R. L. *The ROV manual a user guide to observation-class remotely operated vehicles*. Butterworth-Heinemann, 2007. ISBN: 9780080550169 0080550169 9780750681483 0750681489.

[14] WASIELEWSKI, S., ALDON, M. J. "Dynamic vision for ROV stabilization". In: *OCEANS '96. MTS/IEEE. Prospects for the 21st Century. Conference Proceedings*, v. 3, pp. 1082–1087 vol.3.

[15] LIU, H. T. "A video-based stereoscopic imaging and measurement system (SIMS) for undersea applications". In: *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, v. 1, pp. 275–286 vol.1.

[16] CRAIG, J. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[17] MARCHAND, E., CHAUMETTE, F., SPINDLER, F., et al. "Controlling an uninstrumented ROV manipulator by visual servoing". In: *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, v. 2, pp. 1047–1053 vol.2.

[18] SHENG-WEN, C., HUAI-WEN, H., SHIAO-CHUNG, C. "A dynamic vision system for tracking and localization of underwater objects". In: *Underwater Technology, 2004. UT '04. 2004 International Symposium on*, pp. 215–222.

[19] SHAHRIAR, N., PEZHMAN, F. "An ROV Stereovision System for Ship-Hull Inspection", *Oceanic Engineering, IEEE Journal of*, v. 31, n. 3, pp. 551–564, 2006.

[20] ISHIBASHI, S. "The stereo vision system for an underwater vehicle". In: *OCEANS 2009 - EUROPE*, pp. 1–6.

[21] WHITE, S. N., KIRKWOOD, W., SHERMAN, A., et al. "Development and deployment of a precision underwater positioning system for in situ laser

Raman spectroscopy in the deep ocean", *Deep Sea Research Part I: Oceanographic Research Papers*, v. 52, n. 12, pp. 2376 – 2389, 2005. ISSN: 0967-0637. doi: 10.1016/j.dsr.2005.09.002. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0967063705002128>.

[22] CASALINO, G. "Control and Coordination Problems in Underwater Intervention Robotic Systems: Current Results and Future Challenges". In: *WMR2013: XIV International Conference on Computer Aided Systems Theory Eurocast 2013 and the Marine Robotics Workshop at 2013*, 2013.

[23] PADIR, T. "Kinematic redundancy resolution for two cooperating underwater vehicles with on-board manipulators". In: *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, v. 4, pp. 3137–3142 Vol. 4, 2005. doi: 10.1109/ICSMC.2005.1571628.

[24] SANTOS, C. H. F. D., GUENTHER, R., MARTINS, D., et al. "Virtual kinematic chains to solve the underwater vehicle-manipulator systems redundancy", *J. Braz. Soc. Mech. Sci. & Eng. Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 28, n. 3, 2006.

[25] CASALINO, G., ZEREIK, E., SIMETTI, E., et al. "Agility for underwater floating manipulation: Task amp; subsystem priority based control strategy". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1772–1779, 2012. doi: 10.1109/IROS.2012.6386127.

[26] SOYLU, S., BUCKHAM, B. J., PODHORODESKI, R. P. "A Fault-Tolerant Fuzzy-Logic Based Redundancy Resolution Method for Underwater Mobile Manipulators". In: *OCEANS 2007*, pp. 1–9, .

[27] CHIAVERINI, S. "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators", *Robotics and Automation, IEEE Transactions on*, v. 13, n. 3, pp. 398–410, 1997.

[28] JUN, B.-H., LEE, P.-M., KIM, S. "Manipulability analysis of underwater robotic arms on ROV and application to task-oriented joint configuration", *Journal of Mechanical Science and Technology*, v. 22, n. 5, pp. 887–894, 2008.

[29] ALBIEZ, J., HILDEBRANDT, M., KERDELS, J., et al. "Automatic workspace analysis and vehicle adaptation for hydraulic underwater manipulators". In: *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, pp. 1–6.

[30] HILDEBRANDT, M., CHRISTENSEN, L., KERDELS, J., et al. "Realtime motion compensation for ROV-based tele-operated underwater manipulators". In: *OCEANS 2009 - EUROPE*, pp. 1–6, .

[31] DUNNIGAN, M. W., RUSSELL, G. T. "Reduction of the dynamic coupling between a manipulator and ROV using variable structure control". In: *Control, 1994. Control '94. International Conference on*, v. 2, pp. 1578–1583 vol.2.

[32] ANTONELLI, G. *Underwater Robots: Motion and Force Control of Vehicle-Manipulator Systems (Springer Tracts in Advanced Robotics)*. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2006. ISBN: 354031752X.

[33] MCMILLAN, S., ORIN, D. E., MCGHEE, R. B. "Efficient dynamic simulation of an underwater vehicle with a robotic manipulator", *Systems, Man and Cybernetics, IEEE Transactions on*, v. 25, n. 8, pp. 1194–1206, 1995.

[34] LAPIERRE, L., FRAISSE, P., M'SIRDI, N. K. "Hybrid position/force control of a ROV with a manipulator". In: *OCEANS '98 Conference Proceedings*, v. 2, pp. 931–935 vol.2.

[35] JEE-HWAN, R., DONG-SOO, K., PAN-MOOK, L. "Control of underwater manipulators mounted on an ROV using base force information". In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, v. 4, pp. 3238–3243 vol.4, .

[36] HILDEBRANDT, M., ALBIEZ, J., KIRCHNER, F. "Computer-Based Control of Deep-Sea Manipulators". In: *OCEANS 2008 - MTS/IEEE Kobe Techno-Ocean*, pp. 1–6, .

[37] SOYLU, S., BUCKHAM, B. J., PODHORODESKI, R. P. "Development of a coordinated controller for underwater vehicle-manipulator systems". In: *OCEANS 2008*, pp. 1–9, .

[38] HYUNGWON, S., BONG-HUAN, J., PAN-MOOK, L. "Dynamic workspace control method of underwater manipulator based on motion compensation of an ROV". In: *Underwater Technology (UT), 2011 IEEE Symposium on and 2011 Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC)*, pp. 1–10.

[39] SOYLU, S., BUCKHAM, B. J., PODHORODESKI, R. P. "Dynamics and control of tethered underwater-manipulator systems". In: *OCEANS 2010*, pp. 1–8, .

[40] AGBA, E. I. "SeaMaster: an ROV-manipulator system simulator", *Computer Graphics and Applications, IEEE*, v. 15, n. 1, pp. 24–31, 1995.

[41] OMERDIC, E., TOAL, D. "OceanRINGS: System concept amp; applications". In: *Control Automation (MED), 2012 20th Mediterranean Conference on*, pp. 1391–1396, 2012.

[42] HASHIMOTO, S., I. A. I. M., IGARASHI, T. "TouchMe: An Augmented Reality Based Remote Robot Manipulation". In: *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011*.

[43] CHRISTENSEN, L., KAMPMANN, P., HILDEBRANDT, M., et al. "Hardware ROV simulation facility for the evaluation of novel underwater manipulation techniques". In: *OCEANS 2009 - EUROPE*, pp. 1–8.

[44] FAN, S., LIAN, L., REN, P., et al. "Resistance calculation and motion simulation for deep sea open-framed remotely operated vehicle based on hydrodynamics test". In: *OCEANS, 2012 - Yeosu*, pp. 1–5, 2012.

[45] RIDAO, P., TIANO, A., EL-FAKDI, A., et al. "On the identification of non-linear models of unmanned underwater vehicles", *Control Engineering Practice*, v. 12, n. 12, pp. 1483 – 1499, 2004. ISSN: 0967-0661. doi: 10.1016/j.conengprac.2004.01.004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0967066104000152>. Guidance and control of underwater vehicles.

[46] AVILA, J. P. J., ADAMOWSKI, J. C. "Experimental evaluation of the hydrodynamic coefficients of a {ROV} through Morison's equation", *Ocean Engineering*, v. 38, n. 17–18, pp. 2162 – 2170, 2011. ISSN: 0029-8018. doi: 10.1016/j.oceaneng.2011.09.032. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0029801811002265>.

[47] SAKAGAMI, N., SHIBATA, M., HASHIZUME, H., et al. "Development of a human-sized ROV with dual-arm". In: *OCEANS 2010 IEEE - Sydney*, pp. 1–6, .

[48] SAKAGAMI, N., KANAYAMA, T., UEDA, T., et al. "Design and development of an attitude control system for a human-sized ROV". In: *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pp. 2141–2146, .

[49] CHOI, H.-S., PARK, H., CHUNG, S., et al. "Design and control of a convertible ROV". In: *OCEANS, 2012 - Yeosu*, pp. 1–7, 2012.

[50] NAKAJOH, H., TAKASHI, M., NORIYASU, Y., et al. "Development of Deep Sea ROV "KAIKO7000II"". In: *Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2007. Symposium on*, pp. 7–12, 2007.

[51] BOWEN, A., YOERGER, D., TAYLOR, C., et al. "The Nereus hybrid underwater robotic vehicle for global ocean science operations to 11,000m depth". In: *OCEANS 2008*, pp. 1–10, 2008.

[52] HUTCHINSON, S., HAGER, G., CORKE, P. "A tutorial on visual servo control", *Robotics and Automation, IEEE Transactions on*, v. 12, n. 5, pp. 651–670, 1996. ISSN: 1042-296X. doi: 10.1109/70.538972.

[53] CHAUMETTE, F., HUTCHINSON, S. "Visual servo control Part I: basic approaches", *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2006.

[54] CHAUMETTE, F., HUTCHINSON, S. "Visual Servo Control Part II: Advanced Approaches", *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2007.

[55] ESPIAU, B., CHAUMETTE, F., RIVES, P. "A new approach to visual servoing in robotics", *Robotics and Automation, IEEE Transactions on*, v. 8, n. 3, pp. 313–326, 1992. ISSN: 1042-296X. doi: 10.1109/70.143350.

[56] KRAGIC, D., CHRISTENSEN, H. I. *Survey on Visual Servoing for Manipulation.* Relatório técnico, COMPUTATIONAL VISION AND ACTIVE PERCEPTION LABORATORY, Stockholm, Sweden, 2002.

[57] PALMIERI, G., PALPACELLI, M., BATTISTELLI, M., et al. "A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator", *J. Rob. Journal of Robotics*, v. 2012, 2012.

[58] GARCIA, G. J., CORRALES, J. A., POMARES, J., et al. "Survey of visual and force/tactile control of robots for physical interaction in Spain", *Sensors (Basel, Switzerland)*, v. 9, n. 12, pp. 9689–733, 2009.

[59] DEMENTHON, D., DAVIS, L. "Model-based object pose in 25 lines of code". In: Sandini, G. (Ed.), *Computer Vision — ECCV'92*, v. 588, *Lecture Notes in Computer Science*, cap. 38, pp. 335–343, Springer Berlin Heidelberg, 1992.

[60] ANDREOPOULOS, A., HASLER, S., WERSING, H., et al. "Active 3D Object Localization Using a Humanoid Robot", *Robotics, IEEE Transactions on*,

v. 27, n. 1, pp. 47–64, 2011. ISSN: 1552-3098. doi: 10.1109/TRO.2010.2090058.

[61] TSAI, L.-W. *Robot analysis : the mechanics of serial and parallel manipulators*. New York, Wiley, 1999.

[62] JAZAR, R. N. *Theory of applied robotics kinematics, dynamics, and control*. Springer, 2010. ISBN: 9781441917508 1441917500.

[63] MURRAY, R. M., LI, Z., SASTRY, S. *A mathematical introduction to robotic manipulation*. Boca Raton, CRC Press, 1994.

[64] MOAKHER, M. "Means and Averaging in the Group of Rotations", *SIAM Journal on Matrix Analysis and Applications*, v. 24, n. 1, pp. 1–16, 2002.

[65] RAGHAVAN, M., ROTH, B. "Kinematic analysis of the 6R manipulator of general geometry", *5th Int. Symp. Robot. Res.*, 1990.

[66] MANOCHA, D., CANNY, J. F. "Real Time Inverse Kinematics for General 6R Manipulators". In: *In Proc. IEEE Intern. Conf. Robotics and Automation*, pp. 383–389, 1992.

[67] KOVÁCS, P., HOMMEL, G. "An Extension Theorem for Kinematic Systems of Equations". In: Merlet, J.-P., Ravani, B. (Eds.), *Computational Kinematics '95*, v. 40, *Solid Mechanics and Its Applications*, Springer Netherlands, pp. 31–40, 1995. ISBN: 978-94-010-4147-8. doi: 10.1007/978-94-011-0333-6_4. Disponível em: <http://dx.doi.org/10.1007/978-94-011-0333-6_4>.

[68] TSAI, L. W., MORGAN, A. P. "Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods", *J. of Mech. Trans. Journal of Mechanisms Transmissions and Automation in Design*, v. 107, n. 2, pp. 189, 1985.

[69] WAMPLER, C. W., MORGAN, A. P., SOMMESE, A. J. "Numerical Continuation Methods for Solving Polynomial Systems Arising in Kinematics", *J. Mech. Des. Journal of Mechanical Design*, v. 112, n. 1, pp. 59, 1990.

[70] BIAGIOTTI, L., MELCHIORRI, C. *Trajectory Planning for Automatic Machines and Robots*. Berlin, Heidelberg, Springer Berlin Heidelberg, 2009. ISBN: 978-3-540-85628-3.

[71] LAVALLE, S. "Motion Planning", *Robotics Automation Magazine, IEEE*, v. 18, n. 1, pp. 79–89, 2011. ISSN: 1070-9932. doi: 10.1109/MRA.2011.940276.

[72] LENGYEL, J., REICHERT, M., DONALD, B. R., et al. "Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware". In: *In Proc. SIGGRAPH*, pp. 327–335, 1990.

[73] FOSKEY, M., GARBER, M., LIN, M., et al. "A Voronoi-based hybrid motion planner". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, v. 1, pp. 55–60 vol.1, 2001. doi: 10.1109/IROS.2001.973336.

[74] LOZANO-PEREZ, T. "Automatic Planning of Manipulator Transfer Movements", *Systems, Man and Cybernetics, IEEE Transactions on*, v. 11, n. 10, pp. 681–698, 1981. ISSN: 0018-9472. doi: 10.1109/TSMC.1981.4308589.

[75] LATOMBE, J.-C. *Robot motion planning*. Boston, Kluwer Academic Publishers, 1991.

[76] CÁRDENAS, E., MENDEZ, L., ESMERAL, J. S. "Collision-free path planning in multi-dimensional environments", *Ingeniería e Investigación*, v. 31, n. 2, 2011. ISSN: 2248-8723.

[77] KHATIB, O. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, v. 2, pp. 500–505, 1985.

[78] TAN, G., BERTIER, M., KERMARREC, A.-M. "Visibility-Graph-Based Shortest-Path Geographic Routing in Sensor Networks". In: *INFOCOM 2009, IEEE*, pp. 1719–1727, 2009. doi: 10.1109/INFCOM.2009.5062091.

[79] NGUYET, T. T. N., HOAI, T. V., THI, N. A. "Some Advanced Techniques in Reducing Time for Path Planning Based on Visibility Graph". In: *Knowledge and Systems Engineering (KSE), 2011 Third International Conference on*, pp. 190–194, 2011. doi: 10.1109/KSE.2011.37.

[80] KALUDER, H., BREZAK, M., PETROVIC, I. "A visibility graph based method for path planning in dynamic environments". In: *MIPRO, 2011 Proceedings of the 34th International Convention*, pp. 717–721, 2011.

[81] LI, J.-H., LEE, M.-J., PARK, S.-H., et al. "Real time path planning for a class of torpedo-type AUVs in unknown environment". In: *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES*, pp. 1–6, 2012. doi: 10.1109/AUV.2012.6380728.

[82] SCHOLER, F., LA COUR-HARBO, A., BISGAARD, M. "Generating approximative minimum length paths in 3D for UAVs". In: *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 229–233, 2012. doi: 10.1109/IVS.2012.6232120.

[83] LI, F., ZHANG, J., LETAIEF, K. "Location-aware distributed routing in cognitive radio networks". In: *Communications in China (ICCC), 2012 1st IEEE International Conference on*, pp. 733–738, 2012. doi: 10.1109/ICCChina.2012.6356980.

[84] GAO, Z., JI, L. "Visibility graph analysis of fluid flow signals". In: *Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on*, pp. 44–47, 2012. doi: 10.1109/ICACI.2012.6463119.

[85] PAPADOPOULOU, E. "Net-Aware Critical Area Extraction for Opens in VLSI Circuits Via Higher-Order Voronoi Diagrams", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, v. 30, n. 5, pp. 704–717, 2011. ISSN: 0278-0070. doi: 10.1109/TCAD.2010.2100550.

[86] SUGIHARA, K. "Rescue Boat Voronoi Diagrams for Inhomogeneous, Anisotropic, and Time-Varying Distances". In: *Voronoi Diagrams in Science and Engineering (ISVD), 2011 Eighth International Symposium on*, pp. 91–97, 2011. doi: 10.1109/ISVD.2011.20.

[87] SCHMITT, D., VYATKINA, K. "On Voronoi Diagram in the Line Space and their Generalizations". In: *Voronoi Diagrams in Science and Engineering (ISVD), 2012 Ninth International Symposium on*, pp. 122–125, 2012. doi: 10.1109/ISVD.2012.22.

[88] SAEEDI, S., PAULL, L., TRENTINI, M., et al. "Efficient map merging using a probabilistic generalized Voronoi diagram". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4419–4424, 2012. doi: 10.1109/IROS.2012.6386001.

[89] LINGELBACH, F. "Path planning using probabilistic cell decomposition". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, v. 1, pp. 467–472 Vol.1, 2004. doi: 10.1109/ROBOT.2004.1307193.

[90] ARNEY, T. "An efficient solution to autonomous path planning by Approximate Cell Decomposition". In: *Information and Automation for Sustainability, 2007. ICIAFS 2007. Third International Conference on*, pp. 88–93, 2007. doi: 10.1109/ICIAFS.2007.4544785.

[91] CAI, C., FERRARI, S. "Information-Driven Sensor Path Planning by Approximate Cell Decomposition", *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, v. 39, n. 3, pp. 672–689, 2009. ISSN: 1083-4419. doi: 10.1109/TSMCB.2008.2008561.

[92] SCHEURER, C., ZIMMERMANN, U. "Path planning method for palletizing tasks using workspace cell decomposition". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, 2011. doi: 10.1109/ICRA.2011.5980573.

[93] GALCERAN, E., CARRERAS, M. "Efficient seabed coverage path planning for ASVs and AUVs". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 88–93, 2012. doi: 10.1109/IROS.2012.6385553.

[94] PAULL, L., SAEEDI, S., LI, H., et al. "An information gain based adaptive path planning method for an autonomous underwater vehicle using side-scan sonar". In: *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pp. 835–840, 2010. doi: 10.1109/COASE.2010.5584478.

[95] SHARMA, B., VANUALAILAI, J., SINGH, S. "Lyapunov-based nonlinear controllers for obstacle avoidance with a planar -link doubly nonholonomic manipulator", *Robotics and Autonomous Systems*, v. 60, n. 12, pp. 1484 – 1497, 2012. ISSN: 0921-8890. doi: 10.1016/j.robot.2012.07.014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889012001170>.

[96] LI, G., YAMASHITA, A., ASAMA, H., et al. "An efficient improved artificial potential field based regression search method for robot path planning". In: *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 1227–1232, 2012. doi: 10.1109/ICMA.2012.6283526.

[97] BING, H., GANG, L., JIANG, G., et al. "A route planning method based on improved artificial potential field algorithm". In: *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 550–554, 2011. doi: 10.1109/ICCSN.2011.6014330.

[98] PÊTRÈS, C., ROMERO-RAMIREZ, M.-A., PLUMET, F. "A potential field approach for reactive navigation of autonomous sailboats", *Robotics and Autonomous Systems*, v. 60, n. 12, pp. 1520 – 1527, 2012. ISSN: 0921-8890. doi: 10.1016/j.robot.2012.08.004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889012001364>.

[99] HORNER, D., HEALEY, A. "Use of artificial potential fields for UAV guidance and optimization of WLAN communications". In: *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pp. 88–95, 2004. doi: 10.1109/AUV.2004.1431198.

[100] YANG, Y., WANG, S., WU, Z. L. "Multi-AUV coordination in the underwater environment with obstacles". In: *OCEANS 2010 IEEE - Sydney*, pp. 1–6, 2010. doi: 10.1109/OCEANSSYD.2010.5604019.

[101] CHOIU, R., KAUFMAN, A., LIANG, Z., et al. "An interactive fly-path planning using potential fields and cell decomposition for virtual endoscopy", *Nuclear Science, IEEE Transactions on*, v. 46, n. 4, pp. 1045–1049, 1999. ISSN: 0018-9499. doi: 10.1109/23.790824.

[102] ROSELL, J., INIGUEZ, P. "Path planning using Harmonic Functions and Probabilistic Cell Decomposition". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 1803–1808, 2005. doi: 10.1109/ROBOT.2005.1570375.

[103] GLAVINA, B. "Solving findpath by combination of goal-directed and randomized search". In: *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 1718–1723 vol.3, 1990. doi: 10.1109/ROBOT.1990.126257.

[104] KONDO, K. "Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration", *Robotics and Automation, IEEE Transactions on*, v. 7, n. 3, pp. 267–277, 1991. ISSN: 1042-296X. doi: 10.1109/70.88136.

[105] BARRAQUAND, J., LANGLOIS, B., LATOMBE, J.-C. "Numerical potential field techniques for robot path planning", *Systems, Man and Cybernetics, IEEE Transactions on*, v. 22, n. 2, pp. 224–241, 1992. ISSN: 0018-9472. doi: 10.1109/21.148426.

[106] KAVRAKI, L., SVESTKA, P., LATOMBE, J.-C., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *Robotics and Automation, IEEE Transactions on*, v. 12, n. 4, pp. 566–580, 1996. ISSN: 1042-296X. doi: 10.1109/70.508439.

[107] HSU, D. *Randomized single-query motion planning in expansive spaces*. Tese de Doutorado, Stanford, CA, USA, 2000. AAI9986110.

[108] ADORNO, B. V. *Planejamento Probabilístico de Rotas no Espaço de Configuração e sua Aplicação em Robótica Móvel*. Tese de Mestrado, Universidade de Brasilia, Brasil, 2008.

[109] AL-BLUWI, I. S. T. C. J. "Motion planning algorithms for molecular simulations: A survey", *Computer Science Review Computer Science Review*, v. 6, n. 4, pp. 125–143, 2012.

[110] LE, D., CORTES, J., SIMEON, T. "A path planning approach to (dis)assembly sequencing". In: *Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on*, pp. 286–291, 2009. doi: 10.1109/COASE.2009.5234177.

[111] KUFFNER, J., LAVALLE, S. "RRT-connect: An efficient approach to single-query path planning". In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, v. 2, pp. 995–1001 vol.2, 2000. doi: 10.1109/ROBOT.2000.844730.

[112] LAVALLE, S. M., KUFFNER, J. J. "Randomized kinodynamic planning". In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, v. 1, pp. 473–479 vol.1, 1999. doi: 10.1109/robot.1999.770022.

[113] LAVALLE, S. M., KUFFNER, J. J. "Randomized Kinodynamic Planning", *The International Journal of Robotics Research*, v. 20, n. 5, pp. 378–400, maio 2001. ISSN: 1741-3176. doi: 10.1177/02783640122067453.

[114] ELBANHAWI, M., SIMIC, M. "Sampling-Based Robot Motion Planning: A Review", *Access, IEEE*, v. 2, pp. 56–77, 2014. ISSN: 2169-3536. doi: 10.1109/ACCESS.2014.2302442.

[115] MILGRAM, P., KISHINO, F. "A Taxonomy of Mixed Reality Visual Displays", *IEICE Transactions on Information Systems*, v. E77-D, n. 12, dez. 1994.

[116] NI, T., SCHMIDT, G., STAADT, O., et al. "A Survey of Large High-Resolution Display Technologies, Techniques, and Applications". In: *Virtual Reality Conference, 2006*, pp. 223–236, 2006. doi: 10.1109/VR.2006.20.

[117] HALLER, M. B. M. T. B. *Emerging technologies of augmented reality interfaces and design*. Idea Group Pub., 2007. ISBN: 9781599040684 1599040689.

[118] FURHT, B. S. *Handbook of augmented reality*. Springer, 2011. ISBN: 9781461400646 1461400643.

[119] HUANG, W. A. L. L. M. A. *Human factors in augmented reality environments*. New York, Springer, 2013.

[120] LEE, B., CHUN, J. "Interactive Manipulation of Augmented Objects in Marker-Less AR Using Vision-Based Hand Interaction". In: *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pp. 398–403, 2010. doi: 10.1109/ITNG.2010.36.

[121] ZHOU, F., DUH, H.-L., BILLINGHURST, M. "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR". In: *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, pp. 193–202, 2008. doi: 10.1109/ISMAR.2008.4637362.

[122] KATO, H., BILLINGHURST, M. "Marker tracking and HMD calibration for a video-based augmented reality conferencing system". In: *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, pp. 85–94, 1999. doi: 10.1109/IWAR.1999.803809.

[123] SILTANEN, S. V. T. T. *Theory and applications of marker-based augmented reality*, v. 3. VTT sicence, 2012. ISBN: 9789513874490 9513874494. Disponível em: <http://www.vtt.fi/inf/pdf/science/2012/S3.pdf>.

[124] HUANG, Y., LIU, Y., WANG, Y. "AR-View: An augmented reality device for digital reconstruction of Yuangmingyuan". In: *Mixed and Augmented Reality - Arts, Media and Humanities, 2009. ISMAR-AMH 2009. IEEE International Symposium on*, pp. 3–7, 2009. doi: 10.1109/ISMAR-AMH.2009.5336752.

[125] PYLVANAINEN, T., BERCLAZ, J., KORAH, T., et al. "3D City Modeling from Street-Level Data for Augmented Reality Applications". In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pp. 238–245, 2012. doi: 10.1109/3DIMPVT.2012.19.

[126] KUMARAN, G. S. S. K. R. A. P. M. R. "Impact of Augmented Reality (AR) in Civil Engineering", *AMR Advanced Materials Research*, v. 18-19, pp. 63–68, 2007.

[127] PEÑA-MORA, F., SAVARESE, S., GOLPARVAR-FARD, M. "Automated Model-Based Recognition of Progress Using Daily Construction Photographs and IFC-Based 4D Models". In: *Construction Research Congress 2010*, cap. 5, pp. 51–60, 2010. doi: 10.1061/41109(373)6.

[128] BEHZADAN, A. H., KAMAT, V. R. "Enabling discovery based learning in construction using telepresent augmented reality", *Automation in Construction*, v. 33, n. 0, pp. 3 – 10, 2013. ISSN: 0926-5805. doi: 10.1016/j.autcon.2012.09.003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0926580512001525>.

[129] CHEN, I.-H., MACDONALD, B., WÜNSCHE, B. "Evaluating the Effectiveness of Mixed Reality Simulations for Developing UAV Systems". In: Noda, I., Ando, N., Brugali, D., et al. (Eds.), *Simulation, Modeling, and Programming for Autonomous Robots*, v. 7628, *Lecture Notes in Computer Science*, pp. 388–399, Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-34326-1. doi: 10.1007/978-3-642-34327-8_35. Disponível em: <http://dx.doi.org/10.1007/978-3-642-34327-8_35>.

[130] CHEN, I. Y. H. M. B. W. B. B. G. K. T. "Analysing Mixed Reality Simulation for Industrial Applications: A Case Study in the Development of a Robotic Screw Remover System", *Lecture notes in computer science.*, , n. 6472, pp. 350–361, 2010.

[131] MAIDA, J. C. B. C. K. P. J. "Improving Robotic Operator Performance Using Augmented Reality", *Proceedings of the Human Factors and Ergonomics Society Annual Meeting Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, v. 51, n. 27, pp. 1635–1639, 2007.

[132] REAS, C., BEN, F. *Processing : a programming handbook for visual designers and artists*. Cambridge, Mass., MIT Press, 2007.

[133] REAS, C., BEN, F. *Getting started with Processing*. Beijing; [Cambridge, Mass.], O'Reilly, 2010.

[134] SIMHA, P., SURAJ, K., AHOBALA, T. "Recognition of numbers and position using image processing techniques for solving Sudoku Puzzles". In: *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pp. 1–5, 2012.

[135] HAMZAH, R., ABD GHANI, S., KADMIN, A., et al. "A practical method for camera calibration in stereo vision mobile robot navigation". In: *Research*

and Development (SCOReD), 2012 IEEE Student Conference on, pp. 104–108, 2012. doi: 10.1109/SCOReD.2012.6518620.

[136] NEMEC, B. "Robot grasping using an ArToolKit library = Robotsko prijemanje objektov z uporabo ArToolKit knjiznice", *Elektrotehniski vestnik*, v. 77, n. 1, pp. 69–74, 2010.

[137] FIALA, M. "ARTag, a fiducial marker system using digital techniques". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, v. 2, pp. 590–596 vol. 2, 2005. doi: 10.1109/CVPR.2005.74.

[138] MACINTYRE, B., GANDY, M., DOW, S., et al. "DART: a toolkit for rapid design exploration of augmented reality experiences". In: *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pp. 932–932, New York, NY, USA, 2005. ACM. doi: 10.1145/1186822.1073288. Disponível em: <http://doi.acm.org/10.1145/1186822.1073288>.

[139] LINGTAO, Y., LIXUN, Z., JILIANG, Z. "Research on Clamping Dexterity of Manipulator and Application of Minimally Invasive Surgery Robot". In: *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, pp. 1–4, 2009. doi: 10.1109/IWISA.2009.5073061.

[140] BAEK, Y. M., KOZUKA, Y., SUGITA, N., et al. "Highly precise master-slave robot system for super micro surgery". In: *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pp. 740–745, 2010. doi: 10.1109/BIOROB.2010.5625946.

[141] MA, R., WANG, W., DU, Z., et al. "Design and optimization of manipulator for laparoscopic minimally invasive surgical robotic system". In: *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 598–603, 2012. doi: 10.1109/ICMA.2012.6283175.

[142] OKAMURA, A., SMABY, N., CUTKOSKY, M. "An overview of dexterous manipulation". In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, v. 1, pp. 255–262 vol.1, 2000. doi: 10.1109/ROBOT.2000.844067.

[143] MA, R., DOLLAR, A. "On dexterity and dexterous manipulation". In: *Advanced Robotics (ICAR), 2011 15th International Conference on*, pp. 1–7, 2011. doi: 10.1109/ICAR.2011.6088576.

[144] BULLOCK, I., MA, R., DOLLAR, A. "A Hand-Centric Classification of Human and Robot Dexterous Manipulation", *Haptics, IEEE Transactions on*, v. 6, n. 2, pp. 129–144, 2013. ISSN: 1939-1412. doi: 10.1109/TOH.2012.53.

[145] DOLLAR, A. M., HOWE, R. D. "The Highly Adaptive SDM Hand: Design and Performance Evaluation", *The International Journal of Robotics Research*, v. 29, n. 5, pp. 585–597, 2010. doi: 10.1177/0278364909360852.

[146] AMEND, J., BROWN, E., RODENBERG, N., et al. "A Positive Pressure Universal Gripper Based on the Jamming of Granular Material", *Robotics, IEEE Transactions on*, v. 28, n. 2, pp. 341–350, 2012. ISSN: 1552-3098. doi: 10.1109/TRO.2011.2171093.

[147] JIANG, A., BIMBO, J., GOULDER, S., et al. "Adaptive grip control on an uncertain object". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1161–1166, 2012. doi: 10.1109/IROS.2012.6385922.

[148] FIGLIOLA, R. S., BEASLEY, D. E. *Theory and design for mechanical measurements*. Hoboken, NJ, Wiley, 2011.

[149] NAKANISHI, J., CORY, R., MISTRY, M., et al. "Comparative experiments on task space control with redundancy resolution". In: *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3901–3908.

[150] SHIMIZU, M., WOO-KEUN, Y., KITAGAKI, K. "A Practical Redundancy Resolution for 7 DOF Redundant Manipulators with Joint Limits". In: *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4510–4516.

[151] DAESUNG, J., YOUNGJUN, Y., JAHOO, K., et al. "A novel redundancy resolution method to avoid joint limits and obstacles on anthropomorphic manipulator". In: *SICE Annual Conference (SICE), 2011 Proceedings of*, pp. 924–929.

[152] COLOME, A., TORRAS, C., TH, I. R. S. J. I. C. O. R., et al. "Redundant inverse kinematics: Experimental comparative review and two enhancements", *IEEE Int Conf Intell Rob Syst IEEE International Conference on Intelligent Robots and Systems*, pp. 5333–5340, 2012.

[153] KISHIMOTO, T., FUJIMOTO, Y. "Numerically stable inverse kinematics calculation of robot manipulators even in singular configuration". In: *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, v. 2, pp. 1036–1040 Vol. 2.

[154] GONZALEZ, C., BLANCO, D., MORENO, L. "A memetic approach to the inverse kinematics problem". In: *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 180–185.

[155] MULLER, R., MUKUNDAN, R. "Triangualation - A New Algorithm for Inverse Kinematics". University of Canterbury. Computer Science and Software Engineering., 2007.

[156] HUYNH, D. "Metrics for 3D rotations: Comparison and analysis", *Journal of Mathematical Imaging and Vision*, v. 35, n. 2, pp. 155–164, 2009.

[157] HUDSON, T. C., LIN, M. C., COHEN, J., et al. "V-COLLIDE: accelerated collision detection for VRML", pp. 119–125, 1997. 253472 117-ff.

[158] BERTRAM, D. "An integrated approach to inverse kinematics and path planning for redundant manipulators". In: *in Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'06*, pp. 1874–1879, 2006.

[159] VAHRENKAMP, N., BERENSON, D., ASFOUR, T., et al. "Humanoid motion planning for dual-arm manipulation and re-grasping tasks". In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 2464–2470, 2009. doi: 10.1109/IROS.2009.5354625.

[160] VAHRENKAMP, N., DO, M., ASFOUR, T., et al. "Integrated Grasp and motion planning". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2883–2888, 2010. doi: 10.1109/ROBOT.2010.5509377.

[161] WANG, W., YAN, L. "Path planning for redundant manipulator without explicit inverse kinematics solution". In: *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pp. 1918–1923, 2009. doi: 10.1109/ROBIO.2009.5420547.

[162] ARISTIDOU, A., LASENBY, J. "FABRIK: A fast, iterative solver for the Inverse Kinematics problem", *Graphical Models*, v. 73, n. 5, pp. 243 – 260, 2011. ISSN: 1524-0703. doi: http://dx.doi.org/10.1016/j.gmod.2011.05.003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1524070311000178>.

[163] HSU, D., LATOMBE, J.-C., MOTWANI, R. "Path planning in expansive configuration spaces". In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, v. 3, pp. 2719–2726 vol.3, 1997. doi: 10.1109/ROBOT.1997.619371.

[164] LAVALLE, S. M., KUFFNER, J. J., JR. "Rapidly-Exploring Random Trees: Progress and Prospects". In: *Algorithmic and Computational Robotics: New Directions*, pp. 293–308, 2000.

[165] LAUMOND, J.-P. "Kineo CAM: a success story of motion planning algorithms", *Robotics Automation Magazine, IEEE*, v. 13, n. 2, pp. 90–93, 2006. ISSN: 1070-9932. doi: 10.1109/MRA.2006.1638020.

[166] YOSHIDA, E., ESTEVES, C., BELOUSOV, I., et al. "Planning 3-D Collision-Free Dynamic Robotic Motion Through Iterative Reshaping", *Robotics, IEEE Transactions on*, v. 24, n. 5, pp. 1186–1198, 2008. ISSN: 1552-3098. doi: 10.1109/TRO.2008.2002312.

[167] BRUCE, J., VELOSO, M. "Real-time randomized path planning for robot navigation". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, v. 3, pp. 2383–2388 vol.3, 2002. doi: 10.1109/IRDS.2002.1041624.

[168] FERGUSON, D., KALRA, N., STENTZ, A. "Replanning with RRTs". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1243–1248, 2006. doi: 10.1109/ROBOT.2006.1641879.

[169] ZUCKER, M., KUFFNER, J., BRANICKY, M. "Multipartite RRTs for Rapid Replanning in Dynamic Environments". In: *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1603–1609, 2007. doi: 10.1109/ROBOT.2007.363553.

[170] BARRIGA, N., SOLAR, M., ARAYA-LÓPEZ, M. "Combining a Probabilistic Sampling Technique and Simple Heuristics to Solve the Dynamic Path Planning Problem". In: *Chilean Computer Science Society (SCCC), 2009 International Conference of the*, pp. 43–50, 2009. doi: 10.1109/SCCC.2009.11.

[171] TSAI, Y.-C., HUANG, H.-P. "Motion planning of a dual-arm mobile robot in the configuration-time space". In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 2458–2463, 2009. doi: 10.1109/IROS.2009.5354154.

[172] NIETO, J., SLAWINSKI, E., MUT, V., et al. "Online path planning based on Rapidly-Exploring Random Trees". In: *Industrial Technology (ICIT), 2010 IEEE International Conference on*, pp. 1451–1456, 2010. doi: 10.1109/ICIT.2010.5472492.

[173] AOUDE, G., LUDERS, B., JOSEPH, J., et al. "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns", *Autonomous Robots*, v. 35, n. 1, pp. 51–76, 2013. ISSN: 0929-5593. doi: 10.1007/s10514-013-9334-3. Disponível em: <http://dx.doi.org/10.1007/s10514-013-9334-3>.

[174] SHAN, E., DAI, B., SONG, J., et al. "A Dynamic RRT Path Planning Algorithm Based on B-Spline". In: *Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on*, v. 2, pp. 25–29, 2009. doi: 10.1109/ISCID.2009.155.

[175] KUWATA, Y., KARAMAN, S., TEO, J., et al. "Real-Time Motion Planning With Applications to Autonomous Urban Driving", *Control Systems Technology, IEEE Transactions on*, v. 17, n. 5, pp. 1105–1118, 2009. ISSN: 1063-6536. doi: 10.1109/TCST.2008.2012116.

[176] JU, T., LIU, S., YANG, J., et al. "Dynamic path planning in robot-aided optical manipulation of biological cells". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 652–657, 2012. doi: 10.1109/IROS.2012.6385831.

[177] LIU, H., SUN, Q., ZHANG, T. "Hierarchical RRT for humanoid robot footstep planning with multiple constraints in complex environments". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 3187–3194, 2012. doi: 10.1109/IROS.2012.6385836.

[178] FRAGKOPOULOS, C., GRAESER, A. "A RRT based path planning algorithm for Rehabilitation robots". In: *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8, 2010.

[179] AHMADYAN, S. N., KUMAR, J. A., VASUDEVAN, S. "Runtime verification of nonlinear analog circuits using incremental Time-augmented RRT algorithm". In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pp. 21–26, 2013. doi: 10.7873/DATE.2013.019.

[180] KARAMAN, S., FRAZZOLI, E. "Incremental Sampling-based Algorithms for Optimal Motion Planning", *CoRR*, v. abs/1005.0416, 2010. Disponível em: <http://arxiv.org/abs/1005.0416>.

[181] KARAMAN, S., WALTER, M., PEREZ, A., et al. "Anytime Motion Planning using the RRT*". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1478–1483, May 2011. doi: 10.1109/ICRA.2011.5980479.

[182] CRANOR, L. F. "A Framework for Reasoning About the Human in the Loop". In: *1st Conference on Usability, Psychology, and Security, ser. UPSEC'08*, p. 1:1–1:15. Berkeley, CA, USA: USENIX Association, 2008.

[183] BORTOLAMI, S., DUDA, K., BORER, N. "Markov analysis of human-in-the-loop system performance". In: *Aerospace Conference, 2010 IEEE*, pp. 1–9, March 2010. doi: 10.1109/AERO.2010.5446860.

[184] KIM, D.-J., BEHAL, A. "Human-in-the-loop control of an assistive robotic arm in unstructured environments for spinal cord injured users". In: *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, pp. 285–286, March 2009.

[185] PENCE, W., FARELO, F., ALQASEMI, R., et al. "Visual servoing control of a 9-DoF WMRA to perform ADL tasks". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 916–922, May 2012. doi: 10.1109/ICRA.2012.6225162.

[186] FARELO, F., ALQASEMI, R., DUBEY, R. "Task-oriented control of a 9-DoF WMRA system for opening a spring-loaded door task". In: *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pp. 1–6, June 2011. doi: 10.1109/ICORR.2011.5975484.

[187] DOPICO, D., LUACES, A., GONZALEZ, M., et al. "Dealing with multiple contacts in a human-in-the-loop application", *Multibody System Dynamics*, v. 25, n. 2, pp. 167–183, 2011. ISSN: 1384-5640. doi: 10.1007/s11044-010-9230-y. Disponível em: <http://dx.doi.org/10.1007/s11044-010-9230-y>.

[188] JEON, S., JANG, M., LEE, D., et al. "Control architecture for heterogeneous multiple robots with human-in-the-loop". In: *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, pp. 274–278, Nov 2012. doi: 10.1109/URAI.2012.6462993.

[189] LEEPER, A., HSIAO, K., CIOCARLIE, M., et al. "Strategies for human-in-the-loop robotic grasping". In: *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp. 1–8, March 2012.

[190] BRINGES, C., LIN, Y., SUN, Y., et al. "Determining the benefit of human input in human-in-the-loop robotic systems". In: *RO-MAN, 2013 IEEE*, pp. 210–215, Aug 2013. doi: 10.1109/ROMAN.2013.6628447.

[191] KHOKAR, K., ALQASEMI, R., SARKAR, S., et al. "A novel telerobotic method for human-in-the-loop assisted grasping based on intention recognition". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 4762–4769, May 2014. doi: 10.1109/ICRA.2014.6907556.

[192] ZHANG, L., LI, Y., ZHANG, P., et al. "A teleoperation system for mobile robot with whole viewpoints virtual scene for situation awareness". In: *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pp. 627–632, Feb 2009. doi: 10.1109/ROBIO.2009.4913074.

[193] COTE, S., TRUDEL, P. "Third person perspective augmented reality for high accuracy applications". In: *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pp. 247–248, Oct 2013. doi: 10.1109/ISMAR.2013.6671788.

[194] COELHO BRAGA DE OLIVEIRA, D., ANDRADE CAETANO, F., DE SOUZA DA SILVA, R. "A Method to Automate the Calibration of a Multiple Fiducial Marker Setup". In: *Virtual and Augmented Reality (SVR), 2014 XVI Symposium on*, pp. 89–95, May 2014. doi: 10.1109/SVR.2014.22.

[195] DHIMAN, V., RYDE, J., CORSO, J. "Mutual localization: Two camera relative 6-DOF pose estimation from reciprocal fiducial observation". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1347–1354, Nov 2013. doi: 10.1109/IROS.2013.6696524.

[196] YANG, J., WEN ZHU, F., PENG YUAN, Z., et al. "OpenSURF Algorithm Improvement for Augmented Reality Based on the Natural Features". In: *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on*, v. 1, pp. 62–65, Aug 2012. doi: 10.1109/IHMSC.2012.21.

[197] KOCH, C., NEGES, M., KÖNIG, M., et al. "Natural markers for augmented reality-based indoor navigation and facility maintenance", *Automation in*

Construction, v. 48, n. 0, pp. 18 – 30, 2014. ISSN: 0926-5805. doi: http://dx.doi.org/10.1016/j.autcon.2014.08.009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0926580514001885>.

[198] LEE, S., LIM, Y., CHUN, J. "3D interaction in Augmented Reality with stereo-vision technique". In: *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, pp. 401–405, Jan 2013.

[199] HOSSEINI, A., BACARA, D., LIENKAMP, M. "A system design for automotive augmented reality using stereo night vision". In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 127–133, June 2014. doi: 10.1109/IVS.2014.6856484.

[200] MORALES, R., KEITLER, P., MAIER, P., et al. "An Underwater Augmented Reality system for commercial diving operations". In: *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, pp. 1–8, Oct 2009.

[201] ISHIDA, M., SHIMONOMURA, K. "Marker based camera pose estimation for underwater robots". In: *System Integration (SII), 2012 IEEE/SICE International Symposium on*, pp. 629–634, Dec 2012. doi: 10.1109/SII.2012.6427353.

[202] BELLARBI, A., DOMINGUES, C., OTMANE, S., et al. "Augmented reality for underwater activities with the use of the DOLPHYN". In: *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on*, pp. 409–412, April 2013. doi: 10.1109/ICNSC.2013.6548773.

[203] TZAFESTAS, C. "Teleplanning by human demonstration for VR-based teleoperation of a mobile robotic assistant". In: *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pp. 462–467, 2001. doi: 10.1109/ROMAN.2001.981947.

[204] BELGHITH, K., NKAMBOU, R., KABANZA, F., et al. "An Intelligent Simulator for Telerobotics Training", *Learning Technologies, IEEE Transactions on*, v. 5, n. 1, pp. 11–19, First 2012. ISSN: 1939-1382. doi: 10.1109/TLT.2011.19.

Apêndice A

Especificações do TITAN-4

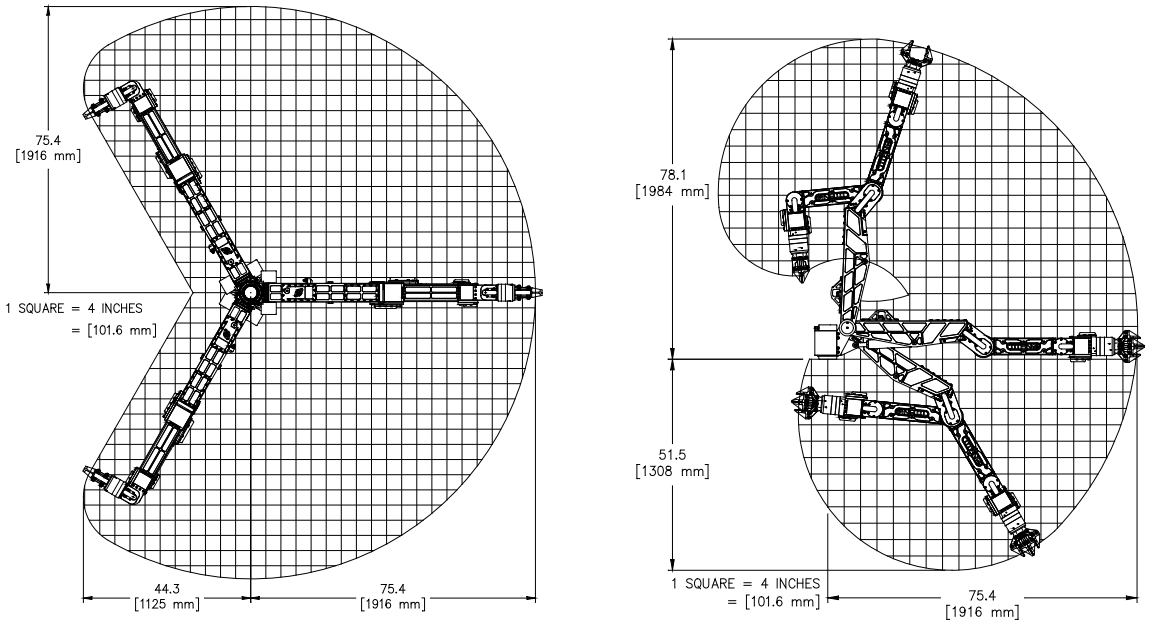


Figura A.1: Espaço de trabalho do TITAN-4. Amplitude dos movimentos, na esquerda vista de teto e na direita vista lateral.

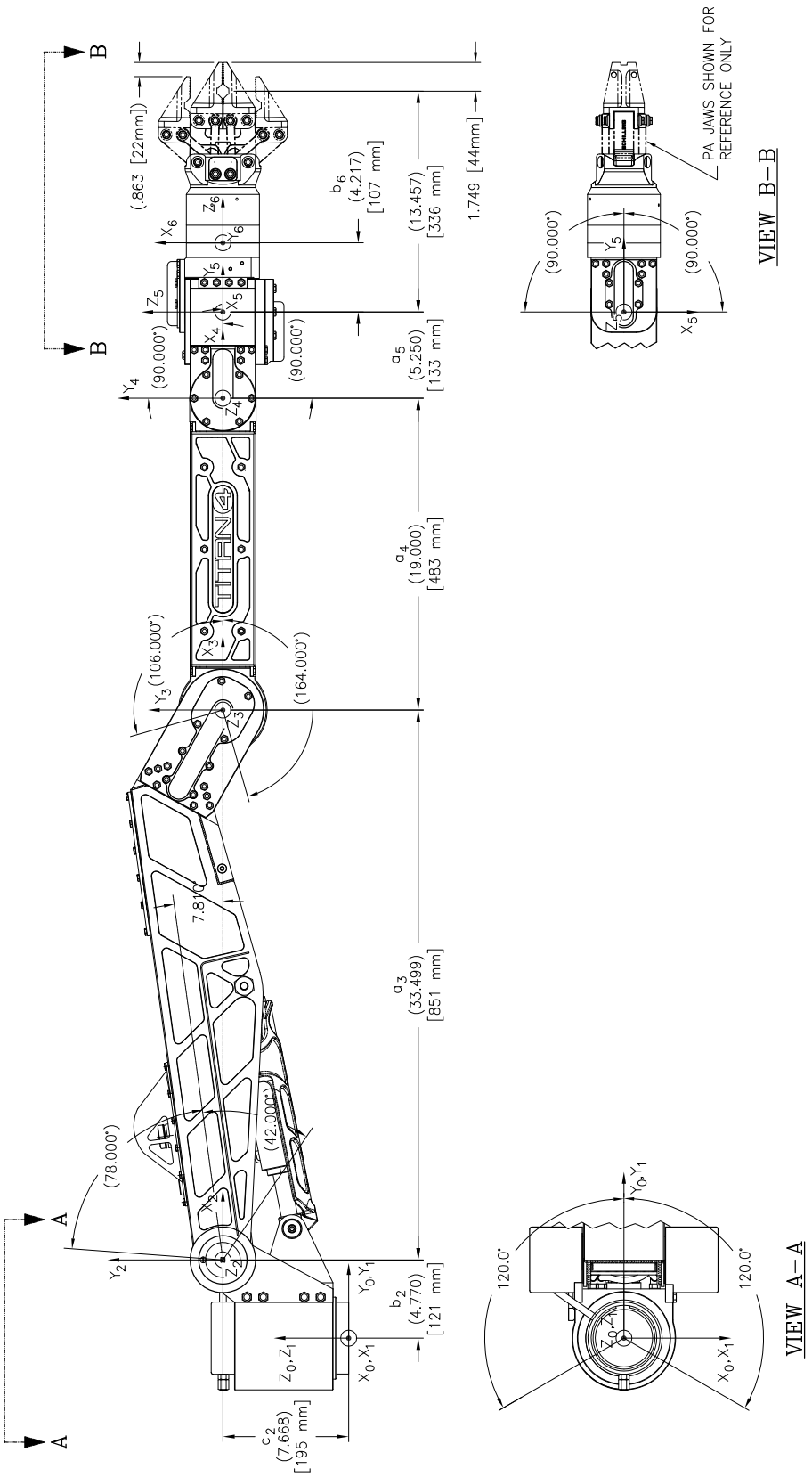


Figura A.2: Configuração em sobreposição de ângulos das juntas em zero.

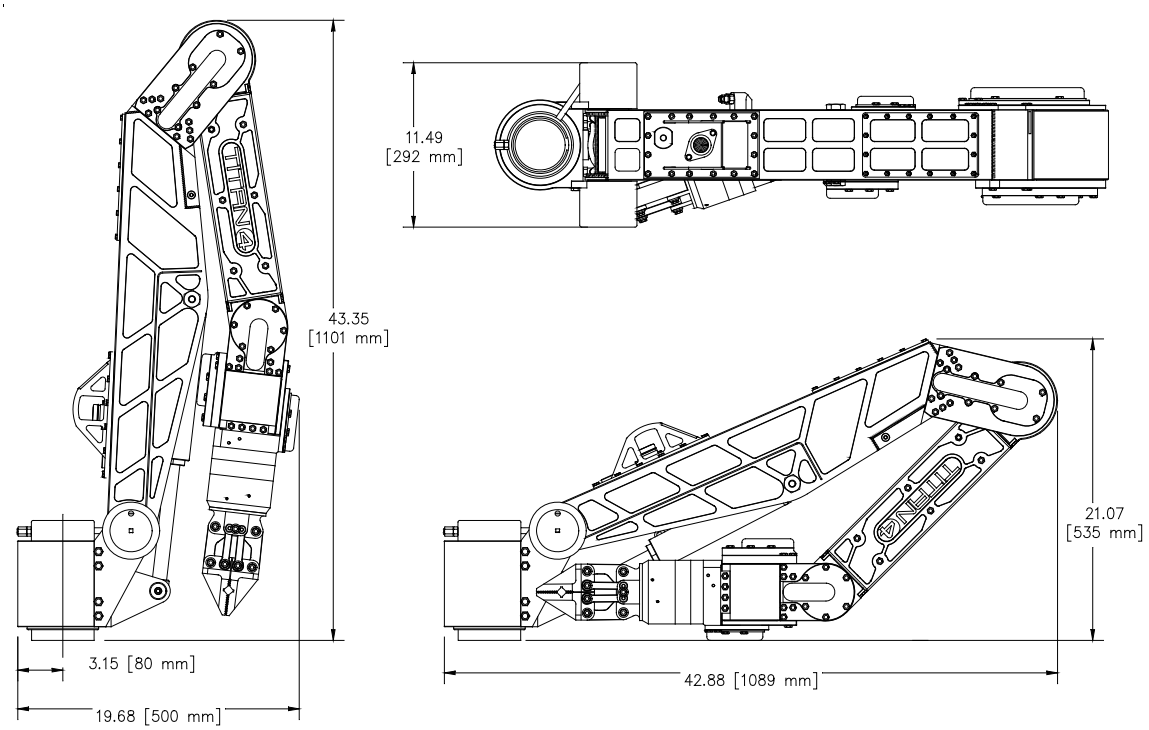


Figura A.3: Cumprimentos de armazenado.

Apêndice B

Descrição da cinemática direta do TITAN-4

B.1 Parâmetros DH

Na Fig. B.1 mostra os sistemas de coordenadas de cada junta usando a notação Denavit-Hartenberg Estândar. Neste caso os subíndices não correspondem à respectiva junta, como é o caso na notação Modificada.

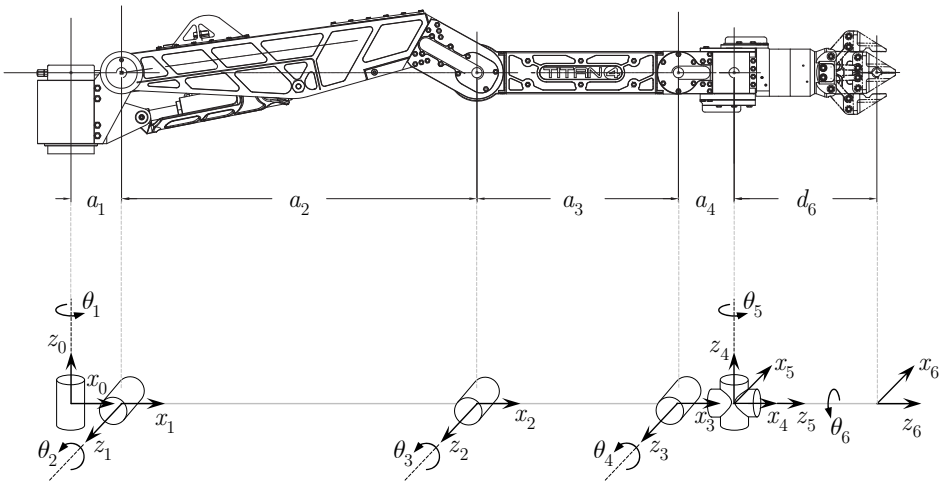


Figura B.1: Descrição do sistema de coordenadas do manipulator TITAN-4.

Usando a Tabela B.1 podem ser obtidas as matrizes de transforação homogêneas para descrever qualquer ponto do robô no espaço de trabalho. Especialmente estas matrizes são usadas para descrever a posição e orientação do efetuador final dado um vetor de configuração $\mathbf{q} = [\theta_1, \theta_2, \ldots, \theta_6]^T$.

Logo, as matrizes de transformação homogênea que vinculam um sistema de referencia em relação com seu sistema imediatamente anterior $T_(i-1)(i)$ são:

Tabela B.1: Matriz de parâmetros Denavith-Hartenberg (padrão) do TITAN-4.

i	a_i	α_i	d_i	θ_i
1	a_1	$\pi/2$	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3
4	a_4	$-\pi/2$	0	θ_4
5	0	$\pi/2$	0	$\theta_5 + \pi/2$
6	0	0	d_6	θ_6

$$T_{01} = \begin{bmatrix} c_1 & 0 & s_1 & a_1c_1 \\ s_1 & 0 & -c_1 & a_1s_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

$$T_{12} = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.2}$$

$$T_{23} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

$$T_{34} = \begin{bmatrix} c_4 & 0 & -s_4 & a_4c_4 \\ s_4 & 0 & c_4 & a_4s_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

$$T_{45} = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.5}$$

$$T_{56} = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.6}$$

Sendo,

$$\begin{aligned} s_1 &= \sin(\theta_1) & c_1 &= \cos(\theta_1) \\ s_2 &= \sin(\theta_2) & c_2 &= \cos(\theta_2) \\ s_3 &= \sin(\theta_3) & c_3 &= \cos(\theta_3) \\ s_4 &= \sin(\theta_4) & c_4 &= \cos(\theta_4) \\ s_5 &= \sin(\theta_5 + \pi/2) & c_5 &= \cos(\theta_5 + \pi/2) \\ s_6 &= \sin(\theta_6) & c_6 &= \cos(\theta_5) \end{aligned}$$

Percebe-se que de acordo com o método Denavit-Hartenberg a variável θ_5 é o ângulo medido desde o eixo x_4 até x_5. Para um caso real, onde o valor de θ_5 é medido entre os eixos x_4 e z_5 (regra da mão direita), simplesmente pode usar $s_5 = \sin(\theta_5)$ e $c_5 = \cos(\theta_5)$.

B.2 Matriz de Transformação para o EF

Usando álgebra de transformações homogêneas $T_{06} = T_{01}T_{12}T_{23}T_{34}T_{45}T_{56}$ temos que a matriz que indica a posição e orientação do efetuador final é $T_e = T_{06}$.

Se T_e é definida como:

$$T_e(\mathbf{q}) = \begin{bmatrix} R_e(\mathbf{q}) & P_e(\mathbf{q}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{B.7}$$

é sendo:

$$R_e = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{B.8}$$

$$P_e = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{B.9}$$

Portanto, seus elementos utilizando $T_e = T_{06}$ são:

$$
\begin{aligned}
r_{11} &= -c_6\left(s_1 s_5 - c_1 c_5 c_{234}\right) - c_1 s_6 s_{234} \\
r_{12} &= s_6\left(s_1 s_5 - c_1 c_5 c_{234}\right) - c_1 c_6 s_{234} \\
r_{13} &= c_5 s_1 + c_1 c_{234} s_5 \\
r_{21} &= c_6\left(c_1 s_5 + c_5 c_{234} s_1\right) - s_1 s_6 s_{234} \\
r_{22} &= -s_6\left(c_1 s_5 + c_5 c_{234} s_1\right) - c_6 s_1 s_{234} \\
r_{23} &= c_{234} s_1 s_5 - c_1 c_5 \\
r_{31} &= c_{234} s_6 + c_5 c_6 s_{234} \\
r_{32} &= c_6 c_{234} - c_5 s_6 s_{234} \\
r_{33} &= s_5 s_{234} \\
p_x &= a_1 c_1 + d_6\left(c_5 s_1 + c_1 c_{234} s_5\right) + a_2 c_1 c_2 - a_3 c_1 s_2 s_3 - a_4 c_1 s_4 s_{23} + a_3 c_1 c_2 c_3 + a_4 c_1 c_4 c_{23} \\
p_y &= a_1 s_1 - c_1 c_5 d_6 + a_2 c_2 s_1 + c_{234} d_6 s_1 s_5 - a_3 s_1 s_2 s_3 - a_4 s_1 s_4 s_{23} + a_3 c_2 c_3 s_1 + a_4 c_4 c_{23} s_1 \\
p_z &= a_2 s_2 + a_3 s_{23} + a_4 s_{234} + d_6 s_5 s_{234}
\end{aligned}
$$

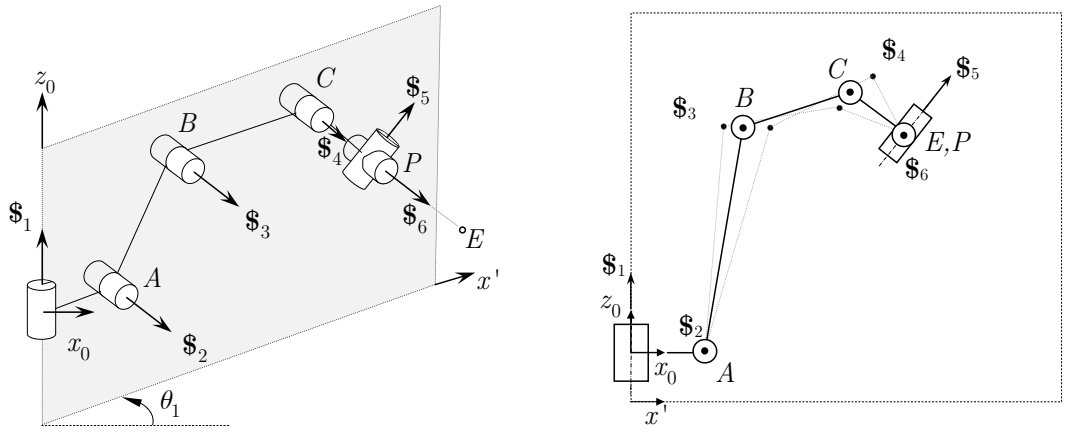
Sendo $s_{ij} = \sin(\theta_i + \theta_j)$, $c_{ij} = \cos(\theta_i + \theta_j)$
e $s_{ijk} = \sin(\theta_i + \theta_j + \theta_k)$, $c_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$
para $i, j, k = \{1, 2, 3, 4\}$.

Apêndice C

Uma Solução Geométrica do Caso de Singularidade do TITAN-4

C.1 Descrição da singularidade

As condições singulares do manipulador TITAN-4 foram apresentadas na seção 3.3.2, uma das condições singulares sem solução fechada é quando $\theta_5 = \pi/2$ ou $\theta_5 = -\pi/2$. Nesta condição o eixo da última junta (eixo screw $\$_6$) é perpendicular ao plano do manipulador z_0x' (Fig. C.1(a)). Como $\$_6$ é paralelo aos eixos $\$_2$, $\$_3$ e $\$_4$, os elos AB, BC e CP podem se movimentar como um mecanismo de quatro barras, quando eles não são colineares. Percebe-se que os pontos C e B podem se movimentar no plano do robô, em oposição as posições fixas de A e P para que o EF fique estático (Fig. C.1(b)).



(a) (b)

Figura C.1: Representação do TITAN-4 com multiplicidade da cinemática inversa. (a) Vista em projeção. (b) Representação linear, os eixos screw $ estão saindo do plano, entre A, B, C e P é formado um mecanismo de quatro barras.

C.2 Solução do caso

Dentre as diversas configurações válidas, tem-se duas soluções para o caso singular $\cos(\theta_5) = 0$ são apresentadas na Fig. C.2, onde $\$_5$ tem a mesma orientação que o eixo $\$_1$, isto é, $\$_5 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

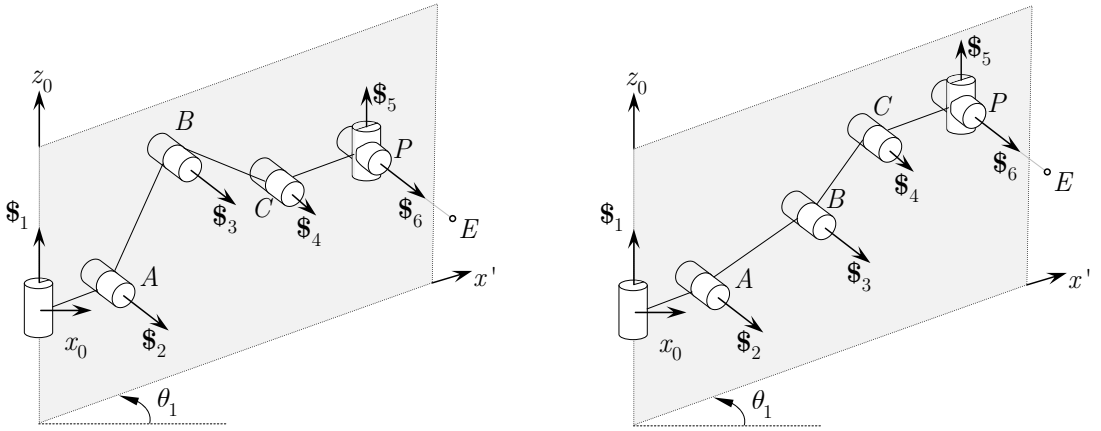


Figura C.2: Duas soluções da cinemática inversa pelo método geométrico. (a) Cotovelo para acima. (b) Cotovelo para abaixo.

As duas soluções são obtidas de maneira geométrica usando a análise da Fig. C.3. Do triângulo ABC tem-se:

$$b^2 = {a_2}^2 + {a_3}^2 - 2a_2a_3\cos(\beta) \tag{C.1}$$

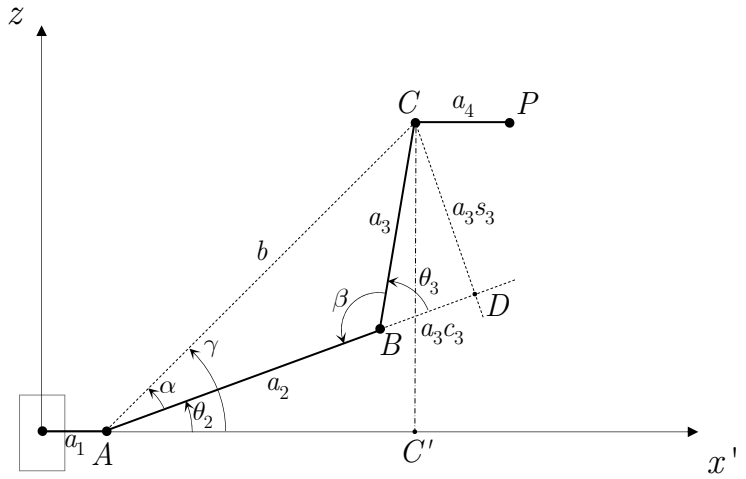


Figura C.3: Caracterização do TITAN-4 em seu plano de robô zx'

Da identidade trigonométrica $\cos(\pi - \theta_3) = -cos$ e como $\beta = \pi - \theta_3$, então $-\cos(\beta) = \cos(\theta_3)$. Nota-se que $s_i = \sin(\theta_i)$ e $c_i = \cos(\theta_i)$

Cálculo de θ_3:

Como $-\cos(\beta) = \cos(\theta_3)$, a Eq. C.1 pode ser rescrita assim:

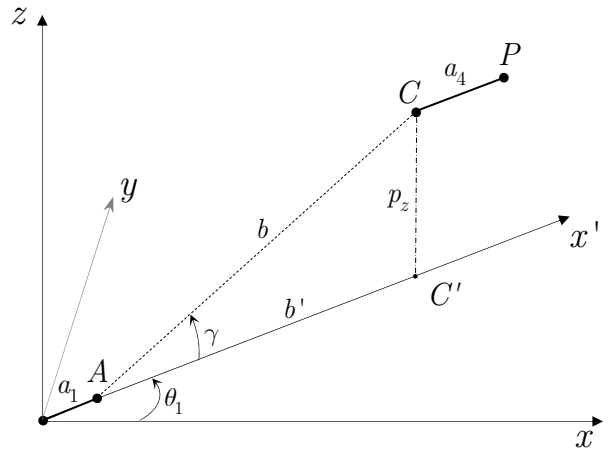
$$b^2 = {a_2}^2 + {a_3}^2 + 2a_2a_3\cos(\theta_3) \tag{C.2}$$

Portanto, conseguimos duas soluções de θ_3

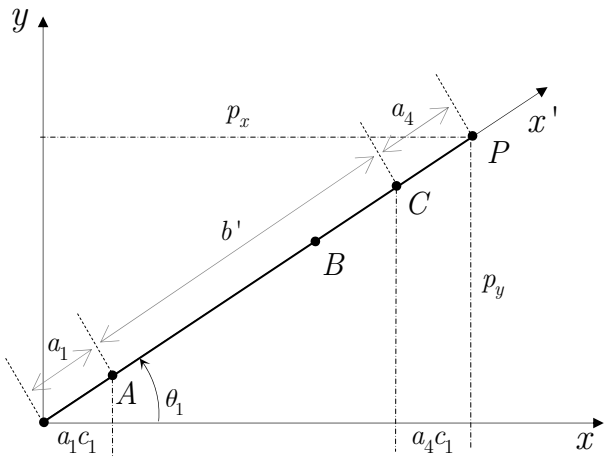
$$\theta_3 = \text{acos}\left(\frac{b^2 - {a_2}^2 - {a_3}^2}{2a_2a_3}\right) \tag{C.3}$$

O valor de b^2 é obtido usando o triângulo ACC' (Fig. C.3 e Fig. C.4(a)) :

$$b^2 = {b'}^2 + {p_z}^2 \tag{C.4}$$



(a)



(b)

Figura C.4: Representação linear do TITAN-4 para o cálculo da cinemática inversa no caso de multiplicidade.

Usando a Fig. C.4(a) temos:

$$(b')^2 = (p_x - a_1c_1 - a_4c_1)^2 + (p_y - a_1s_1 - a_4s_1)^2 \tag{C.5}$$

Substituindo a Eq. C.5 em Eq. C.4 obtemos:

$$b^2 = (p_x - a_1c_1 - a_4c_1)^2 + (p_y - a_1s_1 - a_4s_1)^2 + {p_z}^2 \tag{C.6}$$

Cálculo de θ_2**:**
Do triângulo ADC da Fig. C.3:

$$\tan(\alpha) = \frac{a_3s_3}{a_2 + a_3c_3} \quad \Rightarrow \quad \alpha = \text{atan2}(a_3s_3, a_2 + a_3c_3) \tag{C.7}$$

Do triângulo ACC' da Fig. C.3:

$$\tan(\gamma) = \frac{p_z}{p_x - a_1c_1 - a_4c_1} \quad \Rightarrow \quad \gamma = \text{atan2}(pz, p_x - a_1c_1 - a_4c_1) \tag{C.8}$$

Como $\gamma = \alpha + \theta_2$, então $\theta_2 = \gamma - \alpha$.

Cálculo de θ_4**:**

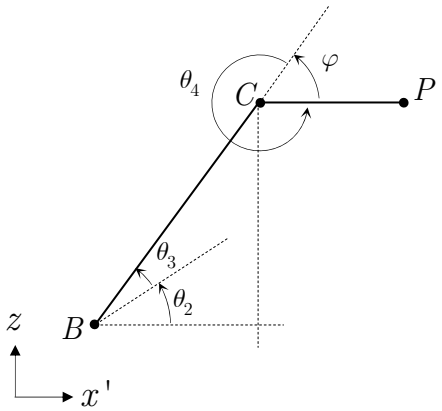


Figura C.5: Análise dos elos BC e CP para o cálculo do θ_4

Da análise da Fig. C.5, sendo $\theta_4 = 2\pi - \varphi$ e $\varphi = \theta_2 + \theta 3$.
Portanto: $\theta_4 = -(\theta_2 + \theta 3)$.

O cálculo dos restantes ângulos é feito da mesma maneira como é apresentado na seção 3.2.