

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA MECÂNICA**

Carlos Rodrigues Rocha

**PLANEJAMENTO DE MOVIMENTO DE SISTEMAS
ROBÓTICOS DE INTERVENÇÃO SUBAQUÁTICA
BASEADO NA TEORIA DOS HELICOIDES**

Florianópolis

2012

Carlos Rodrigues Rocha

**PLANEJAMENTO DE MOVIMENTO DE SISTEMAS
ROBÓTICOS DE INTERVENÇÃO SUBAQUÁTICA
BASEADO NA TEORIA DOS HELICOIDES**

Tese submetida ao Programa de Pós-Graduação em Engenharia Mecânica para a obtenção do Grau de Doutor em Engenharia, Especialidade Engenharia Mecânica.

Orientador: Prof. D.Sc. Altamir Dias

Florianópolis

2012

Catálogo na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

R672p Rocha, Carlos Rodrigues
Planejamento de movimento de sistemas robóticos de
intervenção subaquática baseado na teoria dos helicoides
[tese] / Carlos Rodrigues Rocha ; orientador, Altamir Dias. -
Florianópolis, SC, 2012.
279 p.: il., grafs., tabs.

Tese (doutorado) - Universidade Federal de Santa Catarina,
Centro Tecnológico. Programa de Pós-Graduação em Engenharia
Mecânica.

Inclui referências

1. Engenharia mecânica. 2. Robótica. 3. Cinemática. 4.
Modelagem computacional. I. Dias, Altamir. II. Universidade
Federal de Santa Catarina. Programa de Pós-Graduação em
Engenharia Mecânica. III. Título.

CDU 621

Carlos Rodrigues Rocha

**PLANEJAMENTO DE MOVIMENTO DE SISTEMAS
ROBÓTICOS DE INTERVENÇÃO SUBAQUÁTICA
BASEADO NA TEORIA DOS HELICOIDES**

Esta Tese foi julgada adequada para a obtenção do Título de “Doutor em Engenharia, Especialidade Engenharia Mecânica”, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Mecânica.

Florianópolis, 16 de abril 2012.

Prof. Dr. Júlio César Passos
Coordenador do Curso

Banca Examinadora:

Prof. D.Sc. Altamir Dias
Presidente

Prof. Dr.Eng. Marcelo Becker
Relator

Prof. Dr. Sebastião Cícero Pinheiro Gomes

Prof. Dr. Edson Roberto de Pieri

Prof. Dr.Eng. Daniel Martins

Prof. Dr.Eng. Henrique Simas

À Patrícia, Marina e Júlia, as mulheres
da minha vida.

AGRADECIMENTOS

Ao Prof. Altamir Dias, pela orientação, dedicação e paciência. Agradeço também pela oportunidade de observar e aprender mais sobre as vocações de docente e pesquisador com alguém que mostra tanto empenho e gosto pelo que faz. E, claro, agradeço as boas conversas sobre diversos assuntos, tanto relativos a este trabalho quanto ao software livre, CAD e cotidiano.

À minha esposa Patrícia e minhas filhas Marina e Júlia pela compreensão, suporte e apoio nesse longo período em que me afastei delas para realizar este trabalho. Não há como exprimir em palavras o que sinto por elas e a gratidão por todo o sacrifício que fizeram por mim.

Aos Professores Daniel Martins e Henrique Simas, pelas contribuições e formação que resultaram neste trabalho, além da amizade desenvolvida/retomada ao longo desses anos.

À Cristiane Tonetto, colega e amiga, pelas contribuições neste e em outros trabalhos; pela amizade e o convívio que contribuíram para tornar suportável a experiência de morar longe da família.

Aos demais colegas do LabCADCAM e do Laboratório de Robótica Prof. Raul Guenther pelo convívio e a troca de experiências.

Aos bolsistas do LabCADCAM e do Laboratório de Robótica que contribuíram para o desenvolvimento deste trabalho.

Aos colegas da Universidade Federal do Rio Grande (FURG) e do Campus Rio Grande do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) que deram suporte ao meu afastamento para a realização deste curso e apoio para a sua conclusão.

Aos demais colegas e amigos de Florianópolis e do Rio Grande pelo apoio, amizade e incentivo, de diversas formas, para minimizar as saudades de casa enquanto estive morando em Floripa.

Aos meus pais, Júlia e Luiz Carlos, que nunca pouparam esforços para possibilitar a minha busca por conhecimento e a minha formação.

À Deus, que motiva a busca pelo aperfeiçoamento, dá a força necessária para atingirmos as nossas metas e ampara nos momentos difíceis.

RESUMO

Os *veículos subaquáticos não tripulados* (ou UUV, do inglês *Unmanned Underwater Vehicles*) são responsáveis pela execução de grande parte das operações em ambientes imersos. Os *sistemas veículo-manipulador subaquáticos* (ou UVMS, do inglês *Underwater Vehicle-Manipulator Systems*) são UUV voltados para a execução de tarefas de intervenção. Além de aplicações em missões científicas e de resgate, os UVMS são muito usados em instalações *offshore* de extração/distribuição de petróleo e gás em tarefas de construção, manutenção, inspeção e operação. A maioria dos sistemas de intervenção subaquática é teleoperada devido às dificuldades de operação no ambiente imerso e às características cinemáticas e dinâmicas dos UVMS. A evolução desses *sistemas de intervenção subaquática* envolve o desenvolvimento de sua autonomia. Um requisito básico para isso é a capacidade do sistema planejar as ações necessárias para realizar as tarefas a ele especificadas. Esta tese estuda o *planejamento de movimento* dos UVMS durante a execução de tarefas de intervenção. Este problema consiste em definir os movimentos que o sistema (veículo e manipuladores) deve realizar para executar as tarefas especificadas atendendo às restrições impostas pelo espaço de trabalho. O trabalho utiliza a análise cinemática baseada na teoria dos helicoides, teoria dos grafos e ferramentas derivadas para definir modelos cinemáticos dos UVMS em diferentes cenários de execução de tarefas de intervenção. A cooperação entre manipuladores de um mesmo UVMS e entre dois ou mais UVMS é estudada, assim como a variabilidade dos modelos cinemáticos em função de mudanças no contexto da operação. A partir da análise realizada, define-se uma sistematização da modelagem cinemática dos sistemas de intervenção por componentização, visando facilitar e automatizar esse processo. Um *framework* computacional é projetado para facilitar a implementação desses modelos. Com base nesses resultados, define-se uma estrutura geral para o desenvolvimento de estratégias de planejamento de movimento. Simulações de uso dessa estrutura em diferentes cenários de operação são apresentadas. Assim, este trabalho contribui para a autonomia de UUV/UVMS, considerada o principal objeto de pesquisa da área e que no caso dos sistemas de intervenção subaquática reduzirá custos de operação, além de possibilitar o uso destes em novas missões. **Palavras-chave:** Sistemas subaquáticos de intervenção. Teoria dos helicoides. Modelagem cinemática. Tarefas de intervenção.

ABSTRACT

Unmanned Underwater Vehicles (UUV, for short) are used in most immerse operations. *Underwater Vehicle-Manipulator Systems* (UVMS, for short) are a particular kind of UUV designed for intervention tasks. Besides their application in scientific and rescue missions, UVMS are much used in offshore oil and gas extraction/distribution facilities for construction, maintenance, inspection and operation tasks. Most underwater intervention systems are teleoperated due the operational difficulties in the immerse environment and the UVMS kinematic/dynamic features. The evolution of these *underwater intervention systems* involves the development of their autonomy. The system ability to plan the necessary actions to perform its assigned tasks is a basic requirement for that. This thesis studies the *motion planning* of UVMS while executing intervention tasks. The problem consists of defining the motion that the system (vehicle and manipulators) must do to execute the specified tasks while complying with the workspace imposed restrictions. A computational framework is designed to aid the implementation of these models. A general structure to the developed of motion planning strategies based on these results is defined. Simulations using this strucute in different operation scenarios are presented. So, this work contributes to the autonomy of UUV/UVMS, which is considered a major research field and it will reduce operation costs of underwater intervention systems, besides allowing their use in new missions.

Keywords: Underwater intervention systems. Screw theory. Kinematic modeling. Intervention tasks.

LISTA DE FIGURAS

Figura 1	ROV Oceaneering Millenium conectando um duto em um painel (Oceaneering, 2009)	37
Figura 2	O robô ENDURANCE (Stone Aerospace, 2012).....	39
Figura 3	AUV Remus 6000 da Hydroid (Kongsberg, 2011)	39
Figura 4	Cybernetix ALIVE, UVMS com capacidade de operação autônoma ou teleoperada (Cybernetix, 2008)	40
Figura 5	UVMS em operação: (a)sítio do vazamento de petróleo no Golfo do México(Leff; Plushnick-Masti, 2010); (b)Remora 6000 sendo lançado(Phoenix, 2011)	41
Figura 6	Projeções de expansão do uso de UUV: (a)AUV; (b)ROV de classes de serviço(Douglas-Westwood, 2011)	42
Figura 7	Relação entre UUV e operações subaquáticas	47
Figura 8	Visão atualizada da classificação da robótica subaquática	47
Figura 9	Diagrama de blocos de atividades de um UVMS em fases que não envolvem intervenção	53
Figura 10	Diagrama de blocos de atividades de um UVMS em fases que envolvem intervenção	54
Figura 11	Subsistemas de um UVMS.....	56
Figura 12	Subsistemas de comando de um UUV.....	57
Figura 13	Subsistemas de execução de um UUV.....	61
Figura 14	Subsistemas de estrutura de um UUV	64
Figura 15	Definição da postura do veículo.....	72
Figura 16	Representação cinemática de um UVMS e seus sistemas de coordenadas.....	75
Figura 17	Definição do heligiro para o veículo subaquático	78
Figura 18	Cadeia cinemática do sistema de intervenção subaquática	81
Figura 19	Cadeia do UVMS fechada pela cadeia virtual de posição	81
Figura 20	Dígrafo de movimento do UVMS	82
Figura 21	Grafo contraído de movimento do sistema subaquático de intervenção formado por um veículo e um manipulador	86
Figura 22	Grafo contraído de movimento do UVMS com dois manipuladores independentes.....	87
Figura 23	Grafo contraído de movimento do UVMS com dois manipuladores executando uma tarefa de forma cooperativa	88

Figura 24 Grafo contraído de movimento do sistema de intervenção subaquática cooperativo	90
Figura 25 Grafo contraído de movimento do sistema de intervenção subaquática cooperativo	91
Figura 26 Grafo contraído de movimento de um UVMS operando em presença de obstáculos.....	93
Figura 27 Grafo contraído de movimento de um UVMS fixo à uma estrutura sobre a qual executará uma tarefa.....	95
Figura 28 Diagrama de estados de operação de um UVMS - reparticionamento.....	101
Figura 29 Diagrama de estados de operação de um UVMS - reconfiguração.....	101
Figura 30 Reconfiguração de cadeia cinemática pela adição de um UVMS: (a)antes da adição; (b)após a adição.....	103
Figura 31 Diagrama de classes do <i>framework</i> Kast.....	111
Figura 32 Representação gráfica de estruturas de dados usadas em KCComponent.....	114
Figura 33 Atividades de UVMS envolvidas nas estratégias de planejamento de movimento.....	123
Figura 34 Diagrama de classes do <i>framework</i> de intervenção.....	128
Figura 35 Diagrama de comunicação da resolução cinemática pela classe Guidance.....	133
Figura 36 Fluxo de informações no sistema de intervenção planar	135
Figura 37 Cenário de simulação de um UVMS com um manipulador	142
Figura 38 Movimento do efetuador final.....	144
Figura 39 Erro nas variáveis do efetuador final.....	144
Figura 40 Movimento do UVMS durante a execução da tarefa....	145
Figura 41 Variáveis do manipulador.....	146
Figura 42 Variáveis do manipulador.....	147
Figura 43 Variáveis do veículo.....	148
Figura 44 Variáveis do manipulador.....	150
Figura 45 Variáveis do manipulador.....	152
Figura 46 Variáveis do veículo.....	153
Figura 47 Variáveis do manipulador.....	154
Figura 48 Variáveis do veículo.....	155
Figura 49 Trajetória a ser percorrida na execução da tarefa.....	156

Figura 50	Variáveis do manipulador	157
Figura 51	Variáveis do veículo	158
Figura 52	Cenário de simulação de um UVMS com dois manipula- dores em cooperação	159
Figura 53	Movimento da peça e dos efetuadores finais	161
Figura 54	Movimento do UVMS durante a execução da tarefa	161
Figura 55	Postura dos efetuadores finais ao longo do tempo	162
Figura 56	Variáveis do manipulador 1	163
Figura 57	Variáveis do manipulador 2	164
Figura 58	Cenário de simulação de dois UVMS em cooperação	165
Figura 59	Movimento da peça e dos efetuadores finais	167
Figura 60	Movimento dos UVMS durante a execução da tarefa	167
Figura 61	Postura dos efetuadores finais ao longo do tempo	168
Figura 62	Variáveis do manipulador 1	169
Figura 63	Variáveis do manipulador 2	170
Figura 64	Movimento da peça e dos efetuadores finais	171
Figura 65	Translação dos veículos durante a execução da tarefa	172
Figura 66	Variáveis do manipulador do UVMS 1	173
Figura 67	Variáveis do veículo do UVMS 1	174
Figura 68	Variáveis do manipulador do UVMS 2	175
Figura 69	Variáveis do veículo do UVMS 2	176
Figura 70	Cenário de simulação de dois UVMS com dois manipu- ladores cada em cooperação	177
Figura 71	Movimento da peça e dos efetuadores finais	179
Figura 72	Movimento dos UVMS durante a execução da tarefa	179
Figura 73	Postura dos efetuadores finais 11 e 12 ao longo do tempo	180
Figura 74	Postura dos efetuadores finais 21 e 22 ao longo do tempo	181
Figura 75	Variáveis do manipulador 11	182
Figura 76	Variáveis do manipulador 12	183
Figura 77	Variáveis do manipulador 21	184
Figura 78	Variáveis do manipulador 22	185
Figura 79	Definição geométrica das coordenadas de Plücker	208
Figura 80	Definição geométrica de um helicóide	209
Figura 81	Deslocamento helicoidal e parâmetros de Rodrigues	210
Figura 82	Heligiro em um corpo rígido	211

Figura 83	Transformação de helicoides	214
Figura 84	Representação de um par cinemático por um grafo	215
Figura 85	Mecanismo de 4 barras: (a)representação funcional; (b)grafo de movimento	216
Figura 86	Cadeias virtuais de Assur	220
Figura 87	Manipulador planar PPR: (a)representação funcional; (b)configuração de referência	250
Figura 88	Manipulador planar RRR: (a)representação funcional; (b)configuração de referência	253
Figura 89	UVMS planar com 1 manipulador	256
Figura 90	UVMS planar com 2 manipuladores	258
Figura 91	Execução de tarefas com um UVMS planar com um manipulador: (a)representação funcional; (b)grafo de movimento	261
Figura 92	UVMS planar com dois manipuladores trabalhando em cooperação: (a)representação funcional; (b)grafo de movimento	263
Figura 93	Execução de tarefa cooperativa entre dois UVMS planares: (a)representação funcional; (b)grafo de movimento	266
Figura 94	Representação funcional do cenário de cooperação entre dois UVMS planares com dois manipuladores cada	269
Figura 95	Grafo de movimento do cenário de cooperação entre dois UVMS planares com dois manipuladores cada	270

LISTA DE TABELAS

Tabela 1	Classes de ROV	42
Tabela 2	Usos de ROV	42
Tabela 3	Classes de AUV	43
Tabela 4	Usos de AUV	43
Tabela 5	Atividades que empregam UUV	45
Tabela 6	Sensores empregados nos sistemas de navegação de UUV	60
Tabela 7	Notação SNAME para o movimento de veículos aquáticos	72
Tabela 8	Ambientes de computação numérica avaliados para a implementação do <i>framework Kast</i>	229
Tabela 9	Plataformas de desenvolvimento de software avaliadas para a implementação do <i>framework Kast</i>	230
Tabela 10	Variáveis e atributos da cadeia PPR	249
Tabela 11	Parâmetros de Rodrigues	250
Tabela 12	Parâmetros dos helicoides normalizados	251
Tabela 13	Variáveis e atributos da cadeia RRR	252
Tabela 14	Parâmetros de Rodrigues	253
Tabela 15	Parâmetros dos helicoides normalizados	255
Tabela 16	Variáveis e atributos do UVMS planar com um manipulador	256
Tabela 17	Variáveis e atributos do UVMS planar com dois manipuladores	259

LISTA DE ABREVIATURAS E SIGLAS

UUV	<i>Unmanned Underwater Vehicle</i> , ou Veículo Subaquático Não Tripulado
ROV	<i>Remotely Operated Vehicle</i> , ou Veículo Remotamente Operado
AUV	<i>Autonomous Underwater Vehicle</i> , ou Veículo Subaquático Autônomo
UVMS	<i>Underwater Vehicle-Manipulator System</i> , ou Sistema Veículo-Manipulador Subaquático
I-AUV	<i>Intervention AUV</i> , ou AUV de intervenção
MOOS	<i>Motion-Oriented Operating System</i>
GPS	<i>Global Positioning System</i> , ou Sistema de Posicionamento Global
TCP/IP	<i>Transfer Control Protocol/Internet Protocol</i>
ONR	<i>Office of Naval Research</i> , ou Escritório de Pesquisa Naval da Marinha americana
USV	<i>Unmanned Surface Vehicle</i> , ou Veículo de Superfície Não Tripulado
SNAME	<i>Society of Naval Architects and Marine Engineers</i>
RPY	Ângulos de Euler <i>Roll-Pitch-Yaw</i>
SRC	Sistema Robótico Cooperativo
CAS	<i>Computer Algebra System</i>
SGBD	Sistema Gerenciador de Bancos de Dados
XML	<i>Extended Markup Language</i>
UML	<i>Unified Modeling Language</i>
KAST	<i>Kinematic Analysis by Screw Theory</i>
PPR	Cadeia cinemática planar com juntas Prismática, Prismática, Rotativa
RPR	Cadeia cinemática planar com juntas Rotativa, Prismática, Rotativa
PPPS	Cadeia cinemática espacial com juntas Prismática, Prismática, Prismática, Esférica
RPPS	Cadeia cinemática espacial com juntas Rotativa, Prismática, Prismática, Esférica
RRPS	Cadeia cinemática espacial com juntas Rotativa, Rota-

MCR tiva, Prismática, Esférica
Matlab[®] Compiler Runtime
JRE *Java Runtime Environment*

LISTA DE SÍMBOLOS

O-xyz	Sistema de coordenadas inercial.....	71
O_v-x_vy_vz_v	Sistema de coordenadas do veículo.....	71
η_1, η_2, η_v	Posição, orientação e postura do veículo segundo o referencial inercial.....	72
x_v, y_v, z_v	Componentes da posição do veículo.....	72
ϕ_v, θ_v, ψ_v	Orientação do veículo expressa em ângulos RPY.....	72
ν_1, ν_2, ν_v	Velocidade linear, velocidade angular e velocidade total do veículo.....	72
u_v, v_v, w_v	Componentes da velocidade linear do veículo nas direções $\mathbf{x}_v, \mathbf{y}_v$ e \mathbf{z}_v	72
p_v, s_v, r_v	Componentes da velocidade angular do veículo em torno das direções $\mathbf{x}_v, \mathbf{y}_v$ e \mathbf{z}_v	72
J	Jacobiano.....	73
\dot{x}	Derivada temporal de x	73
\ddot{x}	Derivada temporal segunda de x	73
${}^i\mathbf{R}_j$	Matriz de rotação do sistema de coordenadas j para o sistema de coordenadas i	73
c_α, s_α	Abreviaturas das funções trigonométricas cosseno e seno respectivamente.....	73
$\eta_{e1}, \eta_{e2}, \eta_e$	Posição, orientação e postura do efetuador final segundo o referencial inercial.....	75
O_b-x_by_bz_b	Sistema de coordenadas da base do manipulador.....	75
O_e-x_ey_ez_e	Sistema de coordenadas do efetuador final.....	75
$\mathcal{K}(\cdot)$	Função cinemática direta.....	75
q	Vetor de variáveis das juntas do manipulador.....	75
ζ	Vetor de velocidades de um UVMS.....	75
$\nu_{e1}, \nu_{e2}, \nu_e$	Velocidade linear, velocidade angular e velocidade total do efetuador final do manipulador expressa segundo o referencial do efetuador final.....	76
ω_i	Velocidade angular do elo i	76
$\dot{\omega}_i$	Aceleração angular do elo i	76
\mathbf{v}_i	Velocidade linear de um ponto i	76
$\mathbf{v}_{i,c}$	Velocidade linear do centro de massa do elo i	76

\mathbf{a}_i	Aceleração linear do elo i	76
$\mathbf{r}_{i,j}$	Vetor entre as origens dos sistemas de coordenadas i e j	76
$\mathbf{r}_{i-1,i,c}$	Vetor entre a origem do sistema de coordenadas $i - 1$ e o centro de massa do elo i	76
\mathbf{z}_i	Vetor direção do movimento da junta i	76
q_i	Posição da junta i	76
\dot{q}_i	Velocidade da junta i	76
\ddot{q}_i	Aceleração da junta i	76
\mathcal{S}	Helicoide, heligiro	79
$\hat{\mathcal{S}}$	Helicoide normalizado	79
\mathbf{s}	Direção do eixo de um helicoide (vetor unitário)	79
\mathbf{s}_0	Posição do eixo do helicoide	79
h	Passo do helicoide	79
${}^i\mathbf{T}_j$	Matriz de transformação de helicoides de um referencial j para um referencial i	79
$\mathcal{S}(\boldsymbol{\chi})$	Operador matriz antissimétrica do vetor $\boldsymbol{\chi}$	79
\mathbf{I}_n	Matriz identidade de dimensão n	80
\mathbf{N}	Matriz de rede	82
F_b	Somatório dos graus de liberdade das juntas de uma cadeia cinemática	82
λ	Dimensão do espaço de helicoides	82
l	Número de circuitos independentes de um grafo	82
$\mathbf{0}_{[r \times c]}$	Matriz de dimensão $r \times c$ cujos elementos são iguais a zero	82
\mathbf{D}	Matriz de heligiros normalizados das juntas de uma cadeia cinemática	83
\mathbf{B}	Matriz de circuitos de uma cadeia cinemática	83
$\mathbf{1}_{[r \times c]}$	Matriz de dimensão $r \times c$ cujos elementos são iguais a um	83
$\text{diag}\{\mathbf{r}\}$	Operador gerador de matriz diagonal do vetor linha \mathbf{r}	83
$\mathbf{N}_p, \mathbf{N}_s$	Partições primária e secundária da matriz de rede, respectivamente	83
$\dot{\mathbf{q}}_p, \dot{\mathbf{q}}_s$	Partições primária e secundária das velocidades, respectivamente	83
\mathbf{A}	Matriz de transformação homogênea	97

W	Matriz de pesos do operador de pseudoinversão poderada	134
D	Função dinâmica inversa	136
τ	Torques e forças do sistema	136
\mathbf{K}_p	Matriz de ganhos de realimentação do erro de posição	143
\mathbf{e}_d	Vetor erro de posição	143
Ψ	magnitude do helicóide	208
$\mathbf{0}_c$	Vetor de 0's de dimensão $[1 \times c]$	265
$\mathbf{1}_c$	Vetor de 1's de dimensão $[1 \times c]$	265

LISTA DE ALGORITMOS

1	Pseudocódigo de <code>KCComposable.update</code>	117
2	Pseudocódigo de execução das fases de intervenção de uma missão	125
3	Pseudocódigo de execução das fases de intervenção de uma missão - aspectos cinemáticos	126
4	Ações gerais da intervenção em uma missão	129
5	Pseudocódigo de <code>Intervention.execute</code>	129

LISTA DE SCRIPTS

1	Implementação de <code>_update</code>	237
2	Exemplo de uso de <code>KCComposable</code>	239
3	Exemplo de especificação de trajetória no formato XML .	241
4	Exemplo de uso de geradores de trajetória	242
5	Exemplo de uso <code>TrajFactory</code>	243
6	Exemplo de especificação de trajetória no formato XML .	244
7	Implementação de <code>_update</code>	244
8	Exemplo de classe de intervenção	245

SUMÁRIO

1 INTRODUÇÃO	35
1.1 O AMBIENTE SUBAQUÁTICO	35
1.2 VEÍCULOS SUBAQUÁTICOS NÃO TRIPULADOS	36
1.2.1 Classificação dos UUV	36
1.2.2 Emprego de ROV e AUV	40
1.2.3 Autonomia dos UUV	44
1.2.4 Uma Nova Classificação Para UUV	46
1.3 OBJETIVO DA TESE	48
1.4 ORGANIZAÇÃO DO TEXTO DA TESE	49
2 SISTEMAS VEÍCULO-MANIPULADOR SUBAQUÁTICOS	51
2.1 MISSÕES DE INTERVENÇÃO	51
2.2 SISTEMAS DE UM UVMS	55
2.2.1 Comando	55
2.2.1.1 Arquitetura do Sistema Embarcado	57
2.2.1.2 Acompanhamento da Missão	58
2.2.1.3 Navegação	59
2.2.1.4 Tolerância a Falhas	60
2.2.2 Execução	61
2.2.2.1 Atuação	61
2.2.2.2 Comunicação	62
2.2.2.3 Manipulação	63
2.2.3 Estrutura	64
2.2.3.1 Componentes Estruturais	64
2.2.3.2 Energia	65
2.2.4 Autonomia	66
2.3 TRABALHOS RELACIONADOS AOS UVMS	67
3 CINEMÁTICA DE UM UVMS	71
3.1 CINEMÁTICA DO VEÍCULO	71
3.2 CINEMÁTICA DO SISTEMA VEÍCULO-MANIPULADOR	74
3.3 CINEMÁTICA DE UVMS ATRAVÉS DE HELICOIDES	77
4 ANÁLISE CINEMÁTICA DE SISTEMAS DE INTERVENÇÃO SUBAQUÁTICA	85
4.1 EXTENSÃO DO MODELO CINEMÁTICO BASEADO EM HELICOIDES	85
4.1.1 UVMS Com Manipuladores Operando de Forma Independente Entre Si	86

4.1.2 UVMS Com Manipuladores Operando em Coopera- ção.....	88
4.1.3 Cooperação Entre UVMS	89
4.1.4 UVMS em Situação de Evitamento de Colisão	92
4.1.5 Observações Sobre Os Modelos Cinemáticos	94
4.2 MODULARIZAÇÃO DA MODELAGEM CINEMÁTICA ...	95
4.3 VARIABILIDADE DO MODELO CINEMÁTICO DE SIS- TEMAS SUBAQUÁTICOS DE INTERVENÇÃO	98
5 ASPECTOS DE PROJETO DE UM <i>FRAMEWORK</i> PARA MODELAGEM CINEMÁTICA	105
5.1 MOTIVAÇÃO	105
5.2 REQUISITOS DE PROJETO	106
5.3 O ENFOQUE ORIENTADO A OBJETOS	108
5.4 MODELAGEM DO <i>FRAMEWORK</i>	109
5.4.1 Visão Geral	110
5.4.2 A Classe KCComponent	112
5.4.3 Helicoides e Juntas.....	115
5.4.4 Cadeias Cinemáticas	116
5.4.5 Geradores de Trajetória.....	118
5.4.6 Extensão e Especialização de Cadeias Cinemáticas .	120
6 ESTRATÉGIAS DE PLANEJAMENTO DE MOVI- MENTO PARA UVMS EM INTERVENÇÃO	123
6.1 VISÃO GERAL DO PLANEJAMENTO DE MOVIMENTO EM SISTEMAS DE INTERVENÇÃO	123
6.2 EXECUÇÃO DE TAREFAS DE INTERVENÇÃO	124
6.3 DEFINIÇÃO DE UM <i>FRAMEWORK</i> PARA TAREFAS DE INTERVENÇÃO	127
6.3.1 Classe Intervention	128
6.3.2 Classe Mission	130
6.3.3 Classe Task	131
6.3.4 Classe Guidance	133
6.3.5 Classe Uvms	135
6.4 SISTEMÁTICA DE DESENVOLVIMENTO DE ESTRA- TÉGIAS DE PLANEJAMENTO DE MOVIMENTO	136
7 SIMULAÇÕES E ANÁLISE	141
7.1 UVMS COM UM MANIPULADOR.....	141
7.1.1 Execução de uma trajetória retilínea	141
7.1.2 Reorientando o veículo durante a execução da tarefa	148
7.1.3 Execução da Tarefa na Presença de Falha	151
7.1.4 Execução de uma trajetória fechada.....	155
7.2 UVMS COM DOIS MANIPULADORES EM COOPERAÇÃO	159

7.2.1 Posicionamento e encaixe de uma peça	159
7.3 DOIS UVMS EM COOPERAÇÃO	165
7.3.1 Posicionamento de uma peça	166
7.3.2 Execução de tarefa que exige deslocamento dos UVMS	170
7.4 DOIS UVMS COM DOIS MANIPULADORES EM COO- PERAÇÃO	174
7.4.1 Posicionamento de uma estrutura cilíndrica	177
8 CONCLUSÃO	187
8.1 CONTRIBUIÇÕES DA TESE	187
8.2 PERSPECTIVAS E TRABALHOS FUTUROS	189
8.3 CONSIDERAÇÕES FINAIS	191
Referências Bibliográficas	193
APÊNDICE A – Fundamentos da Análise Cinemática Por Helicoides	207
APÊNDICE B – Implementação e Uso do <i>Framework</i> KAST	227
APÊNDICE C – Modelos Cinemáticos Utilizados	249
APÊNDICE D – Mapa Conceitual da Robótica Subaquá- tica	277

1 INTRODUÇÃO

Esta tese estuda o problema do planejamento de movimento de sistemas veículo-manipulador subaquáticos em tarefas de intervenção. Este problema consiste na definição dos movimentos que o sistema deve realizar para atender às especificações das tarefas. Os sistemas veículo-manipulador subaquáticos são extensamente usados na exploração dos oceanos e outras massas d'água, o que justifica o grande interesse no seu desenvolvimento. Um dos principais desafios de pesquisa é a autonomia de operação, para a qual o planejamento de movimento é um requisito importante. Este capítulo apresenta uma contextualização do problema de pesquisa através da descrição do meio subaquático e dos veículos não tripulados usados para a sua exploração. A seguir, o objetivo da tese é descrito e a organização do restante deste texto é apresentada.

1.1 O AMBIENTE SUBAQUÁTICO

O planeta Terra tem aproximadamente 71% de sua superfície coberta por água, sendo 97,5% desta correspondente a oceanos e mares. Além de sua importância incontestável para a manutenção da vida no planeta, a exploração de seus recursos naturais são de grande importância econômica e social. Além dos oceanos, os cursos de água doce, como rios e lagos, são fundamentais para os ecossistemas, a navegação e a exploração humana. Grande parte das civilizações se desenvolveram próximas a algum corpo d'água. Apesar disso, os oceanos e corpos de água doce são muito pouco conhecidos. Apenas a superfície é extensamente utilizada. As características do meio subaquático tornam-no inóspito aos humanos, dificultando a sua exploração (Schmiegelow, 2004; UNESCO, 2009).

Atualmente, esses locais são considerados importantes fontes de alimentos e de recursos energéticos como petróleo e gás. Outras possibilidades de exploração, como a mineração do leito oceânico, estão sendo estudadas. O uso dos mares como meio de transporte é de grande importância para a economia global. Além disso, boa parte das comunicações intercontinentais se dá através de cabos submarinos depositados no leito oceânico (Schmiegelow, 2004; Antonelli; Fossen; Yoerger, 2008; Whitcomb, 2000).

A matriz energética global, baseada principalmente em combustíveis fósseis, demanda a pesquisa por fontes de obtenção de petró-

leo e gás a profundidades cada vez maiores nos oceanos. Observa-se esse fato pelas sucessivas marcas em profundidades alcançadas, como o atual recorde de perfuração de 3.107m na Índia (Offshore Engineering, 2011b, 2011a), de extração (1.886m) e de produção (1.413m). Estes dois últimos foram obtidos pela Petrobras, empregando boa parte de tecnologia nacional (Petrobras, 2011b, 2011c). As recentes descobertas de reservas no pré-sal em lâminas d'água variando entre 2.000m e 3.000m deverão estabelecer novos recordes e trazem desafios tecnológicos a serem superados (Petrobras, 2011a).

Além disso, o conhecimento dos oceanos é essencial para diversas áreas do conhecimento como a climatologia, a biologia, a história, a geologia e até mesmo a exobiologia.

Os riscos e custos envolvidos nas operações submarinas incentivam a pesquisa e o desenvolvimento dos *Veículos Subaquáticos Não Tripulados* (em inglês, *Unmanned Underwater Vehicles, ou UUV*) (Antonelli; Fossen; Yoerger, 2008; Marani; Choi; Yuh, 2009).

1.2 VEÍCULOS SUBAQUÁTICOS NÃO TRIPULADOS

Os veículos subaquáticos não tripulados são instrumentais para a exploração submarina em diferentes tipos de aplicação, por apresentar vantagens em relação a outros meios. Em relação ao uso de mergulhadores, os UUV reduzem os riscos inerentes à pressão, temperatura e falta de ar (Bennett et al., 1984; Tonjum; Peterson; Florio, 1984), além de alcançarem profundidades muito maiores que as possíveis para humanos (Yuh, 1990). Os veículos tripulados, por sua vez, têm restrições quanto ao espaço que podem alcançar, além de serem muito onerosos (Antonelli, 2006). Uma estimativa de 2007 indica aproximadamente apenas 100 desses sistemas em uso para fins não militares (Kohnen, 2008). Assim, os UUV são responsáveis por grande parte das operações nos ambientes imersos (Antonelli; Fossen; Yoerger, 2008).

Existem diferentes tipos de veículos não tripulados, projetados de acordo com o tipo de missão a ser realizada.

1.2.1 Classificação dos UUV

A classificação comumente adotada pela literatura e pela indústria para os veículos subaquáticos não tripulados é feita de acordo com a capacidade de operação autônoma. Tem-se, então, as classes de *Veí-*

culos Remotamente Operados e de Veículos Subaquáticos Autônomos.

Os Veículos Remotamente Operados, ou *ROV* (do inglês *Remotely Operated Vehicles*), têm como característica principal a sua conexão física com uma base através de um *cordão umbilical* (ou *tether*, em inglês). Ele é usado para transmitir energia ao veículo e fazer sua comunicação de dados com a base. Muitos modelos de *ROV* apresentam manipuladores, através dos quais fazem contato físico com o meio. Por isso, são usados tanto em operações de inspeção quanto de intervenção, em especial nas instalações de petróleo e gás *offshore* (Antonelli; Fossen; Yoerger, 2008), como é ilustrado na Figura 1 (Oceaneering, 2009).

Apesar de sua extensa utilização, os *ROV* apresentam grandes limitações a serem superadas. O cordão umbilical restringe os movimentos do veículo, além de acrescentar efeitos dinâmicos indesejados. A operação do *ROV* exige operadores qualificados e habilidosos, além da presença de uma base de operações à qual ele se conecta fisicamente (em geral, um navio). Por isso, as missões são bastante onerosas, e muitas vezes limitadas pelas condições meteorológicas do sítio de operação e pelo desgaste dos operadores. A duração de uma missão é fortemente influenciada por todos esses fatores (Yuh et al., 1998; Antonelli, 2006; Paschoa, 2010).



Figura 1. *ROV* Oceaneering Millenium conectando um duto em um painel (Oceaneering, 2009)

Embora se afirme que a efetividade do uso de *ROV* diminui com a profundidade, devido aos custos e à dificuldade de manipulação de cordões umbilicais extensos (Antonelli; Fossen; Yoerger, 2008), eles são usados em profundidades de até 3.000m na indústria de petróleo e gás

offshore (Petrobras, 2011a). O avanço na tecnologia de materiais, energia e comunicações está gradualmente superando muitas limitações causadas pelos cordões umbilicais, à medida em que estes diminuem de espessura e peso. O recorde de 11.000m de profundidade alcançado pelo ROV Nereus resulta dessas inovações (Inovação Tecnológica, 2009).

Os Veículos Subaquáticos Autônomos, também conhecidos como AUV (de *Autonomous Underwater Vehicles*), são veículos que prescindem do cordão umbilical. Para tanto, eles possuem suprimento de energia próprio e se comunicam através de enlaces sem fio. Os AUV possuem maior liberdade de movimentos que os ROV, podendo alcançar distâncias maiores e operar por longos períodos. Isso os torna adequados para operações de coleta de dados científicos, mapeamento oceanográfico, inspeção de sítios e missões militares de longa duração.

As limitações da comunicação sem fio no meio subaquático restringem a comunicação com uma base, fazendo com que esses veículos tenham de executar as tarefas atribuídas sem interação contínua com operadores. Essa característica é definida como a *autonomia* do veículo (Button et al., 2009). Tal autonomia implica na habilidade de planejamento de execução das tarefas e na elaboração de soluções para eventos não previstos pelo próprio veículo. Para tanto, os AUV são equipados com vários tipos de sensores para localização e mapeamento. Além disso, demanda grande capacidade de processamento de informações.

O avanço dos sensores e dos meios de comunicação permitem o uso de AUV em meios cada vez mais inóspitos. Um exemplo é o do robô ENDURANCE, originalmente projetado para a exploração dos oceanos da lua Europa, de Júpiter. Depois de construído, esse robô foi utilizado para a exploração de lagos e mares abaixo das calotas polares terrestres, um ambiente até então desconhecido. Além de avaliar o desempenho e funcionalidade do robô, o sucesso da missão teve grande relevância científica. Na Figura 2 é mostrado um teste de campo desse AUV (Stone Aerospace, 2012).

Apesar do grande avanço tecnológico, a maioria dos AUV são projetados para operações do tipo *fly-by*, onde não existe contato físico com o meio. Isso se deve a diversos fatores, como a baixa largura de banda e atrasos da comunicação acústica (a comunicação eletromagnética é praticamente impossível no meio subaquático), o limitado suprimento de energia e a inerente complexidade envolvida em tarefas de intervenção (Whitcomb, 2000; Antonelli; Fossen; Yoerger, 2008; Marani; Choi; Yuh, 2009). Na Figura 3 apresenta-se o Remus 6000, que foi utilizado na localização das caixas pretas do voo AF447 no Oceano Atlântico (Terra, 2011).



Figura 2. O robô ENDURANCE (Stone Aerospace, 2012)



Figura 3. AUV Remus 6000 da Hydroid (Kongsberg, 2011)

Os *Sistemas Veículo-Manipulador Subaquáticos*, ou *UVMS* (do inglês *Underwater Vehicle-Manipulator Systems*) são veículos subaquáticos não tripulados aos quais um ou mais manipuladores são acoplados com a finalidade de realizar operações de interação com o meio. O acréscimo de manipuladores aumenta a complexidade do sistema devido à redundância cinemática intrínseca e à interação dinâmica veículo-manipulador-meio. Os UVMS costumam ser completamente atuados de forma a compensar as forças e momentos gerados pela operação de manipulação. Isso faz com que esses sistemas sejam usualmente teleoperados, por causa das dificuldades de implementar recursos de operação autônoma (Antonelli; Fossen; Yoerger, 2008). A Figura 4 apresenta o ALIVE, da empresa Cybernetix, projetado para uso em instalações de petróleo e gás.



Figura 4. Cybertetix ALIVE, UVMS com capacidade de operação autônoma ou teleoperada (Cybertetix, 2008)

Os UVMS são também fundamentais em missões de atendimento a desastres, como no caso da plataforma Deepwater Horizon da British Petroleum (v. Figura 5a) (British Petroleum, 2010). Investigação, resgates e recuperação de destroços como no acidente do voo AF447 também utilizam esses sistemas (v. Figura 5b) (Terra, 2011; Phoenix, 2011).

1.2.2 Emprego de ROV e AUV

Segundo um relatório de tendências da indústria *offshore*, em 2009 existiam 629 AUV em operação, com uma projeção de demanda de aproximadamente 1.150 veículos em um período de 10 anos. Outro levantamento da mesma consultoria prevê a expansão de 420 para 570 ROV das classes de trabalho geral e pesado para atender às demandas da indústria de petróleo e gás em 5 anos¹ (Douglas-Westwood, 2011). Os gráficos da Figura 6 sintetizam esses dados. De acordo com outro relatório, haviam mais de 1.000 ROV de pequeno porte em operação em 2009, considerando apenas os modelos comerciais e de instituições de pesquisa (Button et al., 2009). Esses números ilustram a importância que os UUV têm para as atividades subaquáticas em diferentes áreas.

Embora não exista uma classificação formal dos UUV quanto ao porte e a capacidade, existem designações aceitas nos meios em que estes são empregados (MTS, 2006; Button et al., 2009). As Tabelas 1 e 3 relacionam essas classes de ROV e AUV, respectivamente, des-

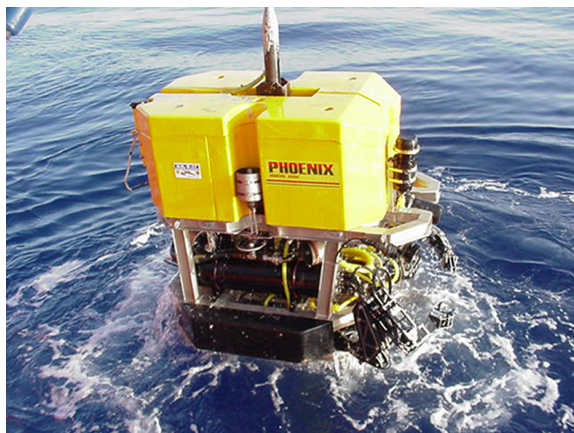
¹em 2006, a Marine Technology Society estimava a existência de 435 ROV desse tipo em operação (MTS, 2006).

crevendo as características desses sistemas. As Tabelas 2 e 4, por sua vez, ilustram usos de acordo com as classes. Os nomes das classes são adaptações dos termos comumente usados em inglês.

Existem alguns sistemas subaquáticos que não se enquadram bem nas classificações de UUV apresentadas. É o caso dos veículos híbridos, que podem operar como ROV ou AUV de acordo com a natureza da tarefa. Estes podem utilizar o cordão umbilical para fornecimento de energia/comunicação ou operar por baterias utilizando a



(a)



(b)

Figura 5. UVMS em operação: (a) sítio do vazamento de petróleo no Golfo do México (Leff; Plushnick-Masti, 2010); (b) Remora 6000 sendo lançado (Phoenix, 2011)

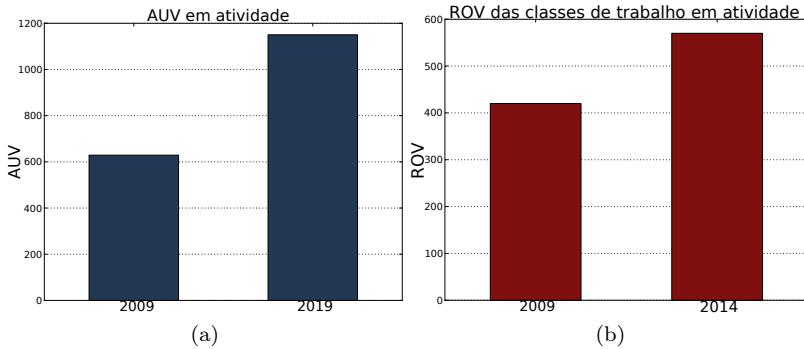


Figura 6. Projeções de expansão do uso de UUV: (a)AUV; (b)ROV de classes de serviço(Douglas-Westwood, 2011)

Tabela 1. Classes de ROV

Classe	Acionamento	Profundidade	Carga
Pequenos	elétrico	até 300m	câmera/sensores
Elétricos de alta capacidade	elétrico	até 6.000m	alguns kg
Trabalho geral	elétrico hidráulico	até 1.000m	até 450kg
Trabalho pesado	hidráulico	até 3.000m	até 750kg

Tabela 2. Usos de ROV

Classe	Usos
Pequenos	Inspeção; supervisão de mergulhadores
Elétricos de alta capacidade	Pesquisas científicas; usos militares; operação de painéis e válvulas
Trabalho geral	Construção, manutenção e inspeção de instalações de petróleo e gás
Trabalho pesado	Operações especiais de construção / manutenção de instalações de petróleo e gás

conexão apenas para comunicação. No modo autônomo, o cordão não é utilizado, e assim não há comunicação constante com a base.

Alguns projetos de AUV também não se enquadram na classificação. Esse é o caso dos sistemas biomiméticos (que emulam o compor-

Tabela 3. Classes de AUV

Classe	Autonomia	Carga	Lançamento
Portáteis	até 20h	até 45kg	manual
Leves	até 40h	até 220kg	por lançadores
Pesados	até 80h	até 1.350kg	compatíveis com submarinos
Grandes	até 400h	até 10.000kg	compatíveis com submarinos e navios

Tabela 4. Usos de AUV

Classe	Usos
Portáteis	Operações de inspeção e coleta de dados
Leves	Coleta de dados, mapeamento, inspeção
Pesados	Aplicações militares
Grandes	Aplicações militares

tamento de seres vivos) e dos *gliders* (que utilizam a energia potencial para seu deslocamento por grandes distâncias).

As operações executadas por UUV podem ser classificadas como sendo de *cruzeiro* ou de *intervenção* em função de seu movimento durante a execução e de sua interação com o ambiente aquático. Na literatura são encontradas variações desses termos, porém conservando as características que definem cada classe, de forma que a distinção se mantém válida (Antonelli; Fossen; Yoerger, 2008; Yuh, 2000; Marani; Choi; Yuh, 2009).

Nas operações de *cruzeiro*, o UUV não interage fisicamente com o meio aquático. O nome se deve ao fato do veículo estar em movimento durante a execução de tarefas, se deslocando segundo uma trajetória predeterminada. O veículo carrega um conjunto de sensores usados para coleta de dados do ambiente, mapeamento do leito marinho, supervisão ou vigilância. Por esse motivo, é comum essas operações serem chamadas de *pesquisa* ou *inspeção* (do inglês *survey*) (Antonelli; Fossen; Yoerger, 2008; Yuh, 2000).

As operações de *intervenção* são aquelas onde há interação com o meio. Em muitas situações, essas operações exigem que o veículo permaneça parado em determinada posição/orientação, a despeito das perturbações do ambiente como as causadas por correntes, marés e ondas. Por isso, também são chamadas de operações *estacionárias*

(do inglês *hovering*)(Antonelli, 2004; Marani; Choi; Yuh, 2009; Soyly; Buckham; Podhorodeski, 2010).

Os AUV estão gradativamente substituindo os ROV nas operações de cruzeiro, por causa das características autônomas que permitem maior liberdade de movimentos, alcance e duração de missões (Tavares, 2003). O cordão umbilical, principalmente, traz limitações quanto às distâncias que podem ser percorridas pelo veículo (aproximadamente 1km de raio, no caso de cordões que fornecem energia ao veículo) (Button et al., 2009). Além disso, acrescenta-se o problema do manejo do cordão umbilical (El-Hawary, 2001).

Nas operações de intervenção, porém, os ROV ainda prevalecem devido à complexidade da interação física com o meio, que exige atuação constante dos operadores. Além disso, para permanecer estacionário, o veículo deve ser capaz de se movimentar de forma independente em qualquer direção no espaço, o que não é possível para a maioria dos AUV, devido ao seu projeto ser voltado para maior eficiência em operações de cruzeiro. A Tabela 5 apresenta uma relação de tarefas em que os UUV são utilizados ou cujo potencial de uso está sendo pesquisado.

1.2.3 Autonomia dos UUV

A autonomia é considerada o maior desafio de longo prazo no desenvolvimento dos AUV (Button et al., 2009). Mesmo existindo resultados significativos na área de navegação, o que viabilizou o uso dos AUV em operações de cruzeiro (como as listadas na Tabela 4), há a necessidade de aperfeiçoamentos nos sistemas de localização espacial, identificação de obstáculos e alvos. Atualmente, várias pesquisas estão em curso na área de sensores e atuadores, assim como em sistemas de mapeamento, localização, estratégias de aproximação de alvos e de evitamento de obstáculos.

O problema da autonomia pode ser dividido em dois enfoques distintos. A *autonomia física* consiste em possibilitar o funcionamento do UUV livre de conexões físicas com uma base ou outros veículos. Já a *autonomia operacional* pode ser definida como a capacidade do UUV analisar a missão a ele designada, planejar o seu movimento e adaptar sua operação de acordo com mudanças ocorridas no espaço de trabalho durante a sua execução (Rocha; Dias, 2010b).

Neste texto, considera-se que as questões relacionadas à autonomia operacional são mais críticas para o desenvolvimento da autonomia dos UUV, pois mesmo sistemas conectados por cordões umbilicais po-

Tabela 5. Atividades que empregam UUV

Tipo	Atividade
Científica	Coleta de amostras físicas, químicas e biológicas Mapeamento do leito oceânico e das calotas polares Caracterização do perfil da coluna d'água Exploração de sítios de grande profundidade Transporte e operação de equipamentos de pesquisa Arqueologia subaquática Ensaio não destrutivo Investigação de sítios de desastres
Ambiental	Monitoramento de propriedades da água Monitoramento de espécimes biológicos Remoção de destroços e detritos
Militar	Investigação de locais de atracação Contramedidas contra minas aquáticas Busca e resgate de submarinos e armamentos Vigilância de portos e costa
Industrial e Comercial	Lançamento e manutenção de cabos Inspeção de cabos e tubulações Manutenção de tubulações Construção e manutenção de estruturas <i>offshore</i> Operação de painéis e válvulas Inspeção de cascos e tanques de navios Inspeção de usinas nucleares Salvatagem

dem se beneficiar da implementação de algum grau de independência de interação com operadores humanos, que assumem então o papel de supervisores da execução de tarefas que foram especificadas em alto nível para os UUV. Porém, isso não diminui a importância das pesquisas para resolver os problemas da autonomia física, como o fornecimento/-suprimento de energia, troca de informações e eficiência de atuadores.

Não existe uma definição precisa para o grau de autonomia de um UUV, embora algumas definições tenham sido propostas pela Marinha americana (Button et al., 2009). Alguns autores usam o termo *semi-autonomia* para se referirem a esse tipo de interação operador-veículo, com o objetivo de evoluir gradativamente os sistemas de missão de forma que as tarefas do operador sejam de nível cada vez mais alto, à medida que os sistemas do UUV tornam-se mais capazes de geren-

ciar os aspectos de operação anteriormente de responsabilidade humana (Marani; Choi; Yuh, 2009; Yuh et al., 1998). Este enfoque reforça a ideia de que a autonomia operacional e a capacidade de planejamento de movimento para resolução das tarefas são problemas de grande importância para a evolução dos UUV de totalmente teleoperados (ROV) para totalmente autônomos (AUV).

No caso das operações de intervenção, executadas por UVMS, a atuação autônoma também deve compreender a utilização dos manipuladores. O seu movimento causa efeitos dinâmicos significativos sobre o sistema veículo-manipulador que são de difícil identificação e modelagem. A aplicação de esforços sobre o ambiente gera desafios que são objeto de estudo até para o caso de manipuladores no ambiente industrial. Por esse motivo, a autonomia dos UVMS ainda é considerado um problema em aberto (Antonelli, 2004).

O desenvolvimento dos chamados I-AUV (do inglês *Intervention-AUV*, ou AUV de intervenção) é um dos grandes desafios de pesquisa na robótica subaquática. Este tipo de sistema deve ser capaz de se deslocar até o sítio de intervenção, reconhecer a tarefa a ser executada e agir sem a necessidade de supervisão e atuação humana contínua. Apesar de vários estudos terem sido publicados, existem poucos I-AUV construídos. Estes operam principalmente em ambientes conhecidos e controlados (Antonelli; Fossen; Yoerger, 2008; Yuh et al., 1998; De Novi et al., 2010). Os primeiros experimentos em mar aberto foram reportados apenas recentemente (Marani; Choi; Yuh, 2009). A execução de tarefas de intervenção de precisão ainda exige avanços em armazenamento de energia, sistemas computacionais, navegação subaquática e sensores (Paschoa, 2010).

1.2.4 Uma Nova Classificação Para UUV

Além de descrever a capacidade de autonomia de um UUV, a distinção entre ROV e AUV também tem sido associada ao tipo de tarefa usualmente empregado. Como ilustra uma visão de mapa conceitual apresentada na Figura 7, os AUV são costumeiramente associados às tarefas de cruzeiro, enquanto os ROV são associados às tarefas de intervenção. Isso se deve ao fato dos AUV serem usualmente projetados para operação em movimento contínuo, enquanto os UVMS são em sua maioria teleoperados e por isso comumente chamados de ROV. Além disso, a necessidade do cordão umbilical é decisiva na denominação do veículo subaquático.

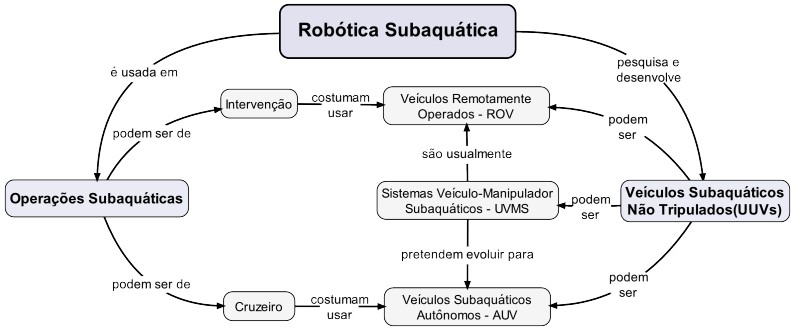


Figura 7. Relação entre UUV e operações subaquáticas

À medida que os sistemas de intervenção subaquática começam a ter características de operação autônoma, porém, a classificação comumente usada perde a sua validade. A autonomia de operação pode ser implementada mesmo em veículos com cordão umbilical, o que invalida o uso do termo ROV.

Uma vez que este texto concerne o planejamento de movimento para implementação de autonomia, será adotada uma classificação em função do tipo de tarefa para a qual o sistema subaquático é projetado. Assim, tem-se os sistemas de *cruzeiro* e os sistemas de *intervenção* subaquática (Rocha; Dias, 2010b). Uma visão de mapa conceitual refletindo essa nova classificação é apresentada na Figura 8. Esta é derivada do mapa conceitual apresentado no Apêndice D.

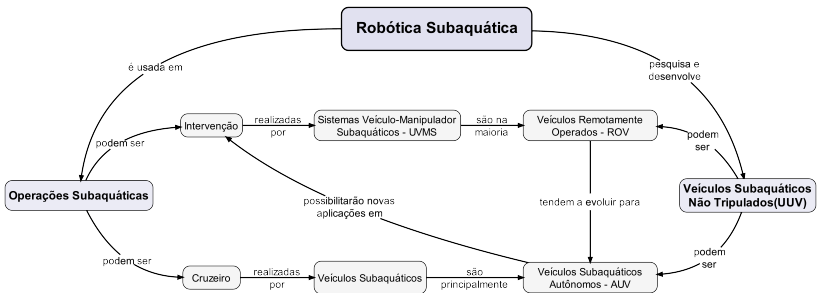


Figura 8. Visão atualizada da classificação da robótica subaquática

Alguns sistemas combinam características para satisfazer os dois tipos de tarefa (Marani; Choi; Yuh, 2009; De Novi et al., 2010). Estes

podem ser denominados sistemas *híbridos de cruzeiro/intervenção*.

1.3 OBJETIVO DA TESE

Com base na contextualização, observa-se que o desenvolvimento de I-AUV é um problema de pesquisa atual. Para a implementação de autonomia operacional nesses sistemas, é necessário desenvolver nos UVMS a capacidade de planejar o seu movimento em função das tarefas especificadas em alto nível por um operador e do meio em que estas deverão ser executadas.

Este trabalho tem por objetivo geral *analisar cinematicamente os sistemas de intervenção subaquática a fim de desenvolver estratégias de planejamento de movimento para esses sistemas robóticos que satisfaçam as especificações de tarefas de intervenção no ambiente da tarefa*.

São consideradas tarefas de intervenção aquelas em que um ou mais UVMS, já localizados no sítio de intervenção, realizam atividades de movimento dos efetuadores finais dos seus manipuladores, utilizando tanto os movimentos dos próprios manipuladores quanto dos veículos.

Para o desenvolvimento das estratégias de planejamento de movimento, as seguintes atividades necessárias à elaboração da tese são tratadas:

1. Análise cinemática de sistemas de intervenção subaquática em diferentes situações de operação, considerando um ou mais manipuladores no mesmo UVMS;
2. Cooperação entre manipuladores e entre UVMS para a execução de tarefas;
3. Modularização e configurabilidade das cadeias cinemáticas dos sistemas de intervenção;
4. Projeto e implementação de um *framework* computacional para modelagem cinemática baseada em helicoides;
5. Desenvolvimento de estratégias para a resolução de tarefas para os sistemas de intervenção considerando as tarefas especificadas e a possibilidade de ocorrência de eventos não planejados que modificam a relação entre o UVMS e o seu espaço de trabalho.

A cinemática por helicoides permite analisar de maneira uniforme e sistemática diferentes tipos de cadeias cinemáticas. Com isso,

distintos contextos de execução de tarefas podem ser representados por modelos baseados em helicoides. A definição desses modelos pode ser feita por modularização, onde cadeias mais simples cujos modelos já são conhecidos são vinculadas entre si para compor cadeias mais complexas. Assim, aplicações pouco tratadas na literatura como a cooperação entre UVMS são abordadas de maneira sistemática e extensível. Além disso, o modelo por helicoides e a resolução cinemática de cadeias cinemáticas é reconfigurável, o que permite definir uma sistematização para o planejamento de movimento dos UVMS quando o contexto da execução de tarefas varia ao longo do tempo.

Essas características são utilizadas na construção de *frameworks* para simplificar a implementação de estratégias de planejamento de movimento. Esses *frameworks* são usados para a modelagem cinemática dos sistemas de intervenção de forma modular e para a definição de tarefas a serem executadas por esses sistemas. A definição de uma estratégia de planejamento de movimento pode utilizar as características de reconfigurabilidade da resolução cinemática do modelo que foram implementadas nos *frameworks*. Assim, pode-se representar as variações no contexto de execução da tarefa de forma que ainda possam ser geradas referências de movimentos dos UVMS que permitam realizar a tarefa proposta. Como característica adicional do *framework* de modelagem, pode-se armazenar as definições de cadeias cinemáticas dos sistemas de intervenção de forma a criar um banco de dados de modelos de sistemas de intervenção e tarefas. Com isso, atende-se a uma das demandas para a obtenção da autonomia operacional plena dos sistemas de intervenção subaquática robótica.

1.4 ORGANIZAÇÃO DO TEXTO DA TESE

Este texto é organizado em mais seis capítulos além da Introdução. O Capítulo 2 faz uma revisão bibliográfica sobre os UVMS, listando pesquisas relacionadas com o presente trabalho e descrevendo os subsistemas que compõem um sistema de intervenção subaquática. Também como uma revisão bibliográfica, o Capítulo 3 trata da cinemática de UVMS apresentada na literatura.

No Capítulo 4 estende-se a análise cinemática dos UVMS baseada na teoria dos helicoides. São identificadas características dos sistemas subaquáticos existentes e demandas de novos tipos de missões. Observa-se a ocorrência de modelos cinemáticos mais complexos e até reconfiguráveis de acordo com as tarefas que compõem uma missão e

mudanças no meio em que esta é executada.

Os aspectos do projeto de um *framework* computacional para a modelagem das cadeias cinemáticas com base na análise do Capítulo 4 são tratados no Capítulo 5. Nele faz-se a justificativa pela abordagem orientada a objetos do *framework*, identificam-se os requisitos para os componentes de software e apresenta-se a solução adotada para essa modelagem.

No Capítulo 6 desenvolve-se uma metodologia para estratégias de planejamento de movimento de UVMS, elaborado a partir dos resultados da análise cinemática do Capítulo 4 e empregando o *framework* projetado no Capítulo 5. São abordados conceitos sobre a especificação de tarefas e a resolução destas de acordo com o contexto do espaço de trabalho em que ela se desenvolve e suas mudanças ao longo do tempo.

Simulações de cenários de execução de diferentes tarefas são descritas no Capítulo 7, bem como a análise dos resultados dessas simulações.

Encerra o texto o Capítulo 8, que trata das conclusões deste trabalho e de perspectivas de trabalhos futuros.

Quatro apêndices complementam o texto da Tese. O Apêndice A sintetiza os fundamentos e ferramentas cinemáticas com base em helicoides. O Apêndice B trata do *framework* desenvolvido para auxiliar a análise cinemática e a implementação de simulações. A descrição dos modelos cinemáticos e cenários das simulações é feita no Apêndice C. Por fim, apresenta-se no Apêndice D um mapa conceitual da robótica subaquática, onde os tópicos tratados neste texto são destacados.

2 SISTEMAS VEÍCULO-MANIPULADOR SUBAQUÁTICOS

Neste capítulo é apresentada uma visão geral dos sistemas veículo-manipulador subaquáticos. Inicia-se por uma caracterização de missões executadas por um UVMS e suas fases. Após, aspectos do projeto de UVMS e seus subsistemas são abordados. Esses aspectos são explorados através de visões de um mapa conceitual que faz uma releitura do panorama da robótica subaquática e das missões de intervenção. Trabalhos relevantes à pesquisa e desenvolvimento de sistemas subaquáticos de intervenção também são aqui relacionados. O mapa conceitual completo é apresentado no Apêndice D.

2.1 MISSÕES DE INTERVENÇÃO

Existem diferentes fases em missões de intervenção. Neste texto, essas missões são caracterizadas pela execução de alguma tarefa pelos manipuladores dos UVMS. De uma forma geral, uma missão inicia pelo lançamento do sistema veículo-manipulador. A seguir, ele se desloca até o sítio de intervenção. Ao chegar, os manipuladores devem ser ativados para em seguida realizarem a operação de manipulação. Ao se completar esta operação, os manipuladores são desativados para que o veículo se desloque até a base, onde por fim será recuperado, desligado e guardado.

Existem algumas variações nessas fases. Um exemplo é o uso de *gaiolas*, que não apenas servem para posicionar o veículo embaixo d'água, mas também são usadas como referencial para os veículos, retransmissor de comunicação e fornecedor de energia. Outros casos particulares são o lançamento por submarinos (em aplicações militares) e os sistemas híbridos, que partem como veículos de cruzeiro fisicamente autônomos e se conectam a um cordão umbilical apenas no sítio de intervenção. Alguns veículos ainda contam com recursos de auto estacionamento na gaiola ou na base.

A caracterização de missões de intervenção feita neste trabalho resultou na classificação e descrição das suas fases, que são descritas a seguir:

Lançamento Muitos UVMS são grandes demais para lançamento manual (v. Tabelas 1 e 2). Assim, eles são normalmente içados até a água e abaixados a uma profundidade onde os efeitos hidrodinâmicos

nâmicos presentes na superfície são minimizados. Ao chegar na profundidade desejada, o veículo é acionado e começa a usar seus sensores e atuadores. Em alguns casos, o lançamento consiste em duas fases, onde uma gaiola é levada à água e, após chegar em determinada profundidade, ocorre a liberação do veículo;

Deslocamento ao sítio de intervenção Após o lançamento, o caminho que o veículo deve percorrer para chegar ao local da intervenção é definido. Com isso, os propulsores são acionados para que ele siga esse caminho. Essas ações podem ser executadas por um operador de forma teleoperada ou pelos sistemas de processamento de informações do próprio veículo. Durante o deslocamento, os manipuladores ficam armazenados dentro do casco, para minimizar os efeitos hidrodinâmicos de interação dos manipuladores com a água e para proteção;

Ativação dos manipuladores Ao chegar no local de intervenção, os manipuladores devem ser liberados e seus sistemas iniciados, assumindo uma postura inicial. Alguns sistemas podem ainda compreender uma fase de calibração dos sensores do manipulador, ou a montagem de ferramentas;

Execução da intervenção Em geral, a operação principal da missão do UVMS ocorre com o veículo estacionário. Para tanto, este deve compensar a influência do meio, como correntes e amortecimentos, bem como os efeitos que os manipuladores causam durante a execução das tarefas. Além disso, o movimento de cada manipulador e do veículo deve ser planejado de forma a minimizar o consumo de energia, que pode ser crítico no caso de sistemas fisicamente autônomos. Deve-se também considerar a possibilidade de ocorrência de situações não previstas na especificação da tarefa e consequentes mudanças nesta para compensar as mudanças de contexto. O evitamento de colisão com obstáculos móveis e a compensação do movimento dos objetos a serem manipulados são exemplos desse tipo de problema;

Desativação dos manipuladores Após o término da operação de intervenção, os manipuladores devem ser recolhidos e desativados para que o veículo possa retornar à sua base. Define-se então o movimento de recolhimento dos manipuladores para que fiquem protegidos na estrutura do veículo. Após o armazenamento, os manipuladores podem ser desativados para economizar energia;

Deslocamento à base Esta fase é essencialmente inversa à do deslocamento ao sítio de intervenção. Uma diferença é a localização da base por sistemas autônomos, que podem ser móveis no caso de navios ou submarinos. Para tanto, podem ser usados faróis acústicos para que o UVMS possa localizar a base ou definir um local de encontro, que será o ponto onde o UUV virá à tona, para que o pessoal de apoio da base o encontre e o recupere;

Recuperação Ao retornar, o UVMS deve ser trazido de volta ao seu local de armazenamento da base. Para tanto, ele é içado de volta. Em sistemas que usam o lançamento por gaiola, o que ocorre é o retorno e o estacionamento do veículo nesta, que após é içada.

A realização dessas fases envolve diferentes atividades que são executadas pelos operadores e pelos subsistemas do UVMS. A participação de cada ator (operadores ou subsistemas) varia de acordo com o grau de autonomia do sistema de intervenção e da natureza da missão.

Para as fases que não envolvem o uso dos manipuladores, o objetivo é o deslocamento do veículo de forma a atender aos caminhos e posturas definidas para o sistema. As atividades relevantes estão inter-relacionadas de acordo com o diagrama da Figura 9. Nesse diagrama, as setas indicam o sentido do fluxo de informações entre as atividades e os círculos são conectores de informações.

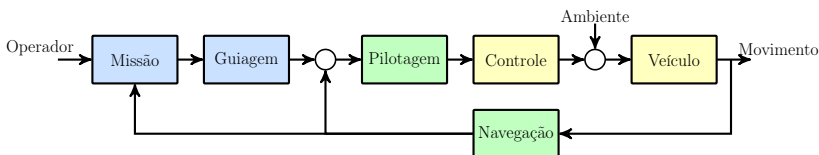


Figura 9. Diagrama de blocos de atividades de um UVMS em fases que não envolvem intervenção

A atividade *Missão* compreende a interação do UVMS com o usuário, fornecendo uma interface homem-máquina para isso. Além de apresentar diferentes informações de sensores relativas ao estado do UVMS, a interface pode apresentar informações dos sensores relativas à missão. É possível também armazenar os dados de sensores para análise posterior. Diferentes dispositivos para o acionamento do sistema são disponibilizados ao usuário de acordo com a necessidade de operação remota. Quanto maior o grau de autonomia, menor a interação do operador com o sistema, o que implica em um caráter mais supervisor do que interventivo para esta atividade. Em um UVMS totalmente

autônomo, o operador usaria a interface para especificar a missão em alto nível e a partir desta a atividade geraria definições de baixo nível para as demais atividades do sistema, tais como o seguimento de um caminho pelo veículo.

A *Guiagem* recebe as descrições das tarefas como caminhos para o UVMS, gerando as referências de movimento dos atuadores necessárias para que esses caminhos sejam seguidos (Fossen, 1994). O planejamento de movimento emprega a resolução da cinemática inversa para a geração das referências. Para lidar com a possibilidade de mudanças de contexto de execução (devido a eventos não previstos ou a requisitos de diferentes subtarefas) a guiagem pode utilizar diferentes modelos cinemáticos do sistema UVMS-meio, escolhendo o mais adequado para a realização da tarefa a cada instante de acordo com a mudança de estado da missão.

A atividade de *Pilotagem* é responsável por determinar as referências de esforços para os atuadores através da dinâmica inversa. Ela usualmente está integrada à atividade de *Controle*, que é responsável por acionar os atuadores do UVMS de forma que estes sigam as referências de esforços.

Por fim, a atividade de *Navegação* processa informações de sensores e dados pré-armazenados do ambiente de operação (como mapas parciais e localização de balizadores) para determinar a posição e orientação do sistema, alimentando a missão e a pilotagem.

Nas fases da missão relacionadas com a intervenção, outras atividades são relacionadas ao uso dos manipuladores. Nessas fases, o objetivo das tarefas usualmente está relacionado ao movimento dos efetuadores finais, enquanto o veículo assume uma postura estacionária ou colabora com o movimento. O diagrama da Figura 10 descreve como as atividades do sistema se interrelacionam nessas fases.

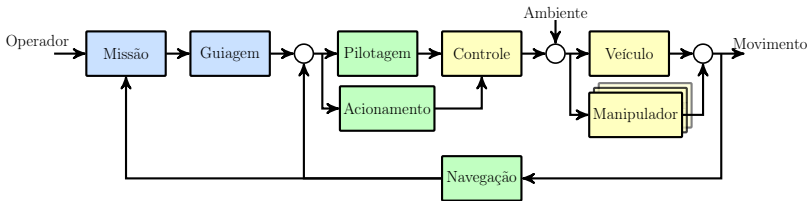


Figura 10. Diagrama de blocos de atividades de um UVMS em fases que envolvem intervenção

As atividades de missão e guiagem passam a considerar os ma-

nipuladores para a especificação das tarefas e geração de referências de movimento. As referências relativas ao veículo continuam sendo enviadas para a pilotagem, enquanto as dos manipuladores são enviadas para uma atividade equivalente que utiliza a dinâmica inversa dos manipuladores para determinar os esforços nas suas juntas. A atividade de controle recebe ambas as referências para acionar os atuadores do sistema. A atividade de navegação nessas fases também recebe os dados de sensores dos manipuladores para determinar as posturas dos efetuadores finais.

2.2 SISTEMAS DE UM UVMS

O projeto de um sistema subaquático de intervenção é de natureza multidisciplinar, uma vez que os seus requisitos envolvem diferentes áreas de conhecimento, como a mecânica, a eletrônica, a automação e a informática, entre outras. Diversos autores apresentam visões gerais sobre o projeto e os subsistemas de UUV e UVMS (Yuh, 2000; Antonelli, 2004; Antonelli; Fossen; Yoerger, 2008; Kinsey; Eustice; Whitcomb, 2006). Uma relação dos diferentes subsistemas e tecnologias atualmente empregadas, bem como uma análise dos requisitos desses veículos para diferentes finalidades, sobretudo a militar, é apresentada em (Button et al., 2009).

Com base na literatura, é possível relacionar os subsistemas de um UVMS. Observa-se que estes podem ser agrupados em três classes, independentemente de seus interrelacionamentos. A classe de *Comando* é formada por subsistemas de obtenção e processamento de informações, bem como a comunicação entre subsistemas. A classe de *Execução*, por sua vez, compreende os subsistemas que realizam as ações. A classe de subsistemas de *Estrutura* suporta os demais subsistemas do veículo. O diagrama da Figura 11 apresenta os subsistemas e seus elementos básicos, agrupando-os nas classes identificadas.

2.2.1 Comando

Os subsistemas classificados como de comando podem ser definidos como aqueles que gerenciam informações. Estas estão pré-armazenadas no sistema, como mapas e especificações da missão, ou são obtidas por sensores. Os subsistemas de comando devem ter capacidade de processamento suficiente para tratar um grande conjunto

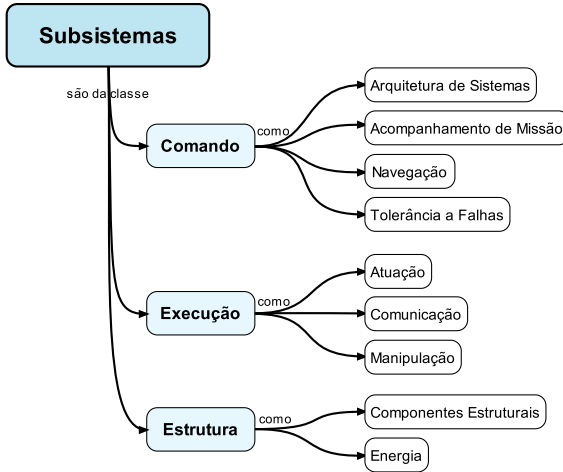


Figura 11. Subsistemas de um UVMS

de dados utilizando algoritmos muitas vezes complexos, com demandas de baixo tempo de resposta. Os resultados do processamento alimentam outros subsistemas de comando ou geram especificações de ações a serem realizadas pelos subsistemas de execução.

É comum existirem várias unidades de processamento de dados, cada qual com um conjunto específico de dados a tratar. A troca de informações entre as unidades é essencial para a execução da missão. Essa comunicação, bem como a definição de funções e relacionamentos entre subsistemas, também compreende um subsistema de comando.

Em relação aos sensores, existem normalmente dois conjuntos¹: os de *navegação* e os de *monitoramento*. Os sensores de navegação são usados para determinar a posição e orientação do UVMS no espaço. Os sensores de monitoramento, por sua vez, fornecem dados sobre as condições de operação do próprio sistema, para procurar manter o sistema dentro de condições de operação aceitáveis ou, em caso de problemas, acionar estratégias de tolerância a falhas.

O acompanhamento da missão também é parte do comando. Ele pode ser feita exclusivamente por operadores humanos ou assistido por sistemas de processamento de informação. A interação com o operador varia de acordo com o grau de autonomia de operação do sistema, variando entre a teleoperação completa até apenas supervisão.

¹Não são considerados aqui os sensores específicos para a execução de uma determinada missão.

A Figura 12 relaciona esses subsistemas, que são descritos a seguir.

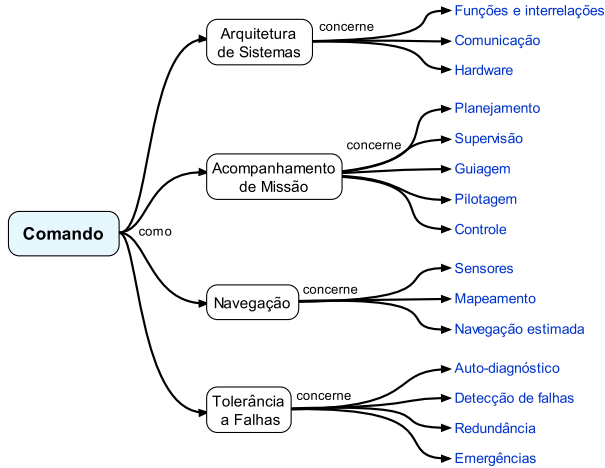


Figura 12. Subsistemas de comando de um UUV

2.2.1.1 Arquitetura do Sistema Embarcado

A maneira como os diversos subsistemas trocam informações com o acompanhamento da missão e entre si varia bastante. Existem diferentes formas de tratar essa comunicação, bem como de separar os subsistemas em *níveis*. Uma visão de algumas arquiteturas de sistemas é apresentada em (Valavanis et al., 1997). Abordagens mais recentes são analisadas em (Benjamin, 2007; Marani; Choi; Yuh, 2009; Antonelli; Fossen; Yoerger, 2008). Um exemplo de *software* livre desenvolvido para a coordenação e comunicação de diversos subsistemas, com enfoque distribuído em comunicação é o *Motion-Oriented Operating System (MOOS)* (Newman, 2009).

Em termos do *hardware* de processamento de informações, as opções também são muito variadas. Os sistemas atuais empregam tecnologias embarcadas baseadas em equivalentes do mercado de informática (arquitetura PC, principalmente), com sistemas operacionais de tempo real (Marani; Choi; Yuh, 2009; Yuh, 2000).

Embora se considere o sistema como embarcado, observa-se que em algumas arquiteturas, os subsistemas e seus componentes são tão

modularizados que podem ser físicos ou simulados por *software*, e podem estar embutidos de fato no UUV ou serem externos a ele (Marani; Choi; Yuh, 2009).

2.2.1.2 Acompanhamento da Missão

Existem diversos elementos inerentes à execução de missões, que muitas vezes são separados em níveis hierárquicos. Fossen (1994) define dois elementos básicos à qualquer missão: *guiagem* e *controle*.

A *guiagem* é a ação de determinar o curso e a velocidade a serem seguidos pelo veículo em relação a um referencial. Ela envolve *planejamento* e *geração de caminhos e trajetórias* para executar a tarefa especificada por um operador. Para tanto, ela depende de informações de localização e orientação do sistema, bem como velocidades, determinadas pelo subsistema de navegação. Além disso, objetivos secundários devem ser considerados, como o evitamento de obstáculos. No caso dos ROV, essa função é responsabilidade do operador que controla o veículo. À medida que o sistema passa a ter maior autonomia, a *guiagem* passa a ser responsabilidade dos sistemas do veículo. Nos AUV, o planejamento da tarefa e a geração global/local de referências deve ser feito totalmente pelo sistema, a partir das informações disponíveis e da especificação dos pontos a serem alcançados (Antonelli; Fossen; Yoerger, 2008).

O *controle* é a ação de definir e aplicar as forças e momentos necessários para que o veículo execute as trajetórias designadas enquanto procura manter o sistema estável. Diferentes ações de controle podem ocorrer, de acordo com o tipo de UUV e a natureza da tarefa a ser realizada (Antonelli; Fossen; Yoerger, 2008). A cinemática e dinâmica de UUV, bem como a interação com o ambiente, tornam o controle um desafio para pesquisadores, com vasta literatura sobre diferentes abordagens para a sua solução. São citadas como características do problema de controle (Yuh, 1990; Fossen, 1994; Yuh, 2000; Antonelli, 2004):

- Modelo dinâmico não-linear e fortemente acoplado, muitas vezes com parâmetros variantes no tempo;
- Dinâmica não-linear e de difícil modelagem dos propulsores;
- Modelo dinâmico das superfícies de controle (lemes, barbatanas);
- Redundância em UVMS;

- Dificuldade de identificação dos coeficientes hidrodinâmicos;
- Perturbações do ambiente, como correntes marinhas.

Veículos totalmente atuados nos seis graus de liberdade através de propulsores é o arranjo mais comum em ROV. Nos UVMS, a interação do manipulador com o veículo exige compensação dos esforços gerados pelo primeiro de forma a manter o veículo na postura desejada, o que justifica a adoção desse tipo de arranjo. Em relação a esses tipos de veículos, uma discussão sobre diferentes técnicas de controle é feita em (Antonelli, 2006), sem, no entanto, procurar esgotar o assunto, que apresenta vasta quantidade de trabalhos.

A *supervisão* da missão é feita através de interfaces gráficas em computadores situados na base, que podem apresentar informações de localização do veículo, dados de sensores e imagens de câmeras e sonares. No caso de veículos totalmente teleoperados, a interface bidirecional de comunicação é feita com telas e instrumentos para entrada de dados, como manches e teclados, sendo por vezes feito o controle por mais de um operador (por exemplo, no caso de manipulação). Na medida em que os UUV passam a ter maior autonomia, a interface com o operador passa a ter caráter mais supervisório e menos interativo, apresentando informações em nível mais alto sobre a situação da execução da tarefa. A autonomia também possibilita utilizar comandos de mais alto nível com o veículo, com a especificação de tarefas usando linguagens de programação ou interação gráfica (Benjamin, 2007; Antonelli; Fossen; Yoerger, 2008; Marani; Choi; Yuh, 2009).

2.2.1.3 Navegação

O subsistema de navegação tem como finalidade básica determinar a posição e orientação do sistema subaquático a cada instante da missão. Nos UVMS, inclui-se a determinação da postura dos efetuidores finais dos manipuladores. Essas informações são imprescindíveis para que os demais subsistemas de comando possam definir e aplicar os movimentos no sistema de forma a executar as tarefas especificadas. O uso de GPS (*Global Positioning System*) é viável em veículos aquáticos de superfície, mas não para o caso de veículos subaquáticos, devido à atenuação dos sinais eletromagnéticos na água.

Para estimar a postura do veículo, são utilizadas diferentes técnicas que fazem uso de conjuntos redundantes de sensores, conhecidas genericamente como fusão sensorial (Antonelli, 2004). Na Tabela 6 es-

tão relacionados alguns dos sensores empregados e quais as grandezas medidas por cada tipo de sensor.

Tabela 6. Sensores empregados nos sistemas de navegação de UUV

Sensor	Variável medida
Bússola	Orientação absoluta
Giroscópio	Velocidade angular
Pressão	Profundidade
Corrente	Velocidade relativa entre corrente e veículo
Sonar frontal e lateral	Distância de obstáculos
Sonar vertical	Distância do fundo
<i>Doppler velocity log</i>	Velocidade relativa entre veículo e fundo
Unidade de medição inercial	Aceleração linear e angular; velocidade angular
Sistemas de visão	Posição e velocidade relativas ao ambiente
Referenciais acústicos	Posição relativa a <i>faróis acústicos</i> fixos
GPS ²	Posição absoluta

A estimação de posição e orientação é um tema bastante ativo de pesquisas, que compreendem tanto o desenvolvimento de sensores quanto técnicas de análise dos dados por eles fornecidos e sua fusão. Uma visão geral dessa linha de pesquisa é apresentado em (Kinsey; Eustice; Whitcomb, 2006). As técnicas de fusão sensorial, como por exemplo a *SLAM* (*Simultaneous Location And Mapping*) são comuns também para a robótica móvel terrestre e aérea, sendo abordadas sucintamente em (Siciliano; Khatib, 2008).

2.2.1.4 Tolerância a Falhas

UUV são equipamentos caros operando em ambientes insalubres e perigosos. A *detecção de falhas* é um requisito importante desse tipo de equipamento para dar subsídios à tomada de decisão ao operador, no caso de ROV, ou à alguma rotina de um sistema autônomo (nos AUV), avaliando se a operação pode continuar, contornando a falha, ou se esta pode comprometer a missão e deve ser interrompida. No caso dos veículos subaquáticos, uma decisão errada pode levar à perda do veículo, com casos relatados na literatura.

As estratégias mais comuns de *tolerância a falhas* compreendem algum tipo de redundância. Os componentes redundantes podem operar em paralelo com os principais ou ficar em situação de sobreaviso,

²Embora não possa ser considerado como um sensor de uso constante, o GPS pode ser usado para definir uma posição inicial de referência, na superfície, para a partir desta se estimar as posições no tempo.

sendo acionados em caso de detecção da falha do componente ativo. Para tanto, deve-se ter a capacidade de monitorar constantemente o sistema. Esse monitoramento é baseado em modelos do veículo, onde são avaliadas variações de parâmetros de operação. Deve-se então decidir quando uma variação corresponde a uma falha.

No caso de falha irrecuperável, devem-se acionar procedimentos de recuperação do sistema, seja ativando algum programa de retorno à base ou à superfície, seja ativando localizadores para possibilitar a posterior recuperação do veículo.

No caso de sistemas autônomos, tais decisões devem ser tomadas por softwares dedicados à detecção, recuperação e tolerância a falhas, o que é um campo de pesquisa que não se restringe à veículos subaquáticos. Uma visão geral é apresentada em (Antonelli, 2006).

2.2.2 Execução

A classe de execução é formada pelos subsistemas que realizam os movimentos no UVMS, como a propulsão do veículo e o acionamento dos manipuladores. O subsistema de comunicação também é enquadrado dentro dessa classe. As características desses sistemas, listados na Figura 13, são apresentadas resumidamente a seguir.

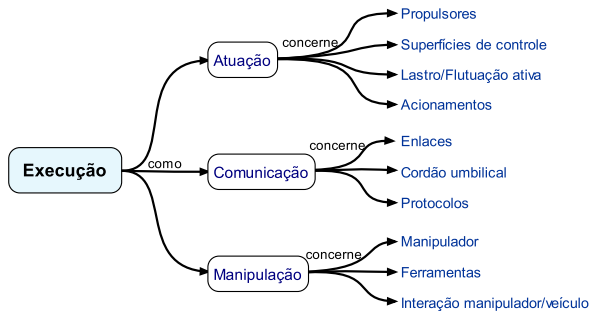


Figura 13. Subsistemas de execução de um UUV

2.2.2.1 Atuação

O movimento próprio de um UUV depende de propulsores. Em veículos de cruzeiro, os propulsores são instalados de forma a garantir

translação na direção longitudinal do veículo, enquanto o movimento em outras direções é feito por superfícies de controle como barbata-nas e lemes, aproveitando os efeitos de arrasto e sustentação causados pelo movimento relativo entre o fluido e o corpo. No lugar de propulsores a hélice, alguns veículos usam propulsores vetoriais (hidrojatos), enquanto alguns veículos de cruzeiro para missões de longa duração (*gliders*) usam grandes superfícies de controle e controle de flutuabilidade para aproveitar a energia potencial para se deslocarem e pouparem energia de baterias (Button et al., 2009). Também são citados sistemas de controle de flutuação para o movimento vertical de veículos de exploração (Stone Aerospace, 2012).

Para sistemas de intervenção, porém, não se considera atuação através de superfícies de controle, pois estas não são efetivas em velocidades baixas/nulas. Para realizar a atuação completa, esses sistemas usam um conjunto de propulsores de forma a garantir controle sobre movimentos translacionais e rotacionais em todas as direções. O controle preciso do movimento do veículo depende da posição dos propulsores no seu corpo e das suas influências em cada grau de liberdade. Além disso, a modelagem dos propulsores é função de diversas variáveis de projeto e do ambiente, apresentando uma relação não-linear entre torque aplicado e velocidade de saída. Esta é apontada como causa de ciclos limite no posicionamento estacionário (Antonelli, 2004).

Além da movimentação do veículo, a atuação compreende também o acionamento de outros subsistemas, como o de manipuladores (tratado em subseção própria), lançamento de bóias/marcadores, liberação de cargas e outras ações físicas que porventura sejam necessárias para a execução de uma operação ou recuperação do veículo.

2.2.2.2 Comunicação

A comunicação do UUV com operadores depende muito do tipo de missão e do grau de autonomia. Em ROV, é possível ter comunicação bidirecional de alta velocidade, com enlaces físicos usando cabos coaxiais ou fibra ótica (Yuh, 2000). Já com os AUV a comunicação é feita por enlace acústico, através de modems com baixa largura de banda, o que limita sobremaneira a taxa de comunicação e a quantidade de dados transmitidos. Algumas alternativas consideram o uso de bóias de comunicação intermediárias entre veículo e mundo, ou mesmo comunicação por *laser* e redes de satélites (Button et al., 2009).

Acima da camada de enlace, os protocolos de comunicação en-

tre o veículo e o controle da missão variam entre os proprietários e o TCP/IP (Marani; Choi; Yuh, 2009). As limitações de velocidade, porém, limitam os tipos de informações trafegados. Enquanto a comunicação por cabo possibilita a transmissão de vídeo em tempo real, a comunicação acústica restringe o tráfego de dados a informações de telemetria, reprogramações e, à curta distância, transmissão de vídeo de baixa qualidade (Yuh, 2000).

2.2.2.3 Manipulação

Para executar operações de intervenção, é necessário que o veículo tenha alguma forma de interagir com o meio, o que é usualmente realizado através de manipuladores. Estes são responsáveis por operações de segurar objetos e posicionamento de ferramentas, exatamente como no caso de manipuladores utilizados no meio industrial. Embora nos ROV seja comum haver manipuladores, estes ainda são objeto de pesquisa nos AUV. No primeiro caso, os manipuladores são teleoperados, enquanto no segundo, eles devem operar de forma autônoma, como o veículo.

Os manipuladores podem ser movimentados por atuadores elétricos ou hidráulicos, dependendo do seu porte e da carga a ser manipulada. Além de sensores para observar o movimento (posição, velocidade, aceleração) das juntas e, através da cinemática direta, do efetuator final, sensores de força/torque são importantes para o controle da força de interação deste com o meio. Além disso, sensores de proximidade podem ser necessários para o evitamento de obstáculos.

A complexidade dos modelos cinemático e dinâmico dos sistemas veículo-manipulador fazem com que esses sistemas ainda sejam teleoperados. Embora vários modelos de veículos subaquáticos sejam apresentados na literatura, existem poucos trabalhos analisando modelos de sistemas de intervenção. Existe uma referência principal nessa linha de pesquisa, que trata da modelagem cinemática e dinâmica de UVMS, sua interação com o meio, e análise de estratégias de controle (Antonelli, 2006). Os UVMS autônomos ainda estão no estágio de pesquisa e desenvolvimento, com os primeiros resultados de campo e aplicações comerciais surgindo recentemente (Marani; Choi; Yuh, 2009).

2.2.3 Estrutura

Os subsistemas da classe estrutura são os que suportam os demais subsistemas. Como ilustra a Figura 14, um desses subsistemas é o dos componentes estruturais, que compreendem o casco, os vasos de pressão e a estrutura de sustentação do UUV. O outro subsistema é o de geração/fornecimento de energia.

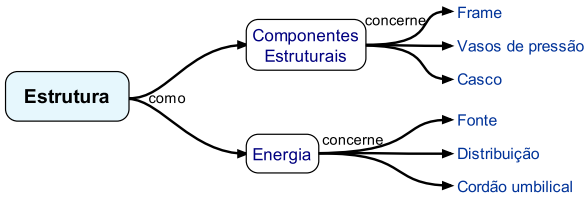


Figura 14. Subsistemas de estrutura de um UUV

2.2.3.1 Componentes Estruturais

Os elementos estruturais de um UUV devem levar em consideração os efeitos do meio, bem como o tipo de operação à que se destinam. Na água, a pressão sobre o veículo aumenta linearmente com a profundidade, em uma razão aproximada de 1 atm (≈ 100 kPa) para cada 10m de coluna d'água (Button et al., 2009). Além disso, o ambiente marinho é altamente corrosivo. No caso de veículos de cruzeiro, particularmente, o efeito hidrodinâmico é um elemento de projeto importante, influenciando a forma do casco.

O *frame* de um UUV caracteriza a sua rigidez, sendo a estrutura onde os módulos do veículo são fixados (atuadores, sensores, eletrônica embarcada, manipulador). No seu projeto deve-se considerar também as forças sobre ele exercidas pelos atuadores e pelo manipulador, além da reconfigurabilidade do UUV de acordo com a missão a ser executada.

Os vasos de pressão contém os elementos sensíveis à água e à pressão como os eletro/eletrônicos. Os vasos podem ser parte do casco ou módulos separados. Neste último caso, eles são montados na estrutura de sustentação do veículo de acordo com as demandas da missão. Alguns mecanismos de fixação permitem a rápida desmontagem dos módulos, o que é útil no caso de manutenção, ou para liberar pesos para melhorar a flutuabilidade em caso de emergência.

Em relação às características estruturais, há vários materiais que suportam grandes pressões, resistem à corrosão e apresentam baixo peso, sendo destacados os materiais compostos, ligas de alumínio e ligas de titânio.

Por fim, o projeto do casco deve minimizar o arraste hidrodinâmico, que pode influenciar mesmo veículos em baixa velocidade. Atualmente, várias técnicas de avaliação por simulação e experimental são utilizadas para isso (Fossen, 1994; Yuh, 2000; Button et al., 2009).

2.2.3.2 Energia

Este é um subsistema que se diferencia claramente nos ROV e AUV. Enquanto nos primeiros pode-se considerar que o suprimento de energia é ilimitado, uma vez que esta é fornecida pelo cordão umbilical, nos AUV a autonomia de operação é função da energia armazenada em baterias.

O fornecimento de energia deve considerar a alimentação de todos os subsistemas do veículo, da eletrônica embarcada aos sistemas de atuação e intervenção, com controle de consumo uniforme e tolerante a problemas elétricos. Por isso, o desenvolvimento de tecnologias que consumam menos energia bem como de fontes mais eficientes é essencial para aumentar o tempo de operação contínua dos veículos (Whitcomb, 2000).

Isso também se aplica aos ROV, pois o consumo de energia está diretamente relacionado à espessura de cabo necessária para o seu fornecimento, o que afeta a dinâmica do veículo pelos esforços devidos à interação entre o cabo e a água. Por esse motivo, alguns projetos utilizam o fornecimento híbrido de energia, tanto pelo cordão umbilical quanto por baterias no próprio veículo. Alguns ROV de pesquisa empregam apenas baterias, usando o cordão umbilical apenas para comunicação, que assim tem a sua espessura bastante reduzida (Yuh et al., 1998; Inovação Tecnológica, 2009).

Além das baterias de diferentes tecnologias, estão sendo desenvolvidos trabalhos com células a combustível e mesmo com geradores usando diesel como combustível ou pó de alumínio. Essa linha de pesquisa tem se beneficiado diretamente dos estudos para o desenvolvimento de tecnologias eficientes para armazenamento e geração de energia para veículos automotivos elétricos (Whitcomb, 2000; Button et al., 2009).

2.2.4 Autonomia

Embora não seja um subsistema, a autonomia é uma característica que influencia diretamente as decisões de projeto de UUV. Não existe uma definição rigorosa, porém, para o grau de autonomia de um UUV. Alguns veículos, por exemplo, apesar de vinculados por um cordão umbilical, tem certa autonomia para executar algumas operações a partir de definições de *alto nível* de um operador. Alguns autores cunharam o termo *semiautonomia* para se referirem a este tipo de interação operador-veículo, com o objetivo de evoluir gradativamente os sistemas de missão de forma que as tarefas do operador sejam de nível cada vez mais alto, à medida que os sistemas do UUV tornam-se mais capazes de gerenciar os aspectos de operação anteriormente de responsabilidade humana (Marani; Choi; Yuh, 2009; Yuh et al., 1998).

O *Escritório de Pesquisa Naval* americano (*Office of Naval Research*, ou *ONR*) define seis níveis de autonomia (Button et al., 2009):

Operado por humano Veículo totalmente teleoperado, em baixo nível, onde a única comunicação do sistema com o operador é para retornar dados de sensores;

Assistido por humano O sistema tem alguma capacidade de realizar tarefas em paralelo com a operação humana, embora nada que não envolva a operação remota;

Delegado por humano O sistema pode executar algumas tarefas a partir de sua delegação pelo operador, como controle dos atuadores, pilotagem e outras operações de baixo nível, que podem ser assumidas imediatamente pelo operador, caso desejado;

Supervisionado por humano O sistema pode realizar grande variedade de atividades a partir de decisões de alto nível do operador. Pode receber especificações de movimentos a executar, e a partir destes, assumir guiagem e pilotagem, por exemplo;

Iniciativa mista As decisões de alto nível podem ser tomadas tanto pelo operador quanto pelos sistemas. Por exemplo, pode-se definir uma localização a alcançar, e tanto o operador quanto o sistema podem definir a trajetória a ser executada;

Autonomia plena O sistema não requer intervenção de um operador, após recebida a tarefa a ser executada, tendo de tomar todas as

decisões e mesmo sem haver comunicação com operador ou base até o fim da execução da tarefa.

Esta classificação, embora não seja um padrão, pode ilustrar a direção da evolução dos sistemas autônomos.

A autonomia de sistemas de intervenção subaquática é uma linha de pesquisa que ainda tem relativamente poucos trabalhos na literatura, ao menos se comparada com as pesquisas com veículos autônomos subaquáticos de cruzeiro. A tecnologia desses veículos é considerada *es-tável* o suficiente para serem usados em missões de exploração (Button et al., 2009). Vários resultados de pesquisas sobre AUV de cruzeiro são aproveitados para o estudo dos sistemas veículo-manipulador, em linhas como autonomia de operação, navegação, guiagem e interação do veículo com o meio aquático.

2.3 TRABALHOS RELACIONADOS AOS UVMS

Alguns temas de pesquisa diretamente relacionados aos UVMS são descritos a seguir. Junto à descrição, são citados alguns trabalhos relevantes que foram consultados no desenvolvimento desta tese.

Visões Gerais e Tendências Análises do panorama da robótica subaquática, identificando tecnologias, aplicações e tendências são periodicamente publicadas, servindo como guia para o desenvolvimento de novas pesquisas. Entre esses trabalhos, são citados (Yuh, 2000; Whitcomb, 2000; Antonelli; Fossen; Yoerger, 2008). Uma revisão sobre a tecnologia de sensores é feita em (Kinsey; Eustice; Whitcomb, 2006). Uma análise do mercado, de tecnologias maduras e em desenvolvimento é apresentada em (Button et al., 2009).

Manipuladores no meio aquático A interação com o meio líquido é um fator relevante no uso dos manipuladores subaquáticos. Alguns resultados sobre os esforços resultantes são extraídos diretamente da modelagem de um corpo rígido movimentando-se em um fluido, que é bastante explorado na literatura sobre dinâmica de veículos subaquáticos. Pesquisas sobre modelagem, desenvolvimento e experimentação de manipuladores operando imersos são reportadas em (McLain; Rock; Lee, 1995; Tarn; Yang, 1997; Lane et al., 1999; Sagara et al., 2001).

Modelagem cinemática e dinâmica Em grande parte da literatura, a modelagem cinemática dos UVMS considera modelos separados

para o veículo e para o manipulador. No primeiro, faz-se a análise de um corpo rígido livre no espaço, enquanto no segundo utiliza-se a notação de Denavit-Hartenberg para a obtenção da cinemática e da dinâmica. Os modelos são posteriormente combinados para a obtenção do movimento do efetuador final do manipulador em relação a um referencial inercial.

Em relação ao veículo, desenvolve-se a cinemática e a dinâmica detalhadamente em (Fossen, 1994). Uma revisão da dinâmica de UUV pequenos e médios é apresentada em (Ananthakrishnan; Decron, 2000). Análises de modelos de UVMS considerando veículo e manipulador são feitas em (Schjølberg; Fossen, 1994; Hsu et al., 2000).

Uma abordagem alternativa empregando a teoria dos helicoides para a análise cinemática de UVMS integrando veículo e manipulador foi proposta por Santos (Santos, 2006; Santos et al., 2006). Nela, tratou-se do caso de um veículo subaquático com um manipulador acoplado. Uma abordagem de análise cinemática por quaternions duais é apresentada em (Oliveira, 2011).

Controle de posição e força O controle de UVMS tem sido pesquisado por muitos autores. A dificuldade da obtenção de um modelo dinâmico preciso para estes sistemas, bem como da interação deste com o ambiente leva a diferentes abordagens para a solução deste problema. Antonelli avaliou o uso de diversas estratégias de controle em diferentes sistemas veículo-manipulador. Além de vários artigos, os resultados foram reunidos em um livro que pode ser considerado como uma referência para UVMS (Antonelli, 2006). Entre outros trabalhos sobre estratégias de controle cinemático e dinâmico, citam-se (Cui; Podder; Sarkar, 1999; Fraisse et al., 2000; de Wit; Diaz; Perrier, 2000; Sarkar; Podder, 2001; Xu et al., 2007; Soyly; Buckham; Podhorodeski, 2010).

Projeto, construção e experimentação Embora UVMS teleoperados sejam uma tecnologia em uso, as pesquisas sobre autonomia ainda empregam simulações. Alguns projetos de desenvolvimento de veículos para experimentação e uso em produção são o SAUVIM (Yuh et al., 1998; Marani; Choi; Yuh, 2009), o ALIVE (Antonelli; Fossen; Yoerger, 2008), o ROBHAZ (Richardson; Woodward; Billingham, 2002) e o RAUVI (De Novi et al., 2010; Ridao, P. and Sanz, P. J. and Oliver, J., 2011). Além desses, pode-se ainda citar o desenvolvimento de um veículo para exploração espacial (Stone Aerospace, 2012).

Cooperação A cooperação de UVMS ainda é um tema recente, com uma visão geral disponível em (Padir; Koivo, 2003; Padir, 2005). Para veículos de cruzeiro, a cooperação também é tratada (Bishop; Stilwell, 2001; Bishop, 2003, 2008). Uma arquitetura de cooperação entre UUV e Veículos não tripulados de superfície, ou USV (do inglês *Unmanned Surface Vehicles*) é abordada em (Benjamin, 2007).

A robótica subaquática é um campo de pesquisa que tem crescido no Brasil, em parte motivado pelo interesse na exploração de petróleo e gás *offshore*. Outra motivação importante é a grande extensão do mar territorial brasileiro, que ainda é pouco conhecido. Os primeiros estudos sobre modelagem, simulação e controle datam da década de 80 (Dominguez, 1989). No controle de UUV são citados diversos trabalhos, como (Hsu et al., 2000; Cunha, 1992; De Souza; Maruyama, 2007; Kuhn, 2011), por exemplo. Aspectos de projeto e construção de UUV são objeto de estudo de diferentes trabalhos (Tavares, 2003; Barros; Soares, 2002; Botelho et al., 2003; Moraes, 2005; Centeno, 2007; Floriani; Dias; Rocha, 2011). A cinemática de UUV tem sido abordada em pesquisas como (Santos et al., 2006; Oliveira, 2011; Rocha; Dias, 2010a).

Apresentou-se neste capítulo os cenários e desafios dos sistemas subsaquáticos de intervenção. A partir dessa compilação de materiais da literatura, observa-se que a autonomia desses sistemas é o objetivo da maioria das pesquisas, bem como uma demanda de mercado.

Este trabalho enquadra-se no desenvolvimento da autonomia dos UVMS. Para tanto, concentra-se na fase de *intervenção das missões*, tratando do *planejamento de movimento* do sistema de intervenção, uma das atribuições da atividade de guiagem. A partir de uma descrição do movimento desejado para o efetuator final e do veículo (se necessário), o planejamento de movimento deve gerar as referências para os atuadores do sistema de forma que a tarefa especificada seja realizada. Para tanto, a análise cinemática desses sistemas é necessária. A partir desta, devem ser definidas estratégias de resolução de tarefas que contemplem o UVMS, os objetos a sofrerem intervenção e o meio. Estes são os temas dos capítulos seguintes.

3 CINEMÁTICA DE UM UVMS

Este capítulo apresenta uma síntese da modelagem cinemática de sistemas veículo-manipulador subaquáticos apresentada na literatura. A cinemática de um UVMS deve considerar o movimento do veículo e dos manipuladores a ele vinculados. A abordagem mais frequente na literatura utiliza uma modelagem composta pelo veículo como um corpo rígido livre no espaço e do manipulador através da notação Denavit-Hartenberg. A combinação dos modelos do veículo e do manipulador pode ser vista como se este tivesse uma base móvel.

Além dessa modelagem, descreve-se também neste capítulo a cinemática de UVMS através da teoria dos helicoides. Esta provê um tratamento unificado à cinemática de UVMS considerando-o uma cadeia cinemática.

3.1 CINEMÁTICA DO VEÍCULO

Na análise cinemática, um veículo aquático é tratado como um corpo rígido cujo movimento se dá em seis graus de liberdade (três translacionais e três rotacionais). Sua *postura* é definida em relação a um *sistema de coordenadas inercial* costumeiramente associado à Terra, aqui denominado $\mathbf{O}-\mathbf{xyz}$.

A postura do veículo é determinada pela posição e orientação do sistema de coordenadas do veículo $\mathbf{O}_v-\mathbf{x}_v\mathbf{y}_v\mathbf{z}_v$ em relação ao referencial inercial e fixado a um ponto qualquer do corpo rígido do veículo¹. Os eixos do sistema são convencionalmente alinhados com as direções longitudinal (\mathbf{x}_v), transversal (\mathbf{y}_v) e normal (\mathbf{z}_v) do veículo, coincidindo com os eixos principais de inércia da maioria das geometrias usadas em veículos aquáticos (Fossen, 1994).

Na área naval, é usual adotar a notação definida pela *Society of Naval Architects and Marine Engineers* (SNAME) para postura e movimento. Ela é apresentada na Tabela 7. Segundo essa notação², o movimento linear se dá segundo as direções de *avanço* (*surge*), *balanço* (*sway*) e *afundamento* (*heave*), enquanto o movimento angular ocorre segundo

¹Muitas vezes, escolhe-se o centro de massa do veículo ou outro ponto que simplifique as suas equações de movimento.

²Os símbolos foram levemente modificados neste texto para identificar as variáveis do veículo no contexto dos UVMS. Além disso, o símbolo para velocidade angular em torno do eixo \mathbf{y} do veículo foi trocado de q_v para s_v a fim de evitar conflito com a notação usada em cadeias cinemáticas.

os eixos de *rolagem(roll)*, *arfagem(pitch)* e *guinada(yaw)* (SNAME, 1950; Tavares, 2003).

Tabela 7. Notação SNAME para o movimento de veículos aquáticos

Postura		Eixo		Velocidade		
Posição	η_1	x_v	<i>avanço(surge)</i>	u_v	ν_1	Linear
		y_v	<i>balanço(sway)</i>	v_v		
		z_v	<i>afundamento(heave)</i>	w_v		
Orientação	η_2	ϕ_v	<i>rolagem(roll)</i>	p_v	ν_2	Angular
		θ_v	<i>arfagem(pitch)</i>	s_v		
		ψ_v	<i>guinada(yaw)</i>	r_v		

A definição da postura do veículo com base nos dois sistemas de coordenadas é mostrada na Figura 15. Nela, o veículo é representado por um elipsoide.

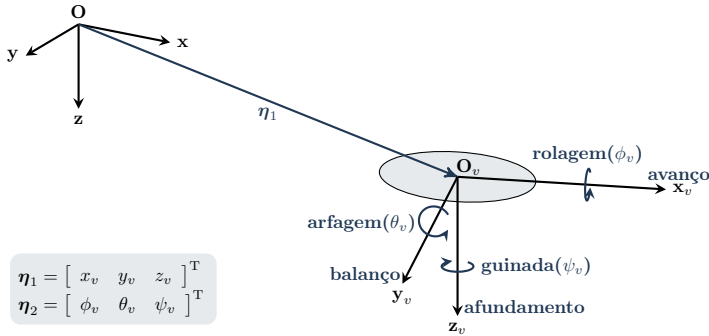


Figura 15. Definição da postura do veículo

A posição do veículo é definida pelo vetor η_1 da posição da origem do referencial a ele vinculado (denominado *referencial do veículo*) em relação ao referencial inercial, como expresso na Equação 3.1. A orientação é convencionalmente descrita por ângulos de Euler RPY (do inglês *Roll-Pitch-Yaw*) segundo o referencial inercial como na Equação 3.2 (Fossen, 1994; Antonelli, 2006). Esses ângulos descrevem a sequência de deslocamentos angulares necessária para que um sistema de coordenadas móvel inicialmente alinhado com o referencial inercial se alinhe com o referencial do veículo.

$$\boldsymbol{\eta}_{1[3 \times 1]} = \begin{bmatrix} x_v & y_v & z_v \end{bmatrix}^T \quad (3.1)$$

$$\boldsymbol{\eta}_{2[3 \times 1]} = \begin{bmatrix} \phi_v & \theta_v & \psi_v \end{bmatrix}^T \quad (3.2)$$

Assim, a postura do veículo é descrita pelo vetor $\boldsymbol{\eta}_v$ definido na Equação 3.3.

$$\boldsymbol{\eta}_{v[6 \times 1]} = \begin{bmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{bmatrix} \quad (3.3)$$

Segundo a notação SNAME, a velocidade do veículo é definida como mostrado na Equação 3.4, sendo expressa segundo o referencial do veículo. $\boldsymbol{\nu}_1$ é a velocidade linear da origem do referencial do veículo em relação à origem do referencial inercial. $\boldsymbol{\nu}_2$ é a velocidade angular do veículo em relação ao referencial inercial, expresso no referencial do veículo.

$$\boldsymbol{\nu}_{v[6 \times 1]} = \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix} = \begin{bmatrix} u_v & v_v & w_v & p_v & s_v & r_v \end{bmatrix}^T \quad (3.4)$$

A relação entre a velocidade da Eq. 3.4 e a derivada temporal da postura do veículo é estabelecida pela matriz Jacobiana \mathbf{J}_v definida na Equação 3.5, que depende da orientação do referencial do veículo,

$$\boldsymbol{\nu}_v = \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix} = \mathbf{J}_{v[6 \times 6]} \dot{\boldsymbol{\eta}}_{v[6 \times 1]} = \begin{bmatrix} {}^v \mathbf{R}_I(\boldsymbol{\eta}_2) & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_o(\boldsymbol{\eta}_2) \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} \quad (3.5)$$

onde ${}^v \mathbf{R}_I$ é a matriz de rotação que expressa a transformação de orientação do referencial inercial para o referencial do veículo e \mathbf{J}_o é a matriz Jacobiana que relaciona a velocidade angular com a derivada temporal de orientação. Essas matrizes são definidas nas Equações 3.6 e 3.7, onde c . e s . correspondem às funções trigonométricas cosseno e seno, respectivamente (Antonelli, 2006).

$${}^v \mathbf{R}_I(\boldsymbol{\eta}_2)_{[3 \times 3]} = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & s_\phi c_\theta \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\phi c_\theta \end{bmatrix} \quad (3.6)$$

$$\mathbf{J}_o(\boldsymbol{\eta}_2)_{[3 \times 3]} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \quad (3.7)$$

A determinação da postura do veículo em um dado instante a

partir de uma postura inicialmente conhecida e da integração temporal da velocidade é bastante utilizada em navegação, particularmente em veículos subaquáticos, onde não é possível obter dados de sistemas GPS, sendo conhecido como *navegação estimada* (em inglês, *dead reckoning*).

A integração da velocidade angular expressa segundo o referencial do veículo não tem significado físico (Fossen, 1994). Assim, integra-se a derivada temporal de orientação, obtida pela relação inversa à da Eq. 3.5 (Antonelli, 2006). A derivada temporal de postura do veículo $\dot{\eta}_v$ é então definida como na Equação 3.8,

$$\dot{\eta}_{v_{[6 \times 1]}} = \begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \mathbf{J}_v^{-1} \nu_v = \begin{bmatrix} {}^v \mathbf{R}_I^{-1}(\eta_2) & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_o^{-1}(\eta_2) \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \quad (3.8)$$

onde a forma assumida pela inversa da matriz \mathbf{J}_v é devido a sua forma bloco diagonal. Por ser ortogonal, a matriz de rotação tem sua inversa igual a sua transposta (Siciliano et al., 2009). A inversa da matriz Jacobiana de orientação é descrita na Equação 3.9, de onde se observa que para $\theta = \{(2i + 1)\pi/2 \mid i \in \mathbb{N}\}$ a matriz é singular.

$$\mathbf{J}_o^{-1}(\eta_2) = \frac{1}{c_\theta} \begin{bmatrix} 1 & s_\phi s_\theta & c_\phi s_\theta \\ 0 & c_\phi c_\theta & -s_\phi c_\theta \\ 0 & s_\phi & c_\phi \end{bmatrix} \quad (3.9)$$

Outra representação de orientação utilizada, principalmente em estratégias de controle, é a baseada em quaternions, cuja característica principal é a não existência de singularidade em sua representação. O uso de quaternions para representar orientação é detalhado em (Tavares, 2003; Antonelli, 2006; Oliveira, 2011). Neste texto, será utilizada apenas a representação por ângulos RPY, por ainda ser a mais empregada na área naval.

3.2 CINEMÁTICA DO SISTEMA VEÍCULO-MANIPULADOR

Nos sistemas veículo-manipulador, as tarefas de intervenção são especificadas em função do movimento do efetuador final do manipulador, que fará contato com o meio, movimentará objetos e utilizará ferramentas. O modelo cinemático de manipuladores empregados em UVMS é similar aos robôs manipuladores utilizados na indústria.

Na literatura, a cinemática de manipuladores emprega principalmente a notação de Denavit-Hartenberg como, por exemplo, em (Siciliano et al., 2009). Nela, o modelo é função das variáveis das juntas e

suas derivadas expressas em relação à sua base. Por isso, a cinemática de UVMS também costuma empregar essa notação, como é descrito nesta seção.

Nos UVMS, a base do manipulador é o veículo. Assim, o manipulador sofre os efeitos de estar sobre uma base móvel, onde a velocidade da base varia ao longo do tempo. As velocidades e acelerações da base são propagadas através dos elos, como é mostrado na modelagem desenvolvida em (Antonelli, 2006), e tratado em outros trabalhos sobre o uso de manipuladores subaquáticos, como por exemplo, em (Soylu; Buckham; Podhorodeski, 2010).

Dado que um UVMS é formado por um veículo e um manipulador serial, como o mostrado na Figura 16, a postura do efetuador final em relação ao referencial inercial $\boldsymbol{\eta}_e$ é função da postura do veículo $\boldsymbol{\eta}_v$ e das variáveis de junta \mathbf{q} da cadeia cinemática do manipulador como é definido na Equação 3.10,

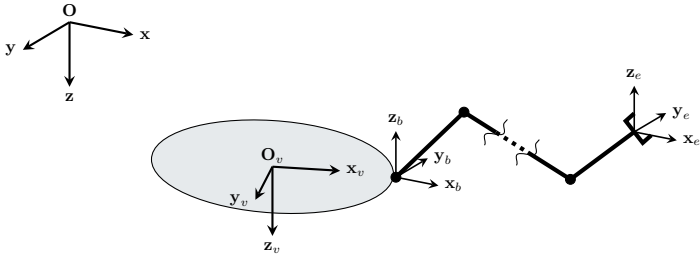


Figura 16. Representação cinemática de um UVMS e seus sistemas de coordenadas

$$\boldsymbol{\eta}_{e[6 \times 1]} = \begin{bmatrix} \boldsymbol{\eta}_{e1} \\ \boldsymbol{\eta}_{e2} \end{bmatrix} = \mathcal{K}(\boldsymbol{\eta}_v, \mathbf{q}) \quad (3.10)$$

onde \mathcal{K} representa a função cinemática direta do sistema, $\boldsymbol{\eta}_{e1}$ é a posição do efetuador final e $\boldsymbol{\eta}_{e2}$ é sua orientação, sendo usados os ângulos RPY neste texto para manter uma representação análoga à da notação SNAME para veículos aquáticos.

Segundo o referencial inercial, a derivada temporal da postura do efetuador final $\dot{\boldsymbol{\eta}}_e$ é definida pela Equação 3.11 (Santos, 2006; Antonelli, 2006),

$$\dot{\boldsymbol{\eta}}_{e[6 \times 1]} = \mathbf{J}_{e,v}({}^I \mathbf{R}_v, \mathbf{q})_{[6 \times 6+n]} \boldsymbol{\zeta}_{[6+n \times 1]} \quad (3.11)$$

onde $\mathbf{J}_{e,v}$ é o Jacobiano que relaciona o vetor de velocidades ζ e a derivada temporal $\dot{\eta}_e$. ζ é o vetor de velocidades do sistema formado pelas velocidades do veículo e das juntas do efetuador final como é definido na Equação 3.12.

$$\zeta_{[6+n \times 1]} = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \dot{\mathbf{q}} \end{bmatrix} \quad (3.12)$$

A velocidade do efetuador final expressa no referencial do próprio efetuador é definida na Equação 3.13, onde o Jacobiano \mathbf{J}_e é definido de forma análoga ao Jacobiano da Eq. 3.7 (Antonelli, 2006).

$$\nu_{e[6 \times 1]} = \begin{bmatrix} \nu_{e1} \\ \nu_{e2} \end{bmatrix} = \mathbf{J}_e \dot{\eta}_e \quad (3.13)$$

As velocidades e acelerações dos elos do manipulador são determinadas a partir de ζ e $\dot{\zeta}$. Assumindo que as grandezas estão expressas no referencial do elo, estas são definidas como

$$\omega_i = \omega_{i-1} + \dot{q}_i \mathbf{z}_{i-1} \quad (3.14)$$

$$\dot{\omega}_i = \dot{\omega}_{i-1} + \omega_{i-1} \times \dot{q}_i \mathbf{z}_{i-1} + \ddot{q}_i \mathbf{z}_{i-1} \quad (3.15)$$

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \omega_i \times \mathbf{r}_{i-1,i} \quad (3.16)$$

$$\mathbf{v}_{i,c} = \mathbf{v}_{i-1} + \omega_i \times \mathbf{r}_{i-1,i,c} \quad (3.17)$$

$$\mathbf{a}_i = \mathbf{a}_{i-1} + \dot{\omega}_i \times \mathbf{r}_{i-1,i} + \omega_i \times (\omega_i \times \mathbf{r}_{i-1,i}) \quad (3.18)$$

onde, em relação a origem do referencial do elo i , ω_i é a velocidade angular, $\dot{\omega}_i$ é a aceleração angular, \mathbf{v}_i é a velocidade linear e \mathbf{a}_i é a aceleração linear. $\mathbf{v}_{i,c}$ é a velocidade do centro de massa do elo, e \mathbf{z}_{i-1} é o vetor de direção de \dot{q} (Siciliano et al., 2009; Antonelli, 2006).

Em relação ao efetuador final, um UVMS pode ser considerado cinematicamente redundante, uma vez que o veículo completamente atuado possui seis graus de liberdade, além dos graus de liberdade do manipulador. A resolução da cinemática inversa, necessária para a determinação dos movimentos do veículo e das juntas necessários para se obter a postura desejada para o efetuador final, apresenta os mesmos problemas de manipuladores cinematicamente redundantes, além de algumas características próprias de sistemas veículo-manipulador.

Em comum com manipuladores redundantes de base fixa, há a existência de infinitas soluções para a cinemática inversa, bem como

a necessidade de evitamento de singularidades e limites de juntas. A resolução de redundância é extensamente discutida na literatura, e uma revisão é feita em (Chiaverini; Oriolo; Walker, 2008).

A resolução da cinemática inversa em UVMS tem de considerar o fato de que alguns graus de liberdade do sistema correspondem ao movimento do veículo, que tem controle mais complexo e cujo movimento requer mais energia do que o movimento das juntas. Aplicados sem a consideração anterior, os métodos de resolução acabam por causar movimentos desnecessários no veículo.

Haverão muitas situações, porém, em que o movimento coordenado do veículo com o manipulador será desejável, como no evitamento de limites de junta do manipulador ou de singularidades, por exemplo. Em outras, este movimento será necessário, como em tarefas onde apenas o movimento do manipulador não permite ao efetuador final alcançar as posições especificadas (na limpeza de cascos de navios, por exemplo). O evitamento de obstáculos é outra situação em que a coordenação dos movimentos entre veículo e manipulador é necessária.

O estudo da cinemática inversa de UVMS considerando os fatores supracitados é um tema de pesquisa com poucos trabalhos ainda realizados. Antonelli (2006) analisa algumas das técnicas baseadas em pseudoinversa do Jacobiano adaptadas para considerar esses fatores. Resultados experimentais de implementação de uma técnica de resolução de cinemática inversa são analisados em (Soylu; Buckham; Podhoredski, 2010).

3.3 CINEMÁTICA DE UVMS ATRAVÉS DE HELICOIDES

A aplicação da teoria dos helicoides e de ferramentas nela baseadas para a análise de sistemas veículo-manipulador foi originalmente tratada em (Santos, 2006; Santos et al., 2006). A partir dessa análise foi obtido um modelo integrado entre veículo, manipulador e meio. As técnicas para resolução de tarefas deste método permitem resolver problemas inerentes a sistemas cinematicamente redundantes como os UVMS com vantagens em relação aos métodos usualmente encontrados na literatura.

A *teoria dos helicoides* é uma ferramenta para o estudo da cinemática e da estática de corpos rígidos livres no espaço ou vinculados em cadeias cinemáticas. O *helicóide* é um ente geométrico que representa simultaneamente grandezas translacionais e rotacionais segundo um *eixo* de referência, e que estão relacionadas por um *passo*. O heli-

coide de movimento, ou *heligiro*, representa velocidade linear e angular em uma única entidade. Um helicoides de ação, ou *heliforça*, representa força e momento (Hunt, 2000; Campos, 2004; Dai, 2006).

O conhecimento de fundamentos da teoria dos helicoides e das ferramentas nela baseadas é necessário para o desenvolvimento dos modelos cinemáticos desse texto. Visões gerais da cinemática por helicoides estão disponíveis na literatura (Tsai, 1999; Davison; Hunt, 2004). Apesar de não ser o objeto deste texto, o Apêndice A introduz resumidamente o assunto para facilitar a análise dos modelos baseados em helicoides da tese.

As cadeias cinemáticas de manipuladores podem ser modeladas através da teoria dos helicoides. Vários trabalhos na literatura discutem as características e usos de tal modelagem para tratamento de redundância, evitamento de obstáculos e análise cinemática de robôs paralelos e cooperativos, como por exemplo (Campos; Guenther; Martins, 2005; Fontan, 2007; Guenther et al., 2008; Simas et al., 2009; Rocha et al., 2009; Tonetto; Rocha; Dias, 2010).

Como já foi visto, veículos subaquáticos são tratados como corpos rígidos livres no espaço pela literatura, e seu movimento é composto por grandezas vetoriais lineares e angulares. Assim, este movimento pode ser representado por heligiros. Um UVMS, por sua vez, combina os movimentos do veículo e do manipulador, que podem ser, então, modelados por métodos baseados em helicoides (Santos, 2006).

Assim, o estado de velocidades de um corpo rígido pode ser descrito por um heligiro, definido para um ponto instantaneamente coincidente com a origem do referencial escolhido e que se move junto com o corpo. Uma representação do heligiro de um veículo subaquático e seus componentes é mostrada na Figura 17.

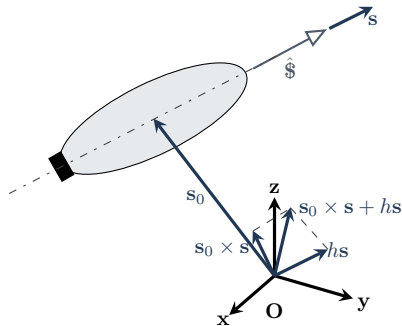


Figura 17. Definição do heligiro para o veículo subaquático

A velocidade do veículo é descrita pelo heligiros $\mathbf{\$}_v = \hat{\mathbf{\$}}_v \dot{q}_v$, onde $\hat{\mathbf{\$}}_v$ é o helicoide normalizado que descreve o movimento instantâneo infinitesimal e \dot{q}_v é a magnitude do movimento³, que pode ser igual à velocidade angular $\|\boldsymbol{\omega}_v\|$ no caso geral, ou à velocidade linear $\|\mathbf{v}_{v,o}\|$ se o movimento for puramente translacional. Utilizando a definição do helicoide normalizado, o heligiros $\mathbf{\$}_v$ é descrito em coordenadas de Plücker como

$$\mathbf{\$}_{v[6 \times 1]} = \begin{bmatrix} \boldsymbol{\omega}_v \\ \mathbf{v}_{v,o} \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ \mathbf{s}_0 \times \mathbf{s} + h\mathbf{s} \end{bmatrix} \dot{q}_v \quad (3.19)$$

onde \mathbf{s} é um vetor unitário que representa a direção do heligiros, \mathbf{s}_0 é o vetor posição do eixo do helicoide e h é o seu passo.

A definição de um heligiros depende do referencial sobre o qual os seus componentes são definidos. Caso seja necessário expressá-lo segundo um referencial diferente do usado para a sua definição, pode-se aplicar uma transformação de helicoides (Tsai, 1999) definida como

$${}^I \mathbf{T}_{v[6 \times 6]} = \begin{bmatrix} {}^I \mathbf{R}_v & 0 \\ \mathcal{S}({}^I \mathbf{r}_v) {}^I \mathbf{R}_v & {}^I \mathbf{R}_v \end{bmatrix} \quad (3.20)$$

onde I e v são os referenciais inercial e do corpo do veículo, respectivamente, adotados como exemplo. ${}^I \mathbf{R}_v$ é a matriz de rotação entre os dois sistemas de coordenadas e $\mathcal{S}({}^I \mathbf{r}_v)$ é a matriz antissimétrica gerada pelo operador $\mathcal{S}(\cdot)$ a partir do vetor posição do referencial do veículo segundo o referencial inercial.

O movimento do veículo no espaço em relação a um referencial pode ser considerado como imposto por uma junta de seis graus de liberdade, que pode ser expandida em seis juntas de um grau de liberdade (Santos, 2006; Featherstone, 2008; Waldron; Schmiedeler, 2008). Sua velocidade, então, é composta pela soma das velocidades de cada junta, representadas por heligiros segundo um mesmo referencial, ou

$$\mathbf{\$}_{v[6 \times 1]} = \sum_{i=1}^6 \mathbf{\$}_{v,i} = \sum_{i=1}^6 \hat{\mathbf{\$}}_{v,i} \dot{q}_i = \mathbf{J}_{[6 \times n]} \dot{\mathbf{q}}_{[n \times 1]} \quad (3.21)$$

onde a matriz Jacobiana \mathbf{J} é formada pelos helicoides normalizados das juntas e $\dot{\mathbf{q}}$ é um vetor coluna contendo as magnitudes dos heligiros.

³O símbolo \dot{q} para magnitude de velocidade é adotado para manter compatibilidade com a notação usada para variáveis de juntas de uma cadeia cinemática na robótica.

É possível definir o Jacobiano como $\mathbf{J} = \mathbf{I}_6$ através da escolha de um referencial no corpo do veículo paralelo ao referencial inercial. Para expressar \mathbf{J} no referencial do veículo (orientado de acordo com os eixos do seu corpo), é necessária uma transformação de helicoides. Esta é simplificada pelo fato das origens dos sistemas de coordenadas dos dois referenciais serem coincidentes. Assim, o estado de movimento do veículo segundo o seu referencial é expresso como

$${}^v\mathcal{S}_v = {}^v\mathbf{T}_{v,c} {}^{v,c}\mathbf{J}\dot{\mathbf{q}} = \begin{bmatrix} {}^v\mathbf{R}_{v,c} & 0 \\ 0 & {}^v\mathbf{R}_{v,c} \end{bmatrix} {}^{v,c}\mathbf{J}\dot{\mathbf{q}} \quad (3.22)$$

onde os referenciais v e v,c são relativos ao veículo e ao referencial segundo o qual os helicoides normalizados foram definidos, respectivamente. A matriz de rotação ${}^v\mathbf{R}_{v,c}$, utilizando ângulos de Euler, é a mesma definida na Equação 3.6.

O modelo cinemático de um manipulador pode ser obtido da mesma forma acima exposta para o movimento do veículo. Para tanto, utiliza-se o método dos helicoides sucessivos, descrito sucintamente no Apêndice A e detalhado em (Tsai, 1999). A velocidade do efetuador final em relação à base do manipulador pode ser descrita como

$${}^m\mathcal{S}_e = {}^m\mathbf{J}\dot{\mathbf{q}}_m \quad (3.23)$$

onde o índice m indica que a grandeza é expressa segundo o referencial da base do manipulador. A matriz coluna $\dot{\mathbf{q}}_m$ é formada pelas magnitudes das velocidades das juntas do manipulador.

No sistema veículo-manipulador, o movimento do efetuador final é função do movimento do veículo e do movimento dos elos do manipulador, sendo expresso no referencial do veículo como

$${}^v\mathcal{S}_e = {}^v\mathcal{S}_v + {}^v\mathbf{T}_m {}^m\mathcal{S}_e \quad (3.24)$$

$${}^v\mathbf{T}_{m,b} = \begin{bmatrix} \mathbf{I}_3 & 0 \\ \mathcal{S}({}^v\mathbf{r}_{v,m}) & \mathbf{I}_3 \end{bmatrix} \quad (3.25)$$

considerando o arranjo mostrado na Figura 18. Nele, o sistema de coordenadas da base do manipulador é paralelo ao sistema de coordenadas do veículo e sua posição é definida pelo vetor ${}^v\mathbf{r}_{v,m}$.

O movimento do efetuador final do UVMS pode ser descrito por uma cadeia cinemática virtual (Campos; Guenther; Martins, 2005). Esta faz o fechamento da cadeia cinemática do UVMS, como mostra a Figura 19, permitindo o uso do método de Davies para o estudo das

cinemáticas direta e inversa (Davies, 1981).

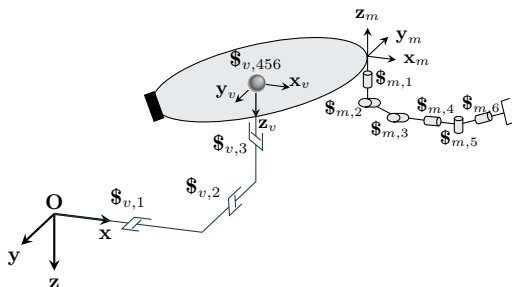


Figura 18. Cadeia cinemática do sistema de intervenção subaquática

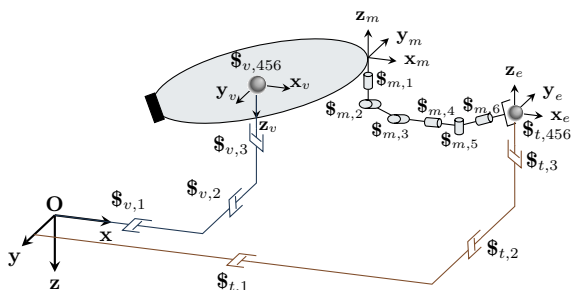


Figura 19. Cadeia do UVMS fechada pela cadeia virtual de posição

Para facilitar a análise de cadeias cinemáticas utiliza-se a teoria dos grafos (Diestel, 2005; Bang-Jensen; Gutin, 2007). Ela é usada tanto para sistematizar e simplificar a representação das cadeias quanto para analisar as suas características através das ferramentas existentes para o estudo das propriedades de grafos. Os fundamentos da aplicação de grafos na análise cinemática também são sucintamente apresentados no Apêndice A.

O *dígrafo de movimento* é uma representação por grafos de cadeias cinemáticas. Nele, cada vértice corresponde a um elo e cada aresta corresponde a uma junta de um grau de liberdade⁴ da cadeia. O termo dígrafo deve-se ao fato das arestas serem direcionadas. Além dos sentidos das arestas, cada circuito independente do grafo tem um

⁴Uma junta com n graus de liberdade é expandida em subcadeia com n juntas de um grau de liberdade.

sentido atribuído. Um exemplo de dígrafo de movimento é apresentado na Figura 20, que corresponde à cadeia fechada da Figura 19.

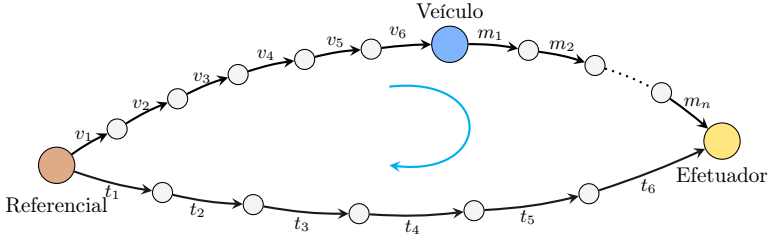


Figura 20. Dígrafo de movimento do UVMS

De acordo com o método de Davies, a soma das velocidades relativas entre os elos de uma cadeia cinemática fechada é igual a zero (Davies, 1981). Em uma cadeia cinemática com l circuitos independentes, F_b graus de liberdade *brutos* (somatório dos graus de liberdade de todas as juntas da cadeia cinemática) e um sistema de helicoides de dimensão λ , a equação de restrição que descreve essa relação assume a forma da Equação 3.26,

$$\mathbf{N}_{[\lambda \times F_b]} \dot{\mathbf{q}}_{[F_b \times 1]} = \mathbf{0}_{[\lambda \times 1]} \quad (3.26)$$

onde \mathbf{N} é a *matriz de rede*, que representa o movimento das juntas em cada circuito independente, através dos seus heligiros normalizados e $\dot{\mathbf{q}}$ é o vetor de velocidades das juntas da cadeia cinemática. A matriz de rede é calculada a partir da matriz dos heligiros normalizados \mathbf{D} e da matriz de circuitos \mathbf{B} , obtida a partir da análise do dígrafo de movimento da cadeia.

Para o grafo da Figura 20, a cadeia é formada por apenas um circuito ($l = 1$), opera no espaço $\lambda = 6$ e $F_b = 6 + 6 + n$, onde n é o número de juntas do manipulador serial. Ambas as cadeias virtuais do circuito possuem 6 graus de liberdade (postura do efetuador final e do veículo). Isso resulta na matriz de rede da Equação 3.27,

$$\mathbf{N}_{[6 \times F_b]} = \begin{bmatrix} \hat{\mathbf{S}}_v & \hat{\mathbf{S}}_m & -\hat{\mathbf{S}}_t \end{bmatrix} \quad (3.27)$$

obtida através de

$$\mathbf{D}_{[6 \times F_b]} = \begin{bmatrix} \hat{\mathbf{s}}_v & \hat{\mathbf{s}}_m & \hat{\mathbf{s}}_t \end{bmatrix} \quad (3.28)$$

$$\mathbf{B}_{[1 \times F_b]} = \begin{bmatrix} \mathbf{1}_{[1 \times 6]} & \mathbf{1}_{[1 \times n]} & -\mathbf{1}_{[1 \times 6]} \end{bmatrix} \quad (3.29)$$

$$\mathbf{N}_{[6 \times F_b]} = \mathbf{D}_{[6 \times F_b]} \text{diag} \{ \mathbf{B}_{[1 \times F_b]} \} \quad (3.30)$$

onde o operador $\text{diag} \{ \mathbf{B}_1 \}$ gera uma matriz diagonal com os elementos da linha 1 de \mathbf{B} .

O vetor de variáveis de juntas $\dot{\mathbf{q}}_{[F_b \times 1]}$ é igual a

$$\dot{\mathbf{q}}_{[F_b \times 1]} = \begin{bmatrix} \dot{\mathbf{q}}_v \\ \dot{\mathbf{q}}_m \\ \dot{\mathbf{q}}_t \end{bmatrix} \quad (3.31)$$

onde os subscritos v , m e t indicam grandezas relativas ao veículo, ao manipulador e à cadeia virtual de postura do efetuador final (tarefa), respectivamente.

A análise do movimento da cadeia cinemática é feita através do particionamento da Equação 3.27 entre movimentos cuja magnitude é conhecida (denominados primários) e movimentos cuja magnitude se deseja determinar (denominados secundários), como na Equação 3.32,

$$\mathbf{N}\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{N}_p & \mathbf{N}_s \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_p^T & \dot{\mathbf{q}}_s^T \end{bmatrix}^T = \mathbf{0} \quad (3.32)$$

onde os subscritos p e s denotam variáveis das partições primária e secundária, respectivamente. As magnitudes secundárias são obtidas ao se isolar $\dot{\mathbf{q}}_s$ como na Equação 3.33, que terá solução se a matriz \mathbf{N}_s for inversível.

$$\dot{\mathbf{q}}_s = -\mathbf{N}_s^{-1} \mathbf{N}_p \dot{\mathbf{q}}_p \quad (3.33)$$

Para a cinemática direta, definem-se como primárias as variáveis da cadeia real do UVMS e como secundárias as variáveis da cadeia virtual de posição do efetuador final, ou seja, $\mathbf{N}_p = \begin{bmatrix} \hat{\mathbf{s}}_v & \hat{\mathbf{s}}_m \end{bmatrix}$, $\dot{\mathbf{q}}_p = \begin{bmatrix} \dot{\mathbf{q}}_v^T & \dot{\mathbf{q}}_m^T \end{bmatrix}^T$, $\mathbf{N}_s = \begin{bmatrix} -\hat{\mathbf{s}}_t \end{bmatrix}$ e $\dot{\mathbf{q}}_s = \begin{bmatrix} \dot{\mathbf{q}}_t \end{bmatrix}$.

Para a cinemática inversa são definidas como primárias as variáveis da cadeia virtual de posição do efetuador final, e como secundárias as variáveis da cadeia real do UVMS. Observa-se, porém, que a matriz \mathbf{N}_s não é quadrada, não podendo ser invertida, como é característico de sistemas redundantes. Nesse caso, pode-se empregar uma solução na

forma da Equação 3.34, onde se utiliza o operador de pseudoinversão de Moore-Penrose,

$$\dot{\mathbf{q}}_s = -\mathbf{N}_s^\dagger \mathbf{N}_p \dot{\mathbf{q}}_p \quad (3.34)$$

onde $\mathbf{N}_s^\dagger = \mathbf{N}_s^T (\mathbf{N}_s \mathbf{N}_s^T)^{-1}$ é a pseudoinversa de \mathbf{N}_s , que minimiza a função custo $\frac{1}{2} \dot{\mathbf{q}}^T \dot{\mathbf{q}}$ energia do sistema (Siciliano et al., 2009).

O método das restrições cinemáticas (Santos et al., 2006), porém, estabelece que deve-se procurar deixar a matriz \mathbf{N}_s quadrada, através de redistribuição das variáveis entre as partições primárias e secundárias. Assim, variáveis consideradas secundárias passam a ser primárias, implicando que terão seu movimento completamente definido. Além de possibilitar a inversão de \mathbf{N}_s , o efeito corresponde a impor restrições à cadeia cinemática. Com isso, pode-se aproveitar a redundância do sistema para resolver objetivos complementares para a tarefa, como por exemplo a minimização do movimento do veículo ou definir uma orientação que minimize o consumo de energia (Santos, 2006).

Uma abordagem mais branda do método das restrições cinemáticas, especialmente para sistemas com elevada redundância, é a de usar a mobilidade adicional para atender aos requisitos adicionais do sistema sem que \mathbf{N}_s seja quadrada. Assim, utiliza-se a inversa se for possível ter \mathbf{N}_s quadrada ou a pseudoinversa caso contrário.

A cinemática inversa de posição é obtida pela integração de $\dot{\mathbf{q}}_s$ no tempo. Uma técnica para minimização do erro que aparece ao se empregar métodos numéricos é apresentada em (Simas et al., 2009), a qual também é resumidamente descrita no Apêndice A.

Neste capítulo fez-se uma revisão bibliográfica da cinemática de sistemas veículo-manipulador subaquáticos. Apresentou-se tanto a modelagem comumente encontrada na literatura quanto a abordagem baseada na teoria dos helicoides. A modelagem por helicoides parte de uma analogia com manipuladores industriais para representar o conjunto veículo-manipulador de modo uniforme, além de definir um método de resolução da cinemática inversa da cadeia cinemática normalmente redundante que apresenta vantagens em relação a outros métodos tradicionalmente usados na literatura. Além disso, esta abordagem tem extensibilidade sistemática e uniforme para diferentes configurações cinemáticas. Essas características justificam a adoção da abordagem por helicoides neste trabalho, como será visto nos próximos capítulos.

4 ANÁLISE CINEMÁTICA DE SISTEMAS DE INTERVENÇÃO SUBAQUÁTICA

Este capítulo trata da análise da cinemática de sistemas subaquáticos de intervenção por helicoides. Considerando os UVMS das classes de trabalho em operação e as demandas de novos tipos de missões, busca-se estender o modelo cinemático descrito no capítulo anterior. A análise de alguns casos leva ao desenvolvimento de um método para generalizar o modelamento cinemático do sistema veículo-manipulador subaquático. Faz-se uma proposta de componentização de cadeias cinemáticas complexas, a fim de agilizar o seu modelamento a partir de subcadeias cuja cinemática é predeterminada. Por fim, discute-se a necessidade de reconfigurabilidade dos modelos cinemáticos dos UVMS ao longo da tarefa e como esta pode ser implementada.

4.1 EXTENSÃO DO MODELO CINEMÁTICO BASEADO EM HELICOIDES

O modelo cinemático por helicoides proposto por Santos (2006) foi desenvolvido para um UVMS composto por um veículo com um manipulador acoplado. A mesma consideração é feita por outros autores em trabalhos relativos ao modelo cinemático, como em (Antonelli, 2006; Marani; Choi; Yuh, 2009; Soyly; Buckham; Podhorodeski, 2010), por exemplo. Apesar de gerar resultados efetivos e inclusive levar ao projeto e construção de sistemas autônomos, esse modelo não contempla diretamente a estrutura cinemática de muitos UVMS das classes de trabalho usados para intervenção industrial, que costumam ter pelo menos dois manipuladores acoplados ao veículo.

O modelamento cinemático dessas classes de UVMS seria de interesse para aproveitar as estruturas cinemáticas de ROV atualmente em operação no desenvolvimento de sistemas autônomos. Apesar de ser possível obter o modelo cinemático para esses sistemas simplesmente duplicando os resultados para o sistema 1 veículo - 1 manipulador, o seu uso em tarefas onde os dois manipuladores atuam em cooperação requer desenvolvimentos adicionais para o planejamento de movimento.

Além disso, verifica-se que novos tipos de missões necessitam a cooperação entre dois ou mais UVMS para movimentação de peças de grande volume e difícil movimentação¹. Nesse caso, o modelo cine-

¹Um exemplo deste tipo de missão é o do posicionamento do sistema de contenção

mático deve ser estendido não apenas para considerar vários sistemas veículo-manipulador, mas também a relação entre estes e os objetos sendo manipulados. Este tema apresenta alguns resultados na literatura (Padir; Koivo, 2003; Padir, 2005).

Com base nas estratégias para a análise de sistemas robóticos cooperativos industriais (SRC) baseadas em helicoides (Dourado, 2005; Ribeiro; Guenther; Martins, 2008; Ribeiro; Martins, 2010; Tonetto, 2011), é possível estender o modelo do UVMS previamente desenvolvido de forma a considerar vários manipuladores, a cooperação entre estes e a cooperação entre UVMS. Os estudos sobre o uso de helicoides para evitamento de colisão de manipuladores (Fontan, 2007; Simas, 2008; Rocha et al., 2009) também podem ser empregados para ampliar a análise cinemática de UVMS de forma a considerar esse problema.

Para as situações acima relacionadas, são identificadas algumas configurações básicas de extensão do modelo cinemático proposto por Santos (2006). Para auxiliar no desenvolvimento das novas configurações, o grafo de movimento do caso original é novamente apresentado na Figura 21, desta vez em forma contraída, onde cada aresta corresponde à uma subcadeia cinemática (Tonetto, 2011). Os índices v , m e t correspondem ao veículo, ao manipulador e à tarefa, respectivamente.

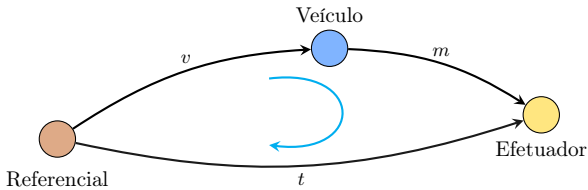


Figura 21. Grafo contraído de movimento do sistema subaquático de intervenção formado por um veículo e um manipulador

4.1.1 UVMS Com Manipuladores Operando de Forma Independente Entre Si

Essa é uma situação comum em missões de operação de painéis de plataformas de petróleo, em que um dos manipuladores é utilizado para fixar o UVMS em uma estrutura enquanto o outro realiza a operação de manipulação. Outro caso que caracterizaria esse tipo de modelo seria a

do vazamento da plataforma *Deepwater Horizon* (Leff; Plushnick-Masti, 2010).

execução simultânea de duas ou mais subtarefas, como posicionar um objeto com um manipulador enquanto outro utiliza uma ferramenta nesse objeto ou em outro do espaço da tarefa.

O grafo de movimento apresentado na Figura 22 ilustra a cadeia cinemática de um sistema de intervenção com dois manipuladores. As cadeias virtuais t_1 e t_2 são usadas para impor o movimento aos efetuadores finais dos manipuladores. O UVMS é representado pelas cadeias reais v (do veículo), m_1 e m_2 (dos manipuladores).

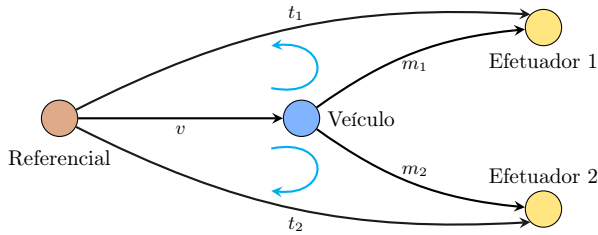


Figura 22. Grafo contraído de movimento do UVMS com dois manipuladores independentes

Existem dois circuitos independentes no grafo de movimento. Adotando a convenção de sentidos mostrada na Figura 22, a matriz de rede é definida como mostrado na Equação 4.1.

$$\mathbf{N} = \begin{bmatrix} \hat{\mathbf{s}}_v & \hat{\mathbf{s}}_{m_1} & -\hat{\mathbf{s}}_{t_1} & \mathbf{0} & \mathbf{0} \\ \hat{\mathbf{s}}_v & \mathbf{0} & \mathbf{0} & \hat{\mathbf{s}}_{m_2} & -\hat{\mathbf{s}}_{t_2} \end{bmatrix} \quad (4.1)$$

O grafo de movimento pode representar sistemas com um número qualquer de manipuladores. Para isso, cada manipulador tem uma cadeia representando a sua estrutura cinemática e uma cadeia virtual de imposição de movimento do respectivo efetuador final no grafo de movimento. Assim, para cada manipulador há um circuito independente correspondente. Na matriz de rede, isso implica em uma linha para cada manipulador onde, para um manipulador i , os heligios normalizados das cadeias v e m_i se mantém com o sinal original, os da cadeia t_i trocam de sinal e os demais heligios são iguais a $\mathbf{0}$.

A especificação da tarefa é feita pelas variáveis das cadeias t_i . Durante a intervenção, o veículo usualmente permanece estacionário ou tem algum movimento que minimiza o consumo de energia (Antonelli, 2006), o que inclui as variáveis da cadeia v na tarefa. O particionamento da matriz de rede é mostrado nas Equações 4.2 e 4.3.

$$\mathbf{N}_p = \begin{bmatrix} \hat{\$}_v & -\hat{\$}_{t_1} & \mathbf{0} \\ \hat{\$}_v & \mathbf{0} & -\hat{\$}_{t_2} \end{bmatrix} \quad (4.2)$$

$$\mathbf{N}_s = \begin{bmatrix} \hat{\$}_{m_1} & \mathbf{0} \\ \mathbf{0} & \hat{\$}_{m_2} \end{bmatrix} \quad (4.3)$$

4.1.2 UVMS Com Manipuladores Operando em Cooperação

Os manipuladores de um UVMS podem trabalhar de forma cooperativa na execução de uma tarefa, da mesma forma que seus equivalentes no ambiente industrial. A principal diferença entre um tipo de sistema e outro é o fato da base dos manipuladores do primeiro caso ser móvel (o veículo). A Figura 23 corresponde ao grafo de movimento de uma configuração de tarefa cooperativa usando dois manipuladores de um mesmo sistema de intervenção.

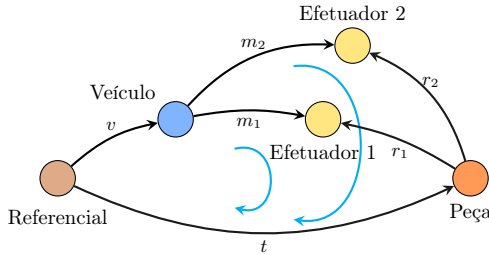


Figura 23. Grafo contraído de movimento do UVMS com dois manipuladores executando uma tarefa de forma cooperativa

A tarefa consiste na manipulação de uma peça cujo movimento é representado pela cadeia virtual t . Como no caso anterior, o UVMS é representado pelas cadeias v , m_1 e m_2 . As cadeias virtuais r_1 e r_2 representam o movimento relativo entre os efetuidores finais dos manipuladores e o ponto de referência da peça.

O grafo de movimento contém dois circuitos independentes. De acordo com a convenção de sentidos definida na Figura 23, a matriz de rede é definida pela Equação 4.4.

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_v & \hat{\$}_{m_1} & -\hat{\$}_{r_1} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} \\ \hat{\$}_v & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{m_2} & -\hat{\$}_{r_2} \end{bmatrix} \quad (4.4)$$

A extensão do grafo para um número maior de manipuladores consiste no acréscimo de um arco para cada manipulador adicional, formado pelas cadeias do manipulador m_i e do movimento relativo entre este e a peça r_i . A matriz de rede terá então uma linha adicional para cada novo circuito. Para a linha correspondente ao manipulador i , os heligiros normalizados das cadeias v e m_i se mantêm com o sinal original, t e r_i invertem seu sinal e os demais heligiros são iguais a $\mathbf{0}$.

A tarefa é especificada pela imposição de movimento da cadeia t . Para um veículo estacionário, a cadeia v tem uma restrição de movimento. O movimento relativo entre os efetuadores e a peça é também especificável (por exemplo, os efetuadores mantêm a mesma posição relativa ao ponto de referência da peça). Restam então as variáveis das cadeias dos manipuladores a serem determinadas, o que leva ao particionamento da matriz de rede descrito nas Equações 4.5 e 4.6.

$$\mathbf{N}_p = \begin{bmatrix} \hat{\$}_v & -\hat{\$}_{r_1} & -\hat{\$}_t & \mathbf{0} \\ \hat{\$}_v & \mathbf{0} & -\hat{\$}_t & -\hat{\$}_{r_2} \end{bmatrix} \quad (4.5)$$

$$\mathbf{N}_s = \begin{bmatrix} \hat{\$}_{m_1} & \mathbf{0} \\ \mathbf{0} & \hat{\$}_{m_2} \end{bmatrix} \quad (4.6)$$

4.1.3 Cooperação Entre UVMS

A cooperação entre UVMS pode ser considerada em situações em que a operação com um único sistema de intervenção é mais complexa ou mesmo impossível de ser realizada, como no compartilhamento de cargas, posicionamento de grandes estruturas e em montagem/manutenção subaquática. Uma configuração que representa a cooperação de dois UVMS com um manipulador cada sobre uma peça é descrita pelo grafo de movimento mostrado na Figura 24.

Esse modelo cinemático foi obtido ao se estender a sistematização feita por Tonetto (2011) para SRC industriais. Nele, cada arco do grafo de movimento corresponde às cadeias do veículo v_i , do manipulador m_i e do movimento relativo entre efetuador final e peça r_i de um UVMS i . Um arco corresponde à cadeia virtual t que define o movimento a ser imposto para a peça. A matriz de rede para esse grafo assume a forma da Equação 4.7.

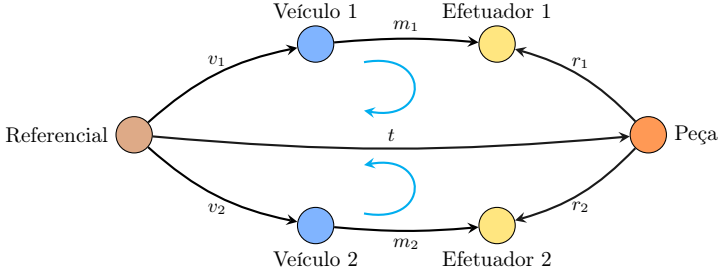


Figura 24. Grafo contraído de movimento do sistema de intervenção subaquática cooperativo

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_{v_1} & \hat{\$}_{m_1} & -\hat{\$}_{r_1} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{v_2} & \hat{\$}_{m_2} & -\hat{\$}_{r_2} \end{bmatrix} \quad (4.7)$$

O sistema cooperativo pode ser formado por um número qualquer de UVMS, com um circuito independente correspondente para cada um no grafo de movimento. Para um UVMS i , haverá um arco composto pelas cadeias do veículo v_i , do manipulador m_i e da posição relativa entre efetuador final e peça r_i . Cada linha da matriz de rede corresponderá a um UVMS, onde os heligiros normalizados da tarefa t e da posição relativa do efetuador do manipulador i à peça invertem seu sinal, enquanto os do veículo v_i e do manipulador m_i mantêm seu sinal e os demais heligiros assumem valor igual a $\mathbf{0}$.

A tarefa é especificada pela imposição de movimento da cadeia t . Se os veículos permanecem estacionários, as cadeias v_i tem seu movimento restrito. As cadeias r_i também podem ter seu movimento imposto de forma a especificar o movimento relativo desejado entre os efetadores e a peça. Assim, o particionamento da matriz de rede é feito de forma que as variáveis das cadeias dos manipuladores sejam determinadas, como é mostrado nas Equações 4.8 e 4.9.

$$\mathbf{N}_p = \begin{bmatrix} \hat{\$}_{v_1} & -\hat{\$}_{r_1} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{v_2} & -\hat{\$}_{r_2} \end{bmatrix} \quad (4.8)$$

$$\mathbf{N}_s = \begin{bmatrix} \hat{\$}_{m_1} & \mathbf{0} \\ \mathbf{0} & \hat{\$}_{m_2} \end{bmatrix} \quad (4.9)$$

A sistematização proposta também permite trabalhar com diferentes arranjos de UVMS atuando de forma cooperativa em uma intervenção. Tome-se como exemplo um cenário onde dois UVMS com dois manipuladores que atuam em cooperação, como o representado na Figura 23, atuam de forma cooperativa entre si. O grafo de movimento da Figura 24 tem acrescidos dois arcos representando os manipuladores adicionais, resultando no grafo da Figura 25.

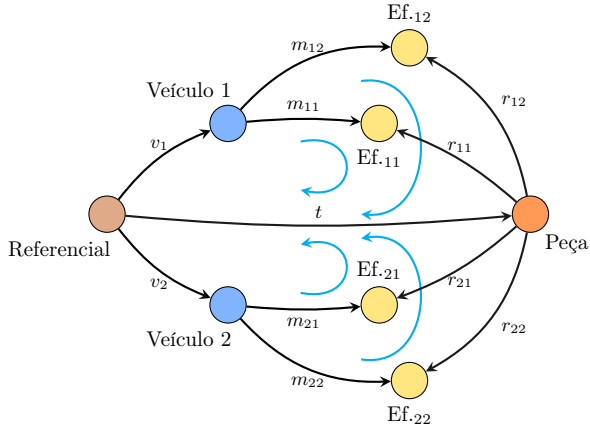


Figura 25. Grafo contraído de movimento do sistema de intervenção subaquática cooperativo

A matriz de rede para esse grafo assume a forma da Equação 4.10. Nela, os helicoides dos manipuladores são identificados pelo subscrito ij , onde i identifica o UVMS do qual o manipulador faz parte e j identifica o manipulador no UVMS. O mesmo subscrito é aplicado para as cadeias virtuais do movimento relativo entre efetuador e peça.

$$\mathbf{N} = \begin{bmatrix}
 \hat{\$}_{v_1} & \hat{\$}_{m_{11}} & -\hat{\$}_{r_{11}} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \dots \\
 \hat{\$}_{v_1} & \mathbf{0} & \mathbf{0} & \hat{\$}_{m_{12}} & -\hat{\$}_{r_{12}} & -\hat{\$}_t & \dots \\
 \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \dots \\
 \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \dots \\
 \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \\
 \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \\
 \dots & \hat{\$}_{v_2} & \hat{\$}_{m_{21}} & -\hat{\$}_{r_{21}} & \mathbf{0} & \mathbf{0} & \\
 \dots & \hat{\$}_{v_2} & \mathbf{0} & \mathbf{0} & \hat{\$}_{m_{22}} & -\hat{\$}_{r_{22}} &
 \end{bmatrix} \quad (4.10)$$

Um possível particionamento para calcular o movimento dos manipuladores na execução de uma tarefa teria as variáveis das cadeias m_{ij} como secundárias enquanto as demais seriam primárias. As partições da matriz de rede assumiriam a forma das Equações 4.11 e 4.12.

$$\mathbf{N}_p = \begin{bmatrix} \hat{\$}_{v_1} & -\hat{\$}_{r_{11}} & \mathbf{0} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hat{\$}_{v_1} & \mathbf{0} & -\hat{\$}_{r_{12}} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{v_2} & -\hat{\$}_{r_{21}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{v_2} & \mathbf{0} & -\hat{\$}_{r_{22}} \end{bmatrix} \quad (4.11)$$

$$\mathbf{N}_s = \begin{bmatrix} \hat{\$}_{m_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{\$}_{m_{12}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\$}_{m_{21}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\$}_{m_{22}} \end{bmatrix} \quad (4.12)$$

Esse exemplo, cuja configuração não era tratada na literatura, mostra como a análise por helicoides pode ser sistematicamente estendida para representar e resolver a cinemática de diferentes cenários de sistemas de intervenção subaquática.

4.1.4 UVMS em Situação de Evitamento de Colisão

Outro contexto de operação que pode ocorrer frequentemente em missões de intervenção subaquática é a necessidade de evitamento de colisão com objetos presentes no espaço de trabalho. Esse é o caso de missões em ambientes restritos como plataformas subaquáticas ou dentro de destroços de acidentes e em naufrágios, por exemplo.

A modelagem por helicoides pode ser usada para definir estratégias de evitamento de colisão do UVMS enquanto este procura executar a tarefa designada. O procedimento é análogo ao empregado na literatura para o evitamento de colisão de manipuladores em ambientes restritos ou incertos (Fontan, 2007; Simas, 2008; Guenther et al., 2008; Rocha; Dias, 2010a). Para tanto, acrescenta-se uma cadeia virtual que representa a posição relativa entre o obstáculo a ser evitado e o componente do UVMS que pode colidir com ele (o veículo ou algum elo do manipulador).

Um exemplo de cadeia cinemática para essa situação é apresentado na Figura 26. O grafo de movimento representa um sistema de um veículo com um manipulador onde um elo m_i está em situação de

possível colisão com um obstáculo. Acrescenta-se uma cadeia virtual o para representar a posição relativa entre este elo e o obstáculo. Esta é utilizada para impor uma restrição de movimento de forma a evitar a colisão. Dependendo da configuração da cadeia virtual o , nem todas as variáveis são utilizadas para impor a restrição, ficando as demais a serem determinadas.

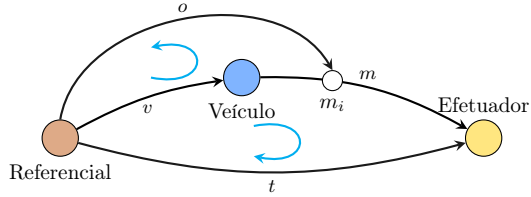


Figura 26. Grafo contraído de movimento de um UVMS operando em presença de obstáculos

Na situação da Figura 26, a matriz de rede é definida como na Equação 4.13. A cadeia do manipulador é aqui particionada em duas, uma do veículo até o elo que pode colidir ($m_{0,i}$) e uma deste até o efetuador final ($m_{i,n}$). A do veículo (v) e a da tarefa (t) se mantêm como no caso do sistema formado por um veículo e um manipulador.

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_v & \hat{\$}_{m_{0,i}} & \hat{\$}_{m_{i,n}} & -\hat{\$}_t & \mathbf{0} \\ \hat{\$}_v & \hat{\$}_{m_{0,i}} & \mathbf{0} & \mathbf{0} & -\hat{\$}_o \end{bmatrix} \quad (4.13)$$

A tarefa é especificada pelas variáveis da cadeia t . A restrição de movimento é dada pelas variáveis da cadeia o necessárias (identificadas como o_p), enquanto as demais estão livres para terem seus valores calculados. Se o veículo permanece estacionário, a cadeia v tem seu movimento restrito. Assim, resta determinar os valores das variáveis da cadeia m e das juntas da cadeia o que não tem seu movimento restrito (denominadas o_s). Assim, o particionamento da matriz de rede assume a forma descrita nas Equações 4.14 e 4.15.

$$\mathbf{N}_p = \begin{bmatrix} \hat{\$}_v & -\hat{\$}_t & \mathbf{0} \\ \hat{\$}_v & \mathbf{0} & -\hat{\$}_{o_p} \end{bmatrix} \quad (4.14)$$

$$\mathbf{N}_s = \begin{bmatrix} \hat{\$}_{m_{0,i}} & \hat{\$}_{m_{i,n}} & \mathbf{0} \\ \hat{\$}_{m_{0,i}} & \mathbf{0} & -\hat{\$}_{o_s} \end{bmatrix} \quad (4.15)$$

4.1.5 Observações Sobre Os Modelos Cinemáticos

Os casos apresentados nesta seção ilustram como é possível entender o modelo original de um UVMS com um único manipulador para cenários mais complexos de execução de tarefas. As estruturas cinemáticas divisadas nos casos anteriores podem ser usadas como base para a composição de cenários ainda mais complexos. A cooperação entre UVMS com mais de um manipulador, onde um deles deve evitar colisão com o meio, é um exemplo dessa composição.

Em relação às cadeias cinemáticas dos sistemas de intervenção, nota-se que os manipuladores não têm necessariamente o mesmo arranjo cinemático. Da mesma forma, as cadeias virtuais usadas para impor movimentos e restrições podem ser diferentes no veículo e na tarefa, sendo escolhidas aquela que, respeitando as regras básicas das cadeias virtuais de Assur, sejam mais adequadas para a descrição de determinado tipo de tarefa. Reitera-se também a necessidade dos heligiros das juntas da cadeia cinemática serem expressos segundo um mesmo referencial.

Os particionamentos das equações de restrição foram feitos a fim de ilustrar o processo de obtenção de uma solução para a resolução de tarefas. Dependendo do contexto de execução destas, diferentes particionamentos podem ser definidos. Por exemplo, o veículo pode não permanecer estacionário quando o manipulador encontra-se em configurações que o aproximam de alguma singularidade ou limite de junta. Nesse caso, algumas variáveis da cadeia do veículo podem compor a partição secundária da cadeia a fim de serem determinadas. É necessário que o sistema tenha a mobilidade necessária para executar a tarefa especificada, o que implica no número de variáveis secundárias ser igual ou maior que a dimensão do espaço da tarefa.

O caso da execução de tarefas por manipuladores independentes foi aqui desenvolvido para demonstrar as possibilidades de extensão da abordagem cinemática por helicoides. Neste trabalho, porém, será dado maior enfoque na interação entre manipuladores e entre UVMS na execução cooperativa de tarefas. Assim, o caso de manipuladores independentes não será posteriormente tratado neste texto. Observa-se, porém, que é uma situação que é usada em diversos casos onde o UVMS deve se fixar a uma estrutura subaquática que sofrerá a intervenção. Nesse caso, pode-se considerar o UVMS como um manipulador único. O efetuador final que se prende a estrutura equivaleria à uma base fixa, enquanto o outro efetuador final seria o responsável pela manipulação. O veículo, nesse caso, seria apenas um elo intermediária-

rio de um manipulador serial. O grafo de movimento correspondente é mostrado na Figura 27. A solução para esse tipo de configuração está disponível na literatura, considerando o sistema um manipulador redundante (Simas et al., 2009; Simas; Fontan; Martins, 2011).

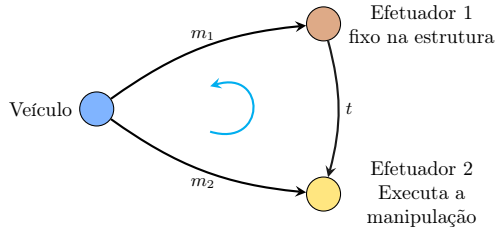


Figura 27. Grafo contraído de movimento de um UVMS fixo à uma estrutura sobre a qual executará uma tarefa

Nos modelos cinemáticos, a peça pode representar não apenas um simples objeto, mas sim o meio com o qual o UVMS irá interagir. A tarefa pode ser definida a partir de um ponto de referência que represente o movimento/restrrição do meio como um todo e do movimento relativo entre este ponto e os efetdutores finais do sistema de intervenção.

4.2 MODULARIZAÇÃO DA MODELAGEM CINEMÁTICA

Como pode-se observar nos casos discutidos na seção anterior, as cadeias cinemáticas dos sistemas de intervenção subaquáticas podem ser complexas de acordo com os arranjos cinemáticos dos UVMS e da natureza das tarefas a serem executadas. A abordagem por helicoides provê um método sistemático e uniforme de modelamento cinemático desses sistemas. Mesmo assim, a definição da equação de restrição continua sendo um processo demorado e de certa dificuldade.

Na análise dos grafos de movimento previamente definidos, pode-se ver a cadeia cinemática do sistema de intervenção como sendo composta por *subcadeias* cujas estruturas cinemáticas podem se repetir em um mesmo sistema ou aparecer em outras situações. O que se diferencia entre duas subcadeias com a mesma estrutura é o conjunto das variáveis de juntas e parâmetros que as definem e a sua posição relativa na cadeia composta por elas. Isso implica diferenças nas definições dos heligiros que as compõem devido às variáveis e parâmetros usados e à

necessidade de expressar eles segundo um mesmo referencial.

Mesmo assim, é possível definir uma abordagem de *composição* de cadeias cinemáticas por subcadeias cuja cinemática é previamente conhecida e parametrizável. As *cadeias compostas* teriam seus elos conectados por *componentes cinemáticos*, que seriam juntas simples² ou subcadeias. Uma *cadeia simples* seria formada apenas por juntas.

Cada cadeia deve ter seus heligiros expressos segundo um referencial comum a todos eles e ter um conjunto próprio de variáveis de juntas e parâmetros (como comprimentos de elos, por exemplo). Logo, se uma cadeia for usada para criar uma cadeia composta, uma transformação de helicoides deve ser associada a ela para que seus heligiros sejam expressos segundo o referencial da cadeia composta. Além disso, os nomes de variáveis e parâmetros devem ser modificados para evitar conflitos e confusões no modelo cinemático.

Para simplificar o desenvolvimento da modularização, assume-se que toda cadeia cinemática tem uma base e um efetuator final e que ela pode se conectar a outras cadeias apenas por esses dois elos. O efetuator final também pode ser o efetuator da cadeia composta. Outra convenção adotada neste texto é que os heligiros de uma subcadeia são definidos segundo um referencial coincidente com a sua base. As subcadeias simples são seriais. Cada subcadeia é definida segundo as suas próprias variáveis (q_i , com $i = 1 \dots n$ a partir da junta conectada à base) e parâmetros.

Com base nessas convenções, obtém-se a cinemática de posição da cadeia representada pela matriz de transformação homogênea ${}^b\mathbf{A}_e$. A cinemática diferencial também é obtida, com a matriz de helicoides normalizados $\mathbf{D} = \left[\hat{\mathbf{s}}_1 \dots \hat{\mathbf{s}}_n \right]$ e o vetor de variáveis $\hat{\mathbf{q}}$, que permitem descrever o estado de velocidades do efetuator final³.

Para definir uma cadeia composta, pode-se partir da análise do grafo contraído de movimento. No caso de uma aresta representar uma subcadeia, associa-se a esta subcadeia a sua matriz de helicoides normalizados \mathbf{D}_{c_i} correspondente e uma transformação de helicoides que faça com que os heligiros de \mathbf{D}_{c_i} sejam expressos segundo o referencial adotado para a cadeia composta. A matriz de helicoides normalizados da cadeia composta é então definida como na Equação 4.16,

$$\mathbf{D} = \left[\begin{array}{ccc} r\hat{\mathbf{s}}_{c_1} & \dots & r\hat{\mathbf{s}}_{c_m} \end{array} \right] \quad (4.16)$$

²Neste texto, juntas de um grau de liberdade apenas.

³O processo de modelagem é brevemente descrito no Apêndice A.

onde ${}^r\hat{\mathbf{S}}_{c_i} = {}^r\mathbf{T}_{c_i} {}^{c_i}\mathbf{D}_{c_i}$ é formado pelos helicoides normalizados da subcadeia c_i expressos no referencial da cadeia composta após sofrer a transformação de helicoides ${}^r\mathbf{T}_{c_i}$.

Em uma cadeia composta, a transformação ${}^r\mathbf{T}_{c_i}$ pode ser definida a partir das próprias subcadeias. A transformação entre a referência de uma subcadeia c_i e a referência da subcadeia à qual ela se conecta (c_{i-1}) é definida a partir da cinemática de posição desta (matriz ${}^{c_{i-1}}\mathbf{A}_{e_{i-1}}$, de transformação entre o referencial do efetuador da subcadeia para a sua base).

Para uma sequência de subcadeias, a matriz de transformação resultante é obtida por premultiplicação das matrizes entre uma subcadeia e sua antecessora. Assim, para uma subcadeia c_i pode-se definir a transformação de helicoides como na Equação 4.17,

$${}^r\mathbf{T}_{c_i} = {}^r\mathbf{T}_{c_1} {}^{c_1}\mathbf{T}_{c_2} \cdots {}^{c_{i-1}}\mathbf{T}_{c_i} \quad (4.17)$$

onde ${}^r\mathbf{T}_{c_1}$ é a transformação de helicoides entre a base da cadeia composta e o referencial adotado para ela e ${}^{c_{i-1}}\mathbf{T}_{c_i}$ é a transformação de helicoides entre os referenciais i e $i-1$. Se o referencial de i coincide com o do efetuador final de $i-1$, ${}^{c_{i-1}}\mathbf{T}_{c_i}$ é determinada a partir da cinemática direta de c_{i-1} . Do contrário, acrescenta-se uma transformação entre esses dois referenciais, sendo a transformação definida como

$${}^{c_{i-1}}\mathbf{T}_{c_i} = {}^{c_{i-1}}\mathbf{T}_{e_{i-1}} {}^{e_{i-1}}\mathbf{T}_{c_i} \quad (4.18)$$

onde ${}^{e_{i-1}}\mathbf{T}_{c_i}$ é a transformação entre os referenciais do efetuador da cadeia $i-1$ e da base da cadeia i (usualmente constante) e ${}^{c_i}\mathbf{T}_{e_i}$ é a transformação entre os sistemas do referenciais da base da cadeia i e de seu efetuador final.

Por fim, é necessário renomear as variáveis de juntas e parâmetros das subcadeias de forma a evitar conflitos nas expressões. Em implementações computacionais, as variáveis e parâmetros podem ser armazenadas em objetos que são referenciados com nomes diferentes em dicionários específicos para cada cadeia.

Esse método de definição de cadeias pode ser sintetizado nos seguintes passos:

1. Desenhar o grafo de movimento do sistema robótico;
2. Identificar as possíveis subcadeias do sistema e suas relações;
3. Localizar as definições prévias dos tipos de subcadeias identificadas. Se necessário, criar os modelos para as subcadeias ainda não definidas obedecendo às convenções estabelecidas;

4. Renomear as variáveis de juntas e parâmetros das subcadeias de forma a evitar identificadores duplicados;
5. Definir as transformações de helicoides para cada subcadeia;
6. Resolver a cinemática da cadeia composta.

O método visa simplificar o processo de modelagem de cadeias complexas, não apenas no caso de sistemas subaquáticos. Ele se aplica a qualquer situação onde possam se divisar subcadeias dentro de uma cadeia complexa, como por exemplo em sistemas robóticos industriais cooperativos. Além disso, é possível modificar com poucos passos uma cadeia já modelada pela adição, remoção e substituição de subcadeias. Isso é possível fisicamente em alguns modelos de UVMS que podem ser reconfigurados antes de serem lançados para missões de acordo com o tipo de tarefas a serem executadas.

O método de composição pode ser usado para definir o modelo cinemático de toda a cadeia cinemática do sistema de intervenção de forma analítica. Com o uso de software CAS (*Computer Algebra System*) é possível obter de forma sistemática os heligiros de toda a cadeia através da composição dos modelos previamente definidos e das matrizes de transformação de helicoides.

Por outro lado, é possível pensar em uma automação da montagem de cadeias através do conceito de *composição* que aparece na literatura de padrões de projeto (Metsker, 2004) e da criação de um *banco de cadeias* contendo modelos predeterminados de cadeias comumente usadas. Esses conceitos são empregados na concepção de um *framework* computacional para implementação de modelos cinemáticos (Rocha; Tonetto; Dias, 2011a) que é discutido no Capítulo 5 e no Apêndice B.

As cadeias cinemáticas simples usadas nos modelos deste trabalho são descritas no Apêndice C, bem como as cadeias compostas relativas aos sistemas de intervenção analisados e simulados.

4.3 VARIABILIDADE DO MODELO CINEMÁTICO DE SISTEMAS SUBAQUÁTICOS DE INTERVENÇÃO

Ao longo da fase de intervenção de uma missão, o contexto de execução de tarefas pode variar. Os motivos para isso compreendem os diferentes objetivos principais/complementares de cada tarefa de intervenção a ser executada e as incertezas do ambiente. Neste último caso, podem ser citados como exemplos a movimentação independente

do objeto a sofrer a intervenção e a possibilidade de colisão com obstáculos que se desloquem dentro do espaço de trabalho (ou que não foram detectados anteriormente devido às dificuldades de modelagem do ambiente).

Embora o objetivo principal deva se manter (execução da intervenção pelos efetuadores finais), observa-se que os objetivos complementares podem mudar em função das variações de contexto. Isso implica em modificações no modelo cinemático do sistema e consequente resolução de sua cinemática inversa para o planejamento de movimento.

No caso do objeto se mover durante a execução da tarefa pode ser necessário que o veículo passe da condição de estacionário para móvel, de forma a evitar singularidades ou limites de juntas de um manipulador. Assim, o seu movimento deve ser determinado juntamente com os das juntas do manipulador.

Em outros casos, restrições ao movimento de veículos e juntas devem ser impostos de forma a evitar uma colisão do UVMS com um obstáculo ou para representar o travamento de alguma junta do manipulador. Em sistemas com UVMS em cooperação, o movimento de um ou mais veículos pode ser necessário para evitar limites de juntas de um ou mais manipuladores do sistema.

A literatura apresenta diferentes abordagens para aproveitar a mobilidade adicional de um sistema veículo-manipulador subaquático de forma a satisfazer diferentes objetivos complementares enquanto realiza uma tarefa. Notadamente, boa parte utiliza o conceito de projeção no espaço nulo para atender a um ou mais objetivos complementares (Antonelli, 2006). Uma abordagem em particular que considera a variação de objetivos complementares ao longo do tempo faz uso de um operador pseudoinverso ponderado, onde os pesos da matriz de ponderação são determinados por uma função objetivo das variáveis de juntas do manipulador (ou da proximidade destas de seus limites), de forma que a cinemática inversa resulte em movimentos para o veículo quando alguma junta se aproxima de seus limites (Sarkar; Podder, 2001).

Santos (2006), por sua vez, propôs um *modelo híbrido baseado em estados*. Nele, diferentes particionamentos da equação de restrição eram utilizados para resolver a cinemática diferencial do sistema para cada estado de operação. Técnicas de inteligência artificial eram usadas para identificar o estado do sistema e promover o chaveamento entre estados.

A ideia de modelo híbrido de estados agrupa em uma estratégia de planejamento de movimento diferentes soluções para a cinemática inversa do sistema relativas aos diferentes contextos de execução de

tarefas. Apesar de poder apresentar descontinuidades no movimento causadas pelas mudanças de estado (Guenther et al., 2008; Simas et al., 2009; Fontan, 2007), esse modelo reflete o requisito de *variabilidade* que o sistema de intervenção subaquática deve ter para atuar em diferentes contextos que surgem durante a execução das tarefas. Ao mesmo tempo, é possível sistematizar a definição das diferentes soluções de cinemática inversa e como elas podem mudar de um estado para outro.

A variabilidade da solução da cinemática inversa pode ser implementada de duas formas, de acordo com o tipo de mudança de contexto de execução da tarefa: através de *reparticionamento* da equação de restrição existente ou de *reconfiguração* da cadeia cinemática que representa o sistema de intervenção executando tarefas.

O reparticionamento consiste na redefinição das partições primária e secundária da cadeia cinemática do sistema, de forma que novos movimentos/restrições sejam impostos enquanto outros passam a ser calculados pelo método de Davies. Assim, alguns heligiros trocam de partição, enquanto outros permanecem na partição originalmente designada. A cadeia cinemática não sofre mudanças estruturais, e portanto a sua equação de restrição permanece a mesma.

Um caso de emprego de reparticionamento ocorre quando o veículo passa de estacionário a móvel para possibilitar que o efetuidor final alcance posturas que seriam impossíveis com o veículo mantendo posição e atitude fixas. O movimento do veículo precisa ser obtido pela solução da cinemática inversa junto com as referências das juntas.

Para um sistema como o da Figura 19, a matriz de rede descrita na Equação 3.27 é particionada como na Equação 4.19 quando o veículo deve se manter estacionário, sendo este o seu estado inicial. Ao ser gerado um evento que exige o movimento do veículo, o seu estado muda para móvel e o particionamento da matriz de rede passa a ser o da Equação 4.20, de forma que o movimento do veículo também possa ser determinado. O sistema volta ao estado inicial quando o manipulador puder operar em condições normais (afastado de singularidades e limites de juntas). Nas Equações 4.19 e 4.20, as variáveis primárias são escritas em vermelho, enquanto as secundárias são escritas em azul⁴.

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_v & \hat{\$}_m & -\hat{\$}_t \end{bmatrix} \quad (4.19)$$

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_v & \hat{\$}_m & -\hat{\$}_t \end{bmatrix} \quad (4.20)$$

⁴Essa convenção de cores será adotada deste ponto do texto em diante.

A Figura 28 ilustra essa situação em um diagrama de estados.

Outra situação de reparticionamento é causada pelo travamento de uma junta do manipulador, o que implicaria na variável a ela correspondente passar a ser primária (imposição de restrição) e que eventualmente as variáveis do veículo passem a ser secundárias se o manipulador não for redundante.

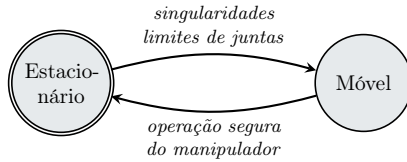


Figura 28. Diagrama de estados de operação de um UVMS - reparticionamento

A reconfiguração da cadeia cinemática é usada quando há necessidade de representar novas relações de movimentos/restrições do sistema ou modificações das relações atualmente existentes. Um exemplo é o evitamento de colisão. Inicialmente, se o sistema opera livre de obstáculos, pode-se assumir uma representação de cadeia cinemática que considere o UVMS e sua relação com o objeto a ser manipulado. Quando um obstáculo é detectado, deve-se impor uma restrição ao movimento do UVMS de forma que este não colida. Essa restrição implica na adição de uma cadeia cinemática virtual entre o elo que pode colidir e o obstáculo. A cadeia cinemática modificada resulta em uma nova equação de restrição, a qual deve ser particionada de acordo com os movimentos/restrições impostos tanto para execução da tarefa quando para evitar a colisão. Um diagrama de estados correspondente é apresentado na Figura 29.

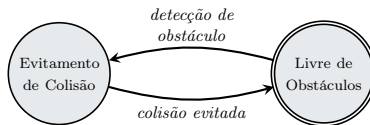


Figura 29. Diagrama de estados de operação de um UVMS - reconfiguração

Considere-se, então, o mesmo sistema de intervenção da Figura 19. Em um estado inicial livre de obstáculos e seguro para o manipulador, a matriz de rede correspondente ao dígrafo de movimento da

Figura 20 é descrita na Equação 3.27.

Ao ser detectado um obstáculo, o sistema deve passar ao estado de evitamento de colisão, onde o movimento relativo entre o obstáculo e o elo do sistema que tem possibilidade de colidir é representado por uma cadeia virtual resultando no dígrafo de movimento da Figura 26 e na matriz de rede da Equação 4.13. O evitamento de colisão impõe uma restrição de movimento do elo que se aproxima mais do obstáculo, de forma que se acrescentem as variáveis da cadeia virtual de evitamento de obstáculo na partição primária, como é mostrado nas Equações 4.14 e 4.15. Uma vez que o sistema esteja em uma situação que possa ser considerada livre do obstáculo, retorna-se ao estado primário e a cadeia cinemática que representa o sistema volta a ser a da Figura 20.

No estado de evitamento de colisão, o movimento do veículo passa a ser necessário se ele for colidir ou se apenas o movimento do manipulador não for capaz de evitar a colisão com um de seus elos. O particionamento da matriz de rede desse estado então assume a forma da Equação 4.21. Novamente, os subscritos p e s das variáveis da cadeia de obstáculos o correspondem respectivamente às variáveis que impõem a restrição e às que não impõem a restrição e podem ter seu movimento calculado pelo método de Davies.

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_v & \hat{\$}_{m_{0,i}} & \hat{\$}_{m_{i,n}} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} \\ \hat{\$}_v & \hat{\$}_{m_{0,i}} & \mathbf{0} & \mathbf{0} & -\hat{\$}_{o_p} & -\hat{\$}_{o_s} \end{bmatrix} \quad (4.21)$$

Na execução de tarefas em cooperação também é possível haver reconfiguração da cadeia cinemática do sistema. Se durante a execução de uma tarefa um UVMS se junta aos já atuantes, um novo arco é acrescentado ao grafo de movimento para representar o movimento do UVMS, como foi discutido anteriormente. A equação de restrição então é modificada e reparticionada de forma a refletir a nova situação e calcular o movimento de todos os componentes do sistema.

A Figura 30 serve para ilustrar esse caso. Em sistema com dois UVMS executando uma tarefa (Fig. 30a) adiciona-se um terceiro UVMS (Fig. 30b). À matriz de rede do sistema cooperativo inicial (Eq. 4.22) é acrescentada uma linha correspondente ao circuito independente do UVMS adicional, resultando na forma da Equação 4.23.

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_{v_1} & \hat{\$}_{m_1} & -\hat{\$}_{r_1} & -\hat{\$}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\$}_t & \hat{\$}_{v_2} & \hat{\$}_{m_2} & -\hat{\$}_{r_2} \end{bmatrix} \quad (4.22)$$

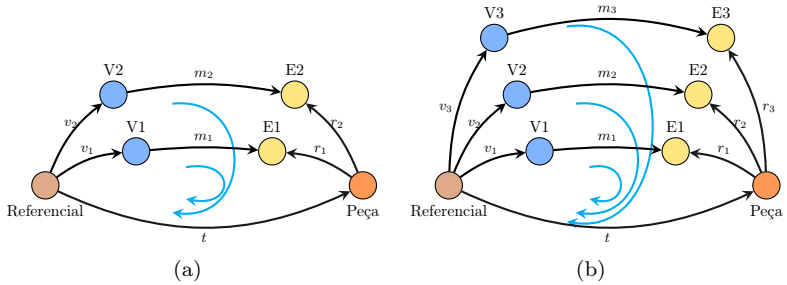


Figura 30. Reconfiguração de cadeia cinemática pela adição de um UVMS: (a)antes da adição; (b)após a adição

$$\mathbf{N} = \begin{bmatrix} \hat{\$}_{v_1} & \hat{\$}_{m_1} & -\hat{\$}_{r_1} & -\hat{\$}_t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\hat{\$}_t & \hat{\$}_{v_2} & \hat{\$}_{m_2} & -\hat{\$}_{r_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\hat{\$}_t & 0 & 0 & 0 & \hat{\$}_{v_3} & \hat{\$}_{m_3} & -\hat{\$}_{r_3} & 0 \end{bmatrix} \quad (4.23)$$

Deve-se observar que sempre que houver um reparticionamento há a necessidade de determinar referências a serem adotadas como valores das variáveis primárias do sistema, mesmo que seja para restringir totalmente o movimento ($\hat{q}_i = 0$).

Embora seja importante para o processo de resolução de tarefas, não serão tratados neste trabalho aspectos da tomada de decisão para a identificação de estados e chaveamento do modelo cinemático, uma vez que o objetivo da tese é a análise cinemática dos diferentes cenários de execução de tarefas. Esse assunto é analisado para o tipo de problema dos UVMS em (Santos, 2006; Santos et al., 2006; Simas et al., 2009; Simas; Fontan; Martins, 2011).

A extensão do modelo cinemático por helicoides considerando diferentes tipos de atividades e contextos de execução de tarefas por UVMS foi o principal assunto deste capítulo. Um novo modelo cinemático, derivado do estudo de sistemas robóticos cooperativos, foi desenvolvido para representar a cooperação entre UVMS e entre manipuladores de um mesmo sistema de intervenção. Ao serem analisados diferentes arranjos cinemáticos, observou-se a possibilidade de componentização dessas cadeias complexas ao considerá-las formadas por subcadeias mais simples. Dessa forma, pode-se agilizar o processo de modelagem cinemática e inclusive automatizá-lo. Uma sistemati-

zação desse processo de modelagem por componentização foi descrita. Verificou-se também que o sistema de intervenção precisa ter características de variabilidade para poder executar as tarefas definidas em contextos que variam por diferentes motivos. A componentização pode auxiliar na rápida reconfiguração do sistema, nos casos em que apenas o reparticionamento não resolve.

Os resultados desse capítulo, para serem implementados de forma sistematizada, porém flexível e reutilizável, demandaram um processo de análise e modelagem para a construção de um conjunto de ferramentas computacionais que facilitassem o desenvolvimento de estratégias de planejamento de movimento e simulações. O projeto de um *framework* computacional resultou dessa análise, sendo este apresentado no próximo capítulo.

5 ASPECTOS DE PROJETO DE UM *FRAMEWORK* PARA MODELAGEM CINEMÁTICA

Neste capítulo é discutido o projeto de um *framework* computacional elaborado a partir dos resultados da modelagem cinemática baseada em helicoides para sistemas de intervenção subaquática tratados nos capítulos anteriores. A motivação para o desenvolvimento desse *framework* e os requisitos de projeto identificados são relacionados. Destaca-se a escolha de uma abordagem orientada a objetos para o projeto, de forma a facilitar a extensão e o reuso. A modelagem computacional do *framework* é então apresentada, com uma visão geral e aspectos dos seus componentes. Maneiras de estender e especializar componentes e o próprio *framework* são então avaliadas.

5.1 MOTIVAÇÃO

A proposta de desenvolvimento de um *framework* para modelagem cinemática por helicoides surgiu da necessidade de sistematizar e facilitar a implementação de simulações dos sistemas de intervenção subaquática estudados. Observou-se que as ferramentas usualmente empregadas para esse fim não contém componentes de software que representem as entidades inerentes à modelagem por helicoides e às operações necessárias para seu uso. Além disso, os conceitos de modularidade de cadeias e seu reparticionamento, tal qual expostos no capítulo anterior, deveriam ser implementados como extensões de uma eventual biblioteca de helicoides já existente.

Outra motivação para a criação do *framework* foi a possibilidade de se organizar um banco de dados de descrições de cadeias cinemáticas que poderiam ser reutilizadas em várias simulações. Isso se daria através da leitura e processamento de um arquivo (ou um registro de um SGBD¹) de descrição por parte de um componente de software que gerasse um modelo funcional da cadeia para uso em uma simulação ou mesmo em uma implementação real. Outros usos desse banco de descrições envolveriam a composição de cadeias mais complexas, sistematizando e estendendo o desenvolvimento de novos cenários de simulação.

Por fim, uma vez que se desejava também explorar a possibilidade de automatizar a variabilidade do modelo cinemático ao longo da

¹Sistema Gerenciador de Banco de Dados.

execução de tarefas, concluiu-se que o uso de um *framework* que sistematizasse esse modelamento e atendesse às demandas para reconfiguração e particionamento de cadeias através de reuso seria mais vantajoso do que implementar cada modelo cinemático como um novo caso.

5.2 REQUISITOS DE PROJETO

A análise inicial do processo de modelagem cinemática por helicoides, sua componentização e sua variabilidade, bem como o modo de aplicação dos modelos para a resolução cinemática de tarefas em simulações, resultou na identificação de requisitos que deveriam ser considerados no projeto de um *framework* para esse fim (Rocha; Tonetto; Dias, 2011a). A capacidade de representar helicoides, seus comportamentos e operações realizadas sobre eles (como transformações de referenciais) é um requisito primário, pois estes são elementos básicos da modelagem cinemática.

A capacidade de representar cadeias cinemáticas de forma sistemática e estruturada é outro requisito primário. Uma entidade do *framework* capaz de representar toda uma cadeia cinemática deve ser capaz de fornecer as informações usualmente utilizadas na análise cinemática por helicoides. Isso implica na obtenção de grafos de movimento que definam a relação entre os elos da cadeia (ou mesmo em utilizá-los nas estruturas de dados de representação). No caso de cadeias fechadas, ter a possibilidade de se obter um conjunto de circuitos independentes, de forma que seja possível definir a equação de restrição da cadeia. A automação da resolução cinemática através do método de Davies é uma ação desejável, desde que para isso seja possível definir o particionamento da equação de restrição na estrutura de dados da cadeia.

Visando a componentização de cadeias, essa entidade deve ser capaz de representar tanto cadeias simples (formadas por elos e juntas) quanto cadeias compostas (formadas por elos, juntas e subcadeias). Uma adaptação do conceito de *joint in the bag* (Phillips, 2007) é uma forma de tratar juntas simples e subcadeias em uma mesma modelagem, como elementos que vinculam dois elos impondo uma restrição de movimento relativo entre eles. Para facilitar a reconfiguração de cadeias, devem existir métodos que permitam adicionar e remover elos e *conexões* entre eles (sejam juntas ou subcadeias). Da mesma forma, o reparticionamento deve ser de fácil execução. As modificações nessa entidade devem então ser refletidas na obtenção de informações do modelo, tais como a matriz de circuitos, a equação de restrição e mesmo

a sua resolução pelo método de Davies.

Um requisito desejável para o *framework* é a capacidade de descrever as cadeias cinemáticas em forma de texto estruturado. Além de simplificar a definição destas, tal recurso facilita a implementação de um banco de dados de cadeias, bem como seu reuso. Para tanto, o *framework* deve prover um analisador de descrições e gerador de entidades que representem as cadeias. O uso de alguma linguagem estruturada de descrição de dados como a XML (*eXtended Markup Language*) (Décio, 2000; W3C, 2011) favorece o uso de ferramentas computacionais prontas para tratamento desse formato de dados.

A modularidade é um requisito importante para a adoção do *framework* em diferentes casos, tanto em simulações quanto em casos reais. Um conjunto de componentes de software modulares, com comportamentos bem definidos e conhecidos, dá ao usuário a possibilidade de fazer diferentes combinações desses componentes para adequá-los às suas necessidades. Além disso, a substituição de componentes com comportamentos iguais (ou de versões aprimoradas) é feita sem um impacto grande sobre o código implementado. Com isso, pode-se trabalhar com componentes que façam a interface com um sistema físico (um manipulador, por exemplo) ou com um modelo matemático que o simule. Isso é útil quando da transposição de um código desenvolvido e testado em um ambiente simulado para um ambiente real.

A extensibilidade é outro requisito identificado como importante para o *framework*. A capacidade de estender e criar novas representações para as cadeias cinemáticas e seus componentes é importante para viabilizar otimizações de desempenho, correções de erros ou implementações de situações não previstas na análise inicial do sistema. A extensibilidade é muitas vezes associada à modularidade, pelo fato de que pode-se criar novos módulos para um determinado fim que mantenham o comportamento básico de módulos já existentes, facilitando a sua adaptação em software já desenvolvido e o seu uso por parte dos usuários do *framework*. A extensibilidade também permite considerar a especialização de entidades, pela criação de novos componentes que mantenham os comportamentos definidos, mas realizando as operações de formas diferentes. Isso muitas vezes é feito para otimizar desempenho de execução dos códigos, às custas de perda de generalidade.

Uma vez que o *framework* deve lidar com diferentes tipos de cadeias cinemáticas, e que a resolução de sua cinemática é feita em função de especificações de movimentos e restrições para as suas variáveis primárias, observou-se que um requisito desejável é o de se ter um conjunto de interpoladores de trajetória que forneça esses valores em função dos

caminhos especificados nas tarefas. Assim, serão geradas as estruturas de dados compatíveis com as que devem ser utilizadas para especificar posições e velocidades das juntas. É importante definir o comportamento básico desses interpoladores, com algumas implementações que viabilizem testes de funcionamento e que, através das características de modularidade e extensibilidade do *framework*, permitam aos usuários implementarem de forma sistemática os geradores de trajetória que precisem.

Por fim, uma vez que se deseja utilizar o *framework* tanto em implementações reais (que executam em um programa embarcado no sistema dos UVMS) quanto em simulações, bem como para análise, verificou-se que é importante que esses componentes funcionem tanto em ambientes interativos (característicos de ambientes de computação numérica/simbólica) quanto em software completo (que roda autonomamente, sem intervenção direta de usuários no seu fluxo de execução). Esse requisito influencia o modo como os comportamentos devem ser invocados para execução, bem como na definição de estruturas de dados que usuários possam manipular diretamente. Além disso, esse requisito tem influência direta na escolha de uma plataforma computacional de desenvolvimento. A escolha da plataforma computacional e aspectos de implementação do *framework* são o tema do Apêndice B.

5.3 O ENFOQUE ORIENTADO A OBJETOS

A adoção do paradigma da orientação a objetos para o desenvolvimento do *framework* baseou-se na sua adequação para o atendimento de vários requisitos do projeto, na grande base de conhecimento existente sobre a análise e o desenvolvimento de sistemas e na experiência prévia com esta abordagem. Os quatro princípios fundamentais de construção de sistemas orientados a objetos – abstração, encapsulamento, herança e polimorfismo – garantem os requisitos de modularização/componentização, extensibilidade e especialização (Pressman, 1995; Fowler, 2004).

O encapsulamento e a abstração permitem descrever comportamentos aos objetos de forma que os usuários destes não se preocupem com sua real implementação, facilitando a utilização destes componentes e também o seu desenvolvimento por diferentes programadores.

A herança e o polimorfismo permitem definir hierarquias de classes de objetos que tem comportamentos similares executando ações distintas de acordo com as especificidades de cada classe. Com isso

define-se um comportamento padrão para geradores de trajetória que permite empregá-los da mesma maneira independentemente dos algoritmos neles implementados, por exemplo. Assim, sabendo como utilizar uma das classes de uma hierarquia, é possível utilizar ou substituir uma instância de outra classe da mesma hierarquia.

Além disso, foi possível relacionar as entidades utilizadas na análise cinemática dos capítulos anteriores com classes de objetos. Esse relacionamento foi possível não apenas na definição de estruturas de dados, mas também na definição de comportamentos/ações que as entidades normalmente realizam e nos interrelacionamentos entre elas.

Apesar dos ambientes de computação numérica usualmente empregados em simulações não se basearem na orientação a objetos, é possível adaptar implementações para programação estruturada, com alguma perda de características que tornam a orientação a objetos vantajosa. Por outro lado, várias plataformas de desenvolvimento de software já se baseiam no paradigma de orientação a objetos, o que torna a implementação direta. Java, Python e C++ são típicos exemplos de linguagens de programação orientadas a objetos que são utilizadas em engenharia (Deitel; Deitel, 2004; Python.org, 2011; Ziviani, 2006).

A *Unified Modeling Language*, ou UML, foi usada para descrever o projeto e facilitar sua análise e implementação, em particular o diagrama de classes. Além de ser um padrão *de facto* da área de desenvolvimento de software, a UML tem aplicações em diferentes campos do conhecimento, e é mantida e padronizada por um órgão gestor (Fowler, 2004; Booch; Rumbaugh; Jacobson, 1999). Assim, pretende-se que a documentação do *framework* seja facilmente compreendida por desenvolvedores que queiram colaborar com o seu desenvolvimento futuro.

Uma das características particulares de ter-se adotado o paradigma da orientação a objetos para o projeto do *framework* foi observar que algumas classes, atributos e métodos enquadravam-se em casos bastante estudados na literatura conhecidos como *padrões de projeto* (Metsker, 2004). Assim, foi possível adotar soluções já definidas e consagradas pelos desenvolvedores de sistemas, com ganhos de tempo de projeto e de implementação.

5.4 MODELAGEM DO *FRAMEWORK*

O *framework* foi denominado **Kast**, acrônimo de *Kinematic Analysis by Screw Theory*, ou *Análise Cinemática por Teoria dos Helicoides*, em português. A ideia era ter um nome curto, porém significativo da

finalidade do *framework*².

5.4.1 Visão Geral

O diagrama de classes da Figura 31 apresenta uma visão geral dos componentes do *framework* Kast. São apresentadas as principais classes, além de algumas que são usadas na composição ou transformação das primeiras.

O *framework* se concentra na hierarquia de classes que parte de *KCComponent*. Tanto helicoides quanto cadeias cinemáticas são implementados como classes derivadas dessa classe base. Em *KCComponent* define-se um conjunto de atributos e métodos comuns a helicoides e cadeias cinemáticas. O comportamento diferenciado dos métodos é descrito na implementação das classes específicas. Além disso, nestas são definidos novos métodos e atributos inerentes às particularidades das entidades que representam. Porém, a assinatura básica definida em *KCComponent* facilita o aprendizado de uso de todas as classes dessa hierarquia.

Nas cadeias cinemáticas, destaca-se a classe *KCComposable*, que as representa de modo totalmente configurável. Instâncias dessa classe permitem definir a estrutura da cadeia cinemática através da adição e remoção de elos e juntas em um grafo de movimento. Os vértices são instâncias da classe *Link* e as arestas são instâncias da classe *Joint*, que representam elos e juntas, respectivamente.

Também são relacionadas classes de suporte a operações com helicoides, como a *ScrewTransformation*. Esta representa uma transformação de helicoides, cuja matriz é calculada em função de informações de posição e orientação de um referencial em relação ao outro. Estas são supridas por uma interface padronizada definida na classe *Posture* e implementada por suas classes derivadas. As principais classes de objetos que representam atributos de *KCComponent* e associados são a *Attribute*, *AttributeList* e *State*. Uma classe auxiliar à manipulação de descrições textuais é a *KCFactory*, que utiliza instâncias de *KCParser* para analisar o código dessas descrições.

Na análise do diagrama de classes, são identificadas situações onde as técnicas de padrões de projeto podem ser aplicadas. Tais técnicas são soluções generalizadas para problemas de modelagem estrutural em software orientado a objetos. Esses padrões definidos, testados e do-

²Ao longo do texto, esse nome poderá aparecer em letras maiúsculas ou minúsculas, ou seja, *KAST*, *Kast* ou *kast* têm o mesmo significado.

cumentados na literatura definem relações entre classes de um sistema. Os padrões podem ser reutilizados em casos onde se percebe a ocorrência de situações similares, mesmo que entre domínios de problemas totalmente diferentes entre si.

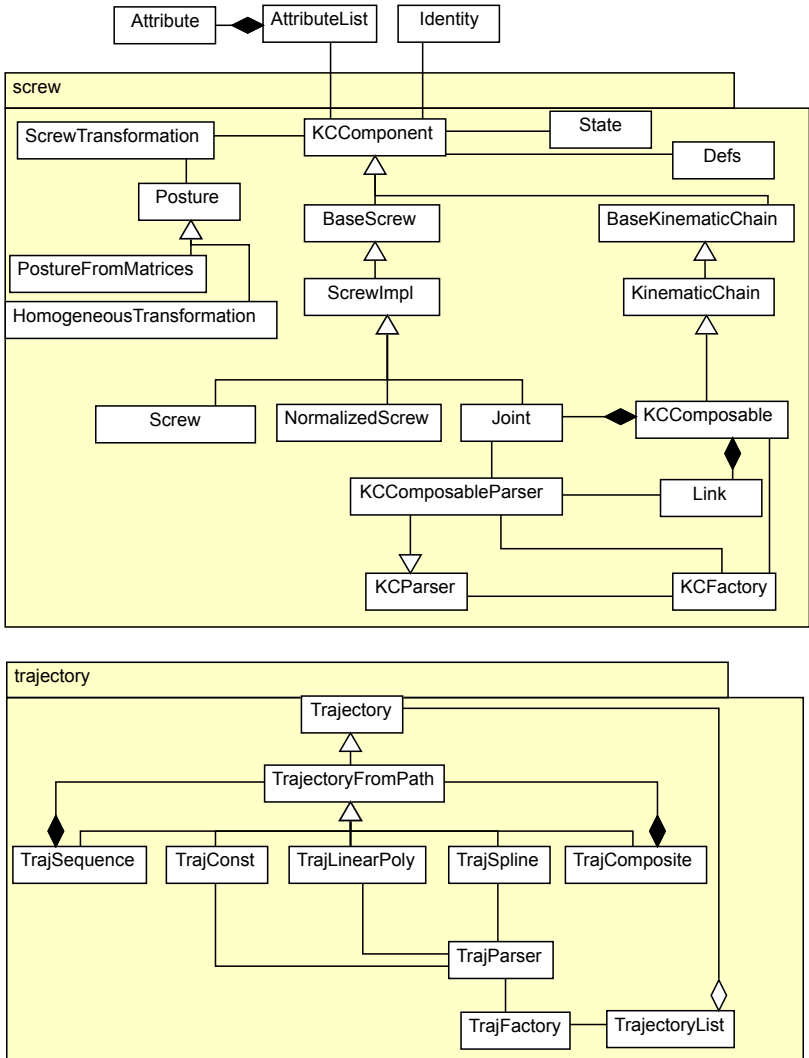


Figura 31. Diagrama de classes do *framework* Kast

O padrão *composite* descreve relações entre peças que por sua vez podem ser compostas por peças menores de mesma natureza. Ele está presente na representação de cadeias cinemáticas, onde uma conexão entre dois elos pode ser tanto uma junta quanto uma subcadeia.

O padrão *observer*, apesar de não aparente, torna-se necessário para que as modificações em componentes dos helicoides e cadeias cinemáticas sejam percebidas pelos objetos que os contém e o acesso a suas informações reflitam essas modificações. Para tanto, as classes apresentadas no diagrama herdam características de classes que implementam uma adaptação de *observer* denominadas *Listener* e *Subject*.

O padrão *façade* aparece na definição de uma interface comum para acesso a tipos de dados diferentes por parte de um mesmo objeto. Esse é o caso da classe *Posture*, cujas classes dela derivadas fornecem um mesmo método para fornecer dados para instâncias de *ScrewTransformation* calcularem sua matriz de transformação.

Além desses, cita-se ainda o caso do padrão *factory method* implementado pela classe *KCFactory*, que instancia objetos da hierarquia de *KCComponent* a partir de descrições textuais armazenadas em arquivos ou em variáveis.

Embora não fosse o objetivo inicial do *framework*, uma hierarquia de classes geradoras de trajetórias também foi definida. Tal desenvolvimento se justifica pelo uso destes na obtenção de referências de movimento pela interpolação de pontos de caminhos especificados nas tarefas. Assim, decidiu-se por definir uma assinatura de classe composta por estruturas de dados e ações que facilitassem a comunicação de dados entre a tarefa e as instâncias de cadeias cinemáticas, que por sua vez resolverão a cinemática a partir desses dados. Estes, por sua vez, são representados pelas classes *State* e *StateList*. Junto com essa hierarquia, também foi definida uma implementação de *factory method* embutida na classe *TrajFactory* para análise e criação de geradores de trajetória a partir de descrições textuais de tarefas.

As definições das classes mais importantes para uso do *framework* são discutidas a seguir, de modo a facilitar seu entendimento para o usuário do *Kast* em suas implementações ou mesmo para estender o *framework*.

5.4.2 A Classe *KCComponent*

A classe *KCComponent* é a fundação de classes que representam tanto helicoides quanto cadeias cinemáticas. A adoção de uma raiz

comum a essas entidades foi a percepção de que havia similaridades em boa parte das estruturas de dados bem como em certos comportamentos. Assim, `KCComponent` define uma assinatura básica para uso dessas entidades e um conjunto de dados comuns a elas.

Em relação às estruturas de dados, são cinco as fundamentais, que são definidas na instanciação de objetos de uma classe e podem ser manipuladas através de propriedades³ da classe:

identity É um identificador de uma entidade. Identidades são instâncias da classe `Identity`, que contém propriedades que define um identificador numérico único (`id`), um nome textual (`name`) e um texto descritivo (`description`). `identity` é implementado como um dicionário, ou seja, como se fosse organizado em pares *chave-valor*. Além disso, é possível definir atributos adicionais, o que torna-se importante para definir a ordem de processamento de helicoides no processamento da matriz de rede da classe `KCComposable`, por exemplo. Outra finalidade dessa informação é descrever entidades em uma interface gráfica com o usuário.

system Definição do sistema de helicoides da entidade. Ele é definido por um vetor de seis elementos com valores 1 e 0, descrevendo quais componentes de um helicóide são utilizados nos cálculos (1's). Para isso, assume-se que um helicóide expresso em coordenadas de Plücker tem a forma $[\omega_x \ \omega_y \ \omega_z \ v_x \ v_y \ v_z]^T$. Os sistemas mais comuns de helicoides são definidos e nomeados na classe `Defs` para facilitar o uso.

attributes São dados que não costumam variar ao longo do tempo, mas que são parâmetros de definição de um helicóide ou uma cadeia cinemática. É uma instância da classe `AttributeList`, que é uma implementação de dicionário que notifica a outros objetos que monitoram mudanças nos valores de cada chave. Dimensões de elos, por exemplo, são armazenadas nesse dicionário.

state São dados que variam ao longo do tempo, e que normalmente estão envolvidos no movimento de cadeias, como as variáveis de juntas. Para isso, foi criada uma classe denominada `State`, que essencialmente é uma matriz indexada por uma chave textual (o nome de uma variável) e um indicador do tipo de dado solicitado (posição, velocidade ou aceleração).

³Métodos *getters* e *setters*, que retornam uma informação ou definem uma informação, respectivamente.

numeric É uma matriz que armazena valores numéricos de um helicóide (ou de uma matriz de helicóides de uma cadeia). Os valores são calculados por métodos das instâncias de classes derivadas de `KCCComponent`, a partir de dados de `attributes` e `state`.

Todos esses dados, com exceção de `identity`, podem ser compartilhados entre diferentes instâncias de classes derivadas de `KCCComponent`. Isso garante economia de definição de dados para várias entidades usadas em um código. No caso de uma cadeia cinemática, os seus dados são compartilhados para os helicóides que fazem parte dela a fim de que sejam calculados os seus componentes. Para isso, pode-se criar as instâncias dessas estruturas de dados antes de instanciar os objetos da hierarquia de `KCCComponent` e utilizar essas estruturas na instanciação dos helicóides e cadeias. Uma representação gráfica de `attributes` e `state` é apresentada na Figura 32.

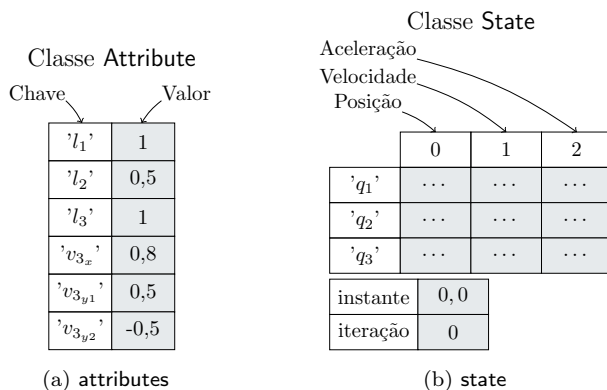


Figura 32. Representação gráfica de estruturas de dados usadas em `KCCComponent`

Além de acesso às informações, são definidos alguns comportamentos básicos, como `magnitude`, que deve retornar o valor da magnitude dos helicóides, de forma complementar a `numeric`, que deve retornar os helicóides propriamente ditos (na maioria dos casos, helicóides normalizados). `update` seria o método que faz o cálculo dos valores dos helicóides e eventualmente da magnitude, dependendo da classe que o implementa.

`KCCComponent` é uma classe abstrata, ou seja, não é possível instanciar objetos a partir dela. Parte dos métodos definidos são apenas

assinaturas, que devem ser implementadas em cada especialização nas classes derivadas. É o caso de `update`, que executa ações diferentes de acordo com o tipo de objeto. Além disso, esse método é concebido *privado*, ou seja, não deveria ser invocado externamente a um objeto. Métodos com essa visibilidade são executados apenas por outros métodos da classe. No caso de `update`, ele é invocado cada vez que alguma informação sobre os helicoides fosse acessada através de uma propriedade.

5.4.3 Helicoides e Juntas

No `Kast`, helicoides são representados por instâncias de `Screw`, `NormalizedScrew` ou `Joint`. As três classes são derivadas de `ScrewImpl`, que por sua vez descende da classe `BaseScrew`. Estas duas últimas classes especializam a definição inicial da classe raiz `KCCComponent`.

A classe `BaseScrew` acrescenta algumas propriedades e métodos. O passo do helicóide é definido pelo atributo `type`, que é um escalar. O nome do atributo se deve aos atalhos para juntas rotacionais (passo 0) e translacionais (passo ∞) definidos na classe de apoio `Defs`. O outro atributo é `components`, cuja finalidade é descrever como os componentes do helicóide são definidos. Além disso, deve-se incluir dois atributos na definição da identidade de um descendente de `BaseScrew`: `var` e `order`. O primeiro é o nome da chave do atributo `state` correspondente à magnitude do helicóide (e no caso de uma junta, que também armazena sua posição e aceleração). O segundo indica a ordem de processamento desse helicóide durante a atualização de uma cadeia, além de definir sua posição na matriz de rede e no vetor de magnitudes correspondente.

A implementação dessa assinatura começa a ser feita na classe `ScrewImpl`. Nela, `components` pode tanto ser uma função que calcula todos os componentes de helicóide (preenchendo a matriz `numeric`) quanto um vetor de seis elementos, um para cada componente do helicóide, que podem ser um escalar ou uma função que define somente aquele componente. A implementação do método `update` utiliza `components` para executar o cálculo do helicóide.

A classe `Screw` é uma implementação que segue a expressão de um helicóide típico em coordenadas de Plücker, sendo mais usado para estudo dos helicoides propriamente ditos do que em cadeias cinemáticas. A implementação do método `decompose`, definido em `BaseScrew`, retorna uma tupla formada pela magnitude do helicóide e por seu helicóide normalizado.

`NormalizedScrew` é uma implementação ligeiramente diferente de `Screw` onde o helicóide é sempre normalizado. Assim, o método `update` sempre normaliza os componentes calculados e a magnitude retornada é sempre igual a 1.

Visando o uso em cadeias cinemáticas, implementou-se a classe `Joint`. Esta decompõe o helicóide em seu helicóide normalizado e sua magnitude. Em `Joint`, a magnitude pode ser definida por sua propriedade, e o helicóide é sempre normalizado após ser calculado. A magnitude é vinculada ao campo velocidade da variável correspondente do dicionário `state`.

5.4.4 Cadeias Cinemáticas

Cadeias cinemáticas são representadas por classes derivadas de `BaseKinematicChain` e `KinematicChain`. A primeira classe define a assinatura básica de objetos que representam cadeias cinemáticas. Embora a princípio não apresente diferenças conceituais em relação à `KCComponent`, a sua existência se justifica pela definição de uma interface que pode ser modificada em evoluções do *framework*, sem que com isso as classes de helicóides acabem sofrendo a modificação. `KinematicChain` é derivada de `BaseKinematicChain`, acrescentando métodos e atributos nessa especialização. A especialização de cadeias pode se dar por qualquer das duas classes, embora observe-se que `KinematicChain` tem métodos e atributos mais voltados para cadeias fechadas.

O atributo `partitioning` é um acréscimo em relação à `KCComponent`, e serve para definir quais variáveis de juntas de uma cadeia fechada são primárias e secundárias, de forma que o método `solveDavies` possa ser executado. Como o nome diz, o método determina as magnitudes das velocidades de juntas definidas como secundárias em função das juntas primárias. Para definir esse particionamento, `partitioning` consiste em um vetor de tamanho igual ao número de helicóides da cadeia cinemática onde cada célula indica se um helicóide está na partição primária (com valor 1) ou secundária (com valor 0).

Para a execução do método `solveDavies`, que é implementado na própria classe `KinematicChain`, é necessário que o método `update` seja especializado, de forma a calcular todos os helicóides normalizados da cadeia, organizá-los na matriz de helicóides (armazenada no atributo `numeric`), e montar a matriz de rede (armazenada no atributo `netMatrix`). `update` não é implementada nessa classe, devendo sê-lo nas classes derivadas.

Uma implementação concreta e funcional de cadeia cinemática é a classe `KCComposable`. Ela é voltada para a modelagem dinâmica da cadeia através da definição de sua estrutura usando um grafo de movimento ao qual são vinculados elos (instâncias da classe `Link`) e juntas (definições de heligiros instanciadas da classe `Joint`). Essa implementação é interessante em trabalhos interativos de definição de cadeias cinemáticas, para seu projeto e análise. Outra utilidade é a automação do trabalho de modelagem por software com interfaces gráficas com o usuário.

A flexibilidade de definição de elos e juntas se dá através de métodos para manipular essas entidades (`addLink`, `delLink`, `addJoint`, `delJoint`, entre outros). As instâncias de `Link` compartilham as definições de estruturas de dados `identity` e `attributes`.

O método `update` de `KCComposable` é definido de forma a otimizar o cálculo da matriz de rede, mesmo com tantos dados. Para tanto, assume-se que, uma vez definida a estrutura do grafo de movimento e determinados sua matriz de circuitos e a matriz de helicoides, novas atualizações das variáveis de juntas apenas necessitarão de atualizações da matriz de helicoides e preenchimento da matriz de rede. O pseudocódigo do Algoritmo 1 resume esses passos.

Algoritmo 1. Pseudocódigo de `KCComposable.update`

```

se atualizacao = total então
    | calcularMatrizDeCircuitos();
    | redimensionarMatrizDeRede();
    | atualizacao ← normal;
fim se
para cada junta j de grafoDeMovimento faça
    | j.atualizar();
fim para cada
preencheMatrizDeRede();

```

Um método adicional para auxiliar o acesso aos dados da cadeia por parte dos helicoides é o `pack`. Ele tem a finalidade de unificar os conjuntos de atributos `attributes`, `state` e `numeric` reunindo os dados de outras instâncias desses atributos que eventualmente algum elo ou junta utilizem. Novos conjuntos de atributos criados pelo método, que reúne todas essas definições, substituem os conjuntos individuais em todos os elos e juntas da cadeia. Assim, todas as entidades compartilham o mesmo conjunto de dados.

Um formato de descrição textual de cadeias utilizando `KCComposable`, baseado em XML, foi definido para facilitar tanto a criação dessas cadeias quanto o seu reuso. Arquivos contendo essas descrições podem ser utilizados para criar instâncias de `KCComposable` através de objetos da classe `KCFactory`. Esta funciona como um padrão de projeto *factory method*, cuja finalidade é a de criar objetos de uma família de classes em função de definições externas. Classes de analisadores são necessárias para isso, implementando a interface definida pela classe base `KCParser`. Para descrições de `KCComposable`, existe uma classe derivada chamada `KCComposableParser` que faz a análise de elementos específicos da descrição para esse tipo de objetos.

5.4.5 Geradores de Trajetória

A hierarquia de geradores de trajetória foi incluída em `Kast` para complementar o uso das classes de helicoides e cadeias cinemáticas na implementação de simulações.

A fim de compatibilizar as trajetórias resultantes dos geradores com as variáveis das cadeias cinemáticas, ambas utilizam a mesma estrutura de dados baseada na classe `State`. Isso vale também para a especificação dos pontos do caminho por onde uma trajetória deve passar (que seria a entrada de dados de um gerador). Para trabalhar com múltiplos pontos, foi criada uma classe agregadora denominada `StateList`. Ela funciona como uma lista de instâncias de `State`, com facilidades de acesso e modificação dos valores de um ponto particular, redimensionamento e pesquisa em função de tempo ou de número de interações.

A classe raiz dos geradores é a `Trajectory`. Nela é definida uma assinatura que deve ser seguida e implementada pelas demais classes geradoras de trajetória. As principais estruturas de dados necessárias para um gerador são:

interval Identifica o intervalo de tempo em que ocorrerá a geração de trajetória. É composto por três escalares, que definem o instante inicial, o instante final e o passo de iteração (Os geradores inicialmente implementados são de passo fixo). Esses valores estão relacionados de forma que a divisão do intervalo pelo passo sempre resulte em um número inteiro de iterações. Para isso, a classe deve prover ajustes nos valores do intervalo para garantir essa condição e que o instante inicial e o instante final correspondam à primeira e à última iteração, respectivamente.

attributes São dados que servem de parâmetros para o gerador de trajetória. Dependem de implementação específica de cada tipo de gerador. É uma instância da classe `AttributeList`.

vars Vetor contendo os nomes das variáveis cujas trajetórias serão calculadas. É utilizado na definição da `StateList` do gerador.

identity Instância da classe `Identity`, usada para descrever textualmente o gerador e identificá-lo unicamente no código. De interesse para criar interfaces com o usuário.

Para o caso de geradores baseados em informações de caminhos que devem ser seguidos, foi definida a classe derivada `TrajectoryFromPath`, que acrescenta às estruturas de dados anteriormente descritas a estrutura `path`, que é uma instância de `StateList` onde cada elemento representa uma referência, com tempo ou interação desejada. Cada referência é uma instância de `State`, onde cada variável contém dados de posição, velocidade e aceleração que podem ser especificadas.

Foram definidos quatro geradores de trajetória concretos, para uso imediato do *framework* e demonstração de implementações. A classe `TrajLinearPoly` define trajetórias lineares entre cada dois pontos do caminho, utilizando um polinômio de 5ª ordem para suavizar o movimento. A classe `TrajSpline` utiliza B-splines cúbicas para gerar uma trajetória que passa por todos os pontos do caminho. Para uniformizar o tratamento de valores constantes, criou-se a classe `TrajConst`, que retorna um valor constante entre dois pontos de um intervalo do caminho. Para cada uma dessas classes, os métodos `update` e `calcPoint` são os que implementam o algoritmo de geração de trajetórias particular.

Se for necessário que uma trajetória tenha trechos onde diferentes geradores devem ser empregados, pode-se utilizar a classe `TrajSequence`. Ela implementa suporte para definir um conjunto de geradores e respectivos trechos da trajetória em que eles serão empregados.

O comportamento de todos os geradores, independente de seu algoritmo de geração de trajetória ou da definição dos dados necessários para seu funcionamento segue uma mesma interface, definida na classe `Trajectory`. É implementado um método `next`, que a cada chamada retorna o resultado do cálculo de uma iteração do gerador, avançando o instante de tempo a seguir em um passo de simulação. Se necessário, pode-se avançar ou retroceder dentro do intervalo de tempo da geração utilizando o método `mark`. Outra variação é o método `at`, que retorna uma instância de `State` para um instante requerido. O método `generate`, por sua vez, gera um `StateList` com toda a trajetória de uma só vez.

Assim como no caso das cadeias cinemáticas, foi definida uma descrição textual baseada em XML para caminhos e trajetórias. Ela é específica para o caso de geradores derivados de `TrajectoryFromPath`, e por isso consiste na especificação de pontos que devem ser percorridos na trajetória com o instante de tempo desejado. Junto com isso, também se especificam os geradores de trajetória desejados para as variáveis. A classe `TrajFactory` cria os objetos geradores de trajetória de acordo com as classes especificadas nas descrições. Para tanto, os dados são analisados por um objeto da classe `TrajParser`, que cria as estruturas de dados necessárias para a instanciação do gerador de trajetória e descreve para `TrajFactory` qual deverá ser criado. Com isso, é possível pensar em definir tarefas diretamente através dessas descrições textuais.

5.4.6 Extensão e Especialização de Cadeias Cinemáticas

O *framework* foi definido com um conjunto mínimo de funcionalidades implementadas, porém completamente funcional. É possível trabalhar exclusivamente com a classe `KCComposable` para tratar da análise cinemática de cadeias, bem como definir estratégias de planejamento de movimento. Cadeias definidas em objetos `KCComposable` podem ser definidas pelo usuário em um ambiente interativo, ser dinamicamente modificadas ao longo da execução de uma tarefa, além de poderem ser definidas textualmente em um arquivo.

Essa flexibilidade, apesar de atender às necessidades de modelagem cinemática, tem um custo associado relativo ao desempenho dos cálculos computacionais. Uma vez que cada helicóide é calculado separadamente, as equações que os definem são inteiramente calculadas, sendo que alguns trechos são repetidos várias vezes. Além disso, o preenchimento da matriz de rede exige várias cópias dos helicóides.

Por esse motivo, pode ser interessante trabalhar com especializações das cadeias cinemáticas. Isso é feito através da criação de classes derivadas de `KinematicChain`, onde o método `update` deve ser implementado para realizar o cálculo dos helicóides. Aí pode-se definir otimizações como o pré-cálculo de termos que aparecem em várias equações, aproveitar resultados de algumas equações em outras, evitar cálculos de expressões com resultados constantes e outras que se fizerem necessárias. Para tanto, o modelo cinemático deve estar previamente obtido e analisado quanto a otimizações de cálculos possíveis.

A especialização de uma determinada cadeia cinemática em uma

classe não a torna específica para uma única situação de análise cinemática. Observa-se que ainda deve-se utilizar as estruturas de dados definidas nas classes base para poder parametrizar essa cadeia quando possível. Assim, apesar de se ter um único cenário descrito, pode-se trabalhar com diferentes configurações dimensionais e especificações de tarefas. Um exemplo disso é a possibilidade de reparticionar a equação de restrição pela simples modificação do atributo `partitioning`.

A principal desvantagem de se trabalhar com classes especializadas para uma dada configuração de cadeia cinemática é a perda da reconfigurabilidade, que pode ser necessária em casos como o evitamento de obstáculos não previstos. Outra possível desvantagem é a perda de clareza da definição da cadeia cinemática na implementação, que pode tornar difícil a manutenção do código.

Observa-se que a ideia de se desenvolver um *framework* foi definir um conjunto de interfaces que modelassem computacionalmente o problema da análise cinemática por helicoides. Além de ter implementações utilizáveis dessas interfaces, o objetivo de um *framework* é fornecer uma base para implementações, extensão e evolução de códigos, o que não seria o caso de uma biblioteca fechada ou de uma aplicação específica. Assim, existe um potencial para criação e aplicação do *Kast* em um grande conjunto de aplicações que necessitem de análise cinemática por helicoides.

O projeto do *framework Kast* foi tratado nesse capítulo. A sua estrutura geral e componentes foram descritos, bem como as possibilidades de uso e extensão. Além da representação de cadeias cinemáticas, verificou-se a importância de se ter componentes para gerar trajetórias, que foram acrescentados ao projeto do *framework*. O Apêndice B discute aspectos de implementação do *Kast*, com apresentação de um exemplo. O *Kast* auxiliará o uso dos conceitos de componentização e variabilidade de cadeias cinemática para o desenvolvimento de estratégias de planejamento de movimento, que será abordado no próximo capítulo.

6 ESTRATÉGIAS DE PLANEJAMENTO DE MOVIMENTO PARA UVMS EM INTERVENÇÃO

Neste capítulo discute-se as estratégias para o planejamento de movimento dos sistemas veículo-manipulador subaquáticos executando atividades de intervenção. Uma visão geral é apresentada, onde os blocos de atividades de UVMS envolvidos no planejamento de movimento são identificados e suas atribuições definidas. A seguir, uma descrição algorítmica da execução de uma intervenção é analisada, a fim de se poder modelar essa fase de missão. Com base no *framework* para análise cinemática apresentado no capítulo anterior, elabora-se um modelo orientado a objetos de componentes que representam os blocos de atividades do sistema de intervenção subaquática relevantes para o planejamento de movimento. A partir desse modelo, propõe-se então uma sistematização de implementação de estratégias de planejamento de movimento.

6.1 VISÃO GERAL DO PLANEJAMENTO DE MOVIMENTO EM SISTEMAS DE INTERVENÇÃO

Para iniciar a análise do problema de planejamento de movimento dos UVMS, procurou-se identificar quais os blocos de atividades de um sistema robótico subaquático estariam envolvidos na sua resolução. Foi considerado o aspecto local da execução da intervenção (compreendida como as fases de ativação dos manipuladores, execução da intervenção e desativação dos manipuladores) e a organização de atividades de um UVMS para essas fases estabelecida no Capítulo 2. Assim, com base no diagrama da Figura 10 fez-se essa identificação, que é ressaltada na rerepresentação do diagrama feita na Figura 33.

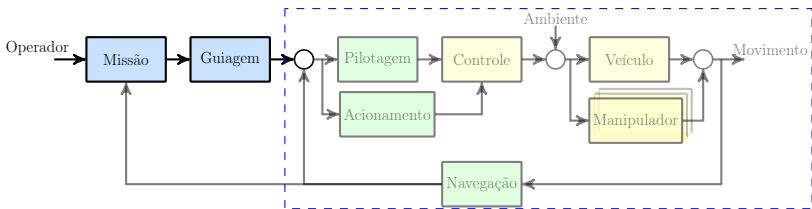


Figura 33. Atividades de UVMS envolvidas nas estratégias de planejamento de movimento

Em se tratando do planejamento de movimento, as atividades Missão e Guiagem são as diretamente envolvidas. As demais atividades se relacionam com as primeiras fornecendo dados para elas, como é o caso da navegação, ou consumindo as referências de movimento geradas para poder executar as ações do sistema robótico, como é o caso da pilotagem e do acionamento.

A atividade missão tem como atribuições avaliar continuamente o estado de operação do sistema, definir o modelo cinemático do sistema de intervenção compatível com o estado de operação e com a tarefa, fornecer referências relacionadas à tarefa e determinar o erro de postura do sistema.

A guiagem, por sua vez, utiliza o modelo cinemático definido por missão para determinar o movimento do sistema necessário para atender às especificações da tarefa e minimização do erro. Considerando a abordagem por helicoides para a resolução cinemática, isso envolve determinar as velocidades dos elementos atuados a partir das referências de velocidade da tarefa e subsequente integração para se definirem as posições desses elementos.

Por tarefa, entende-se o conjunto de posições e velocidades que o operador do sistema define para os elementos que devem executar a intervenção (efetadores finais dos manipuladores), o movimento/restrrição dos veículos envolvidos e as restrições impostas pelo espaço de trabalho ou condições do sistema. Em relação aos veículos, eventualmente seu movimento faz parte do conjunto da solução cinemática. Da mesma forma, as restrições podem não ser relevantes à execução da tarefa ou podem aparecer ao longo do tempo em que esta é efetuada.

6.2 EXECUÇÃO DE TAREFAS DE INTERVENÇÃO

A partir da visão geral do planejamento de movimento para as fases envolvidas na intervenção, foi elaborado um pseudocódigo que descreve a execução dessas fases. O Algoritmo 2 corresponde a esse pseudocódigo, que foi elaborado com enfoque orientado a objetos.

Considerando apenas a cinemática para a estratégia de planejamento de movimento, esse pseudocódigo pode ser reduzido ao apresentado no Algoritmo 3. Nele, os componentes que não são diretamente relacionados ao planejamento de movimento são agrupados em um objeto denominado **sistema**, que retorna apenas informações relativas à cinemática do sistema robótico. Esses blocos estão delimitados por uma fronteira na Figura 33. Assim, as linhas sombreadas do Algo-

Algoritmo 2. Pseudocódigo de execução das fases de intervenção de uma missão

```

configurarInicio();
para cada instante t de missão faça
    missão.verificarEstadoDeOperação();
    qd ← missão.tarefa.obter(t);
    qa ← missão.obterAtual();
    qr ← guiagem.executar(t, qd, qa);
    fr ← pilotagem.executar(qr);
    fa ← navegação.obterAmbiente();
    f ← controle.executar(t, fr, fa);
    qa ← uvms.executar(f);
    navegação.atualizar(t, qa);
    a ← navegação.obterAtual();
    missão.atualizar(qr, a);
fim para cada
finalizar();

```

ritmo 2 são substituídas pela linha sombreada do Algoritmo 3. Essa simplificação é utilizada para ressaltar a análise cinemática e facilitar o desenvolvimento de uma estratégia de planejamento de movimento focando apenas nos componentes relevantes à essa finalidade.

A definição da configuração inicial do sistema é atribuída ao método `configurarInicio`. No caso de uma simulação, ele deve instanciar os diversos objetos do processamento de informações do sistema robótico e executar as ações necessárias para que a missão comece a execução com as condições iniciais definidas. No caso de implementação em um sistema real, esse método marca o início das fases relacionadas à intervenção, com ativação dos manipuladores e posicionamento destes na configuração inicial de execução da tarefa.

O elemento central do pseudocódigo é um laço que repete um bloco de código a cada instante da tarefa a ser executada. A cada iteração do laço, objetos que representam as atividades do sistema de intervenção são acionados pela invocação de ações inerentes a eles.

O objeto `missão` tem por atribuições verificar o estado de operação do sistema (método `verificarEstadoDeOperação()`), refletindo mudanças de estado na estrutura da cadeia cinemática do sistema robótico e na geração de referências da tarefa. Além disso, gerencia o fornecimento de referências da tarefa (através do seu atributo `tarefa`). Por fim,

Algoritmo 3. Pseudocódigo de execução das fases de inter-
venção de uma missão - aspectos cinemáticos

```

configurarInicio();
para cada instante t de missão faça
    missão.verificarEstadoDeOperação();
    qd ← missão.tarefa.obter(t);
    qa ← missão.obterAtual();
    qr ← guiagem.executar(t, qd, qa);
    a ← sistema.executar(t, qr);
    missão.atualizar(qr, a);
fim para cada
finalizar();

```

agrega os dados de movimento do sistema, tanto as referências quanto os atuais (método `atualizar`) para o seu próprio processamento (verificação do estado do sistema) e para o registro/disponibilização dos dados ao usuário do sistema.

A tarefa a ser executada pelo sistema é representada pelo objeto `tarefa`, que é um atributo de `missão`. Ele tem por finalidade fornecer as referências de posição e movimento dos elementos primários da cadeia cinemática, usualmente envolvendo as posições e velocidades dos efetadores finais dos manipuladores. Dependendo do contexto de execução, o movimento dos veículos também pode ser imposto, bem como restrições adicionais à cadeia. Assim, esse objeto deve ser capaz de acompanhar as mudanças da cadeia cinemática para produzir as referências de acordo com o contexto de execução da missão.

O objeto `guiagem` resolve a cinemática do sistema, calculando as referências de movimento dos elementos atuados para que os UVMS realizem a tarefa especificada. Para isso, além de utilizar o modelo de cadeia cinemática para calcular as velocidades das variáveis secundárias, ele ainda deve poder integrar essas velocidades para gerar as posições dessas variáveis.

No pseudocódigo do Algoritmo 3, `sistema` é um objeto que representa todos os componentes que são externos ao planejamento de movimento. No caso de simulações, ele pode ser implementado dessa forma para fazer testes das estratégias de planejamento. Porém, ele deve ser considerado apenas como um artifício para isolar a fronteira do problema de planejamento de movimento, não substituindo as demais atividades em implementações reais ou em simulações que con-

siderem os efeitos dinâmicos na execução da missão. Pode-se, porém, pensar em sistema como um objeto *container* dos demais objetos, se se desejar manter essa abstração do restante do sistema em relação ao planejamento de movimento.

Após a execução das tarefas, algumas ações devem ser executadas para a finalização da intervenção. O método *finalizar* representa essas ações no pseudocódigo. No caso de simulações, ele consiste em executar códigos de encerramento dos objetos, limpeza da memória, pós-processamento, análise e apresentação dos dados. Em uma implementação em um sistema real, compreende a fase de desativação dos manipuladores e seu recolhimento.

6.3 DEFINIÇÃO DE UM *FRAMEWORK* PARA TAREFAS DE INTERVENÇÃO

Os objetos e métodos utilizados nos pseudocódigos levaram à identificação de um conjunto de classes que poderia ser projetado para estruturar a implementação de uma estratégia de planejamento de movimento dentro do contexto de execução. Este *framework* seria complementar ao *framework* Kast, fazendo uso deste e utilizando instâncias de suas classes para diferentes elementos da intervenção. O diagrama de classes da Figura 34 é uma descrição da estrutura do *framework* e suas associações com o Kast.

Além das classes dos objetos do pseudocódigo do Algoritmo 3, como *Mission*, *Task*, *Guidance* e *Uvms*, foi definida uma classe que gerencia a execução da intervenção chamada *Intervention*. Pode-se também associar essa classe à fase de intervenção de uma missão. Observe-se no diagrama as associações com as classes do *framework* Kast.

As informações que fluem de uma atividade para a outra no diagrama da Figura 33 são posições e velocidades relativas às variáveis de juntas da cadeia cinemática do sistema de intervenção. De acordo com o pseudocódigo do Algoritmo 3 tem-se valores desejados fornecidos pela tarefa (*qd*), valores atuais do sistema robótico (*qa*) e valores de referência obtidos pela resolução cinemática da guiagem (*qr*). No projeto do *framework* de intervenção, essas informações são representadas por instâncias da classe *State*. Além de serem da mesma classe, esses objetos têm a mesma lista de variáveis, que representam todas as juntas da cadeia cinemática. Isso facilita o intercâmbio entre objetos e permite que cada atividade tenha acesso a toda configuração da cadeia se assim necessitar.

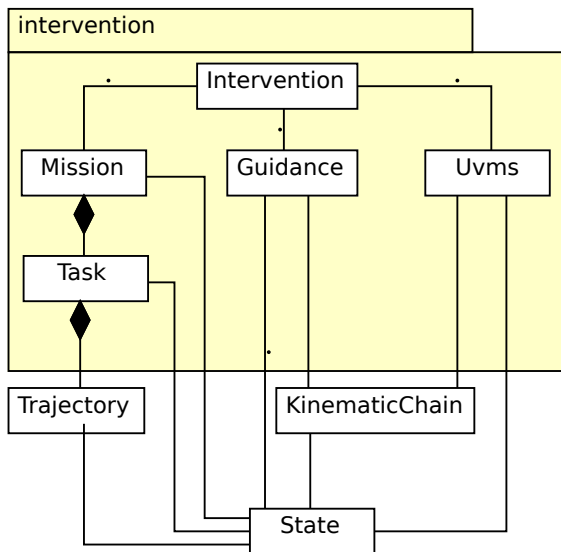


Figura 34. Diagrama de classes do *framework* de intervenção

Os modelos das cadeias cinemáticas são representados por instâncias de classes derivadas de `KinematicChain`. Por esse motivo, ela é incluída no diagrama de classes, para representar toda a hierarquia de classes que podem representar cadeias cinemáticas. Isso também aproveita a característica de encapsulamento da orientação a objetos, pois implica que todas as classes de cadeias cinemáticas devem implementar a mesma assinatura e assim serem utilizáveis da mesma forma, independentemente do arranjo cinemático do sistema representado.

Interpoladores de trajetórias são relevantes para a geração de referências da tarefa. Por isso, `Trajectory` representa toda a hierarquia de geradores que podem ser empregados em uma instância de `Task`, e que devem implementar a mesma assinatura de uso.

6.3.1 Classe `Intervention`

A fase de execução da intervenção da missão é descrita pela classe `Intervention`. Além de delimitar as fases, ela define as atribuições para a intervenção. Como é descrito no Algoritmo 4, a fase de intervenção inicia logo após a fase de ativação dos manipuladores (identificada pelo

método `ativarManipuladores` e encerra imediatamente antes da fase de desativação destes (identificada pelo método `desativarManipuladores`). À instanciação de um objeto gerenciador da intervenção segue-se a operação de instanciação dos seus objetos componentes e definição das condições iniciais da fase (método `init`). O método `execute` consiste no laço de execução da intervenção. Após encerrada a intervenção, o método `end` deve proceder com as ações de término da intervenção.

Algoritmo 4. Ações gerais da intervenção em uma missão

```
ativarManipuladores();
intervenção ← Intervention();
intervenção.init();
intervenção.execute();
intervenção.end();
desativarManipuladores();
```

O método `execute` é, então, o que representa toda a execução da missão. Observe-se que é essencialmente um laço que repete um trecho de código para cada instante da intervenção. Esse bloco de iteração consiste em verificar se houve mudanças de estado de execução da tarefa, atualizar o modelo cinemático se houve mudança de estado, gerar referências da tarefa a serem atingidas, determinar referências de movimento para o sistema robótico, aplicá-las e obter as variáveis de juntas do sistema e demais informações do ambiente para manter a missão atualizada. O Algoritmo 5 descreve o pseudocódigo desse método.

Algoritmo 5. Pseudocódigo de `Intervention.execute`

```
para cada instante t ← missão.t faça
    missão.verifyState();
    qd ← missão.task.q;
    qa ← missão.actualState;
    qr ← guiagem.execute(t, qd, qa);
    a ← uvms.execute(t, qr);
    missão.update(qr, a);
fim para cada
```

Deve-se observar que os objetos utilizados em `execute` são instanciados no método `init`.

6.3.2 Classe Mission

As atribuições da atividade missão relativas ao planejamento de movimento definem a classe `Mission`. Os elementos relevantes de sua assinatura são identificados ao se observar a instância dessa classe no Algoritmo 5 (objeto missão). São algumas propriedades (`t`, `task`, `actualState`) e métodos (`verifyState`, `update`).

As propriedades fornecem acesso aos atributos da classe. O instante de tempo da iteração em execução é dado por `t`, que é atualizado sequencialmente a cada vez que é acessado. O gerador de referências da tarefa é acessado pela propriedade `task`. As variáveis de juntas do sistema são fornecidas por `actualState`.

O método `verifyState` tem duas finalidades, que são avaliar se houve mudanças no contexto de execução da tarefa e atualizar os componentes necessários para que a estratégia de planejamento de movimento consiga atender às essas mudanças. Esses componentes são o modelo da cadeia cinemática e o gerador de referências da tarefa.

Para a primeira finalidade, o método deve utilizar as informações das variáveis das juntas do sistema robótico e do ambiente (obtidas através do método `update`). Essas informações podem provir de modelos conhecidos do meio ou de sensores do sistema robótico e do ambiente (Rocha et al., 2009). A identificação de mudança de contexto de execução utiliza algum método de tomada de decisão baseada nas informações disponíveis como, por exemplo, inteligência artificial com lógica difusa (Santos, 2006; Santos et al., 2006). Eventualmente, é possível que as informações coletadas indiquem a necessidade de mais de uma modificação de estado (por exemplo, uma junta de um manipulador travada e necessidade de evitamento de colisão). A tomada de decisões deve ser capaz de aplicar as mudanças correspondentes a essas modificações de estado. Considera-se que o sistema é capaz de aplicar sucessivas modificações sem que uma anule a outra, atendendo às restrições de operação.

Em relação à atualização do modelo da cadeia cinemática, é possível se dar tanto por reparticionamento como por reconfiguração, como foi discutido no Capítulo 4. Utilizando as definições de `Kast`, o reparticionamento consiste em se redefinir a propriedade `partitioning` de uma instância de `KinematicChain`. A reconfiguração depende de como o modelo da cadeia cinemática é implementado. Em instâncias da classe `KCComposable`, isso consiste em modificar o grafo de movimento da cadeia por adição e remoção de elos, juntas e subcadeias. Em classes especializadas para um determinado cenário de intervenção, o desen-

volvedor deverá criar métodos que viabilizem as mudanças possíveis no arranjo cinemático da cadeia. De todo modo, as modificações na cadeia devem ser definidas de acordo com os contextos detectáveis pelo sistema de tomada de decisão e verificação do contexto de execução.

Uma alternativa para reconfiguração seria manter mais de um objeto para a cadeia cinemática de um sistema de intervenção, cada qual descrevendo um arranjo cinemático correspondente a um contexto de execução da missão. A cada instante, um dos objetos assumiria a condição de ativo, de acordo com o contexto identificado. Essa possibilidade pode ser viável no caso das possíveis mudanças de contexto durante a execução de tarefas serem conhecidas de antemão.

6.3.3 Classe Task

A geração de dados da tarefa é gerenciada pela classe *Task*. Ela deve processar as especificações da tarefa, utilizar os geradores de trajetória necessários para calcular posições e velocidades e se modificar de acordo com as demandas de dados da cadeia cinemática.

Para o *framework* de intervenção, uma tarefa consiste nas definições de movimento desejados para a execução da intervenção (movimentos/restrições de efetuadores finais, peças e veículos), objetivos complementares e restrições impostas pelo cenário de execução. Isso se traduz em informações de velocidades e posições das variáveis das juntas primárias da cadeia cinemática que modela o sistema de intervenção, que por sua vez são representadas por instâncias da classe *State*.

Usualmente, essas definições consistem em caminhos ou especificações de curvas. Elas são tratadas pelos geradores da hierarquia de classes *Trajectory* do *framework* *Kast*. Isso é mostrado no diagrama de classes da Figura 34 pela agregação de *Trajectory* na classe *Task*. Isso também implica em que vários geradores de trajetória podem ser utilizados para determinar as referências da tarefa, de acordo com o tipo de movimento/restrrição desejado e o número de variáveis envolvidos.

A cada acesso da propriedade *q*, deve ser gerado um conjunto de dados das variáveis das juntas de toda a cadeia cinemática, sendo calculadas apenas as que correspondem à partição primária do modelo. As demais mantêm os valores determinados em outras atividades do sistema, como a guiagem.

Mudanças efetuadas no modelo da cadeia cinemática devem ser refletidas na tarefa. A cada instante, uma instância de *Task* deve for-

necer referências de posição e movimento para todas as variáveis da partição primária do modelo. Assim, modificações nesta demandam a adição e remoção de geradores de trajetórias associados às variáveis de juntas alteradas, enquanto os demais devem se manter para as variáveis que não foram modificadas. Isso pode ser verificado pelo acesso aos dicionários de variáveis das instâncias de `State` e pelo vetor de particionamento de da cadeia (instância de `KinematicChain`). O tipo de gerador de trajetória, por sua vez, depende dos requisitos de movimento/restrrição da variável que ele deverá atender. No caso de imobilização, ele deverá gerar um valor de posição constante, com velocidade igual a zero. Em situações onde exista movimento, o gerador utilizado deve ser capaz de lidar com os dados de especificação de tarefas disponíveis para ele. Este caso pode assumir certa complexidade em função de se determinar condições iniciais compatíveis com o seu estado anterior à modificação da estrutura cinemática do sistema e do perfil de suavização desejado.

A descontinuidade de movimento ocasionada pela mudança da cadeia cinemática é um problema que se deve muitas vezes ao fato de uma determinada variável ter imposta uma restrição de forma imediata, quando na realidade a junta à qual ela se refere já teria um movimento determinado antes da mudança da cadeia. Isso causa efeitos de elevadas acelerações nas juntas, com eventual saturação dos atuadores e esforços elevados que causam desgastes excessivos nos componentes físicos do sistema. Por esse motivo, deve-se ponderar o uso de alguma técnica de suavização das referências na transição entre estados de operação. Ela pode se dar pela determinação dos valores de posição e velocidade da junta em um momento imediatamente anterior à transição de estados com posterior aplicação de um interpolador entre este valor e o novo a ser adotado. No caso de planejamento *off-line* em ambientes completamente modelados, foi proposto um método em duas fases, onde na primeira a tarefa é executada e gerando as descontinuidades de movimento para com base nos dados obtidos a segunda fase define as transições necessárias em cada mudança de estado (Simas; Fontan; Martins, 2011).

A especificação de tarefas pode fazer uso da descrição XML definida em `Kast` e as classes que a processam.

6.3.4 Classe Guidance

A resolução da cinemática do sistema de intervenção é feita por instâncias de Guidance. Esta se dá em duas etapas, a de determinação das velocidades das variáveis da partição secundária da equação de restrição do sistema e a integração dessas velocidades para obtenção das posições dessas variáveis. A primeira etapa é realizada pela instância de KinematicChain que modela a cadeia cinemática do sistema e calcula a sua equação de restrição, enquanto a segunda é feita por algum algoritmo de integração. Esse processo é graficamente descrito pelo diagrama de comunicação da Figura 35.

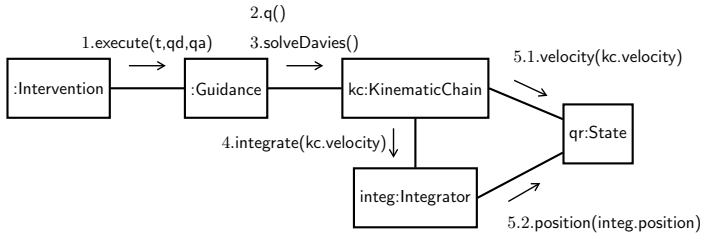


Figura 35. Diagrama de comunicação da resolução cinemática pela classe Guidance

A resolução da equação de restrição para as variáveis secundárias da cadeia é obtida pela execução do método `solveDavies` que é implementado em todas as classes derivadas de `KinematicChain`. Para tanto, é necessário que os dados de posição e velocidade de todas as juntas estejam atualizados para o cálculo dos helicoides normalizados e definição das velocidades das variáveis primárias. Isso é dado pelas informações das variáveis da cadeia passadas na invocação do método `execute` de `Guidance`.

Além disso, a solução da Equação 3.33 requer a inversão da partição secundária da matriz de rede (\mathbf{N}_s). Quando quadrada, ela é inversível (salvo estar em alguma condição singular). Em sistemas redundantes, porém, é possível que ela assuma uma forma retangular, denotando que existem mais variáveis a serem resolvidas que o espaço de helicoides do sistema. Nesse caso, uma possível solução é a aplicação de uma operação de pseudoinversão, como já foi discutido em capítulos anteriores.

O uso da pseudoinversão definida para a Equação 3.34 minimiza uma função custo de energia do sistema (Siciliano et al., 2009). Esse

operador, porém, gera soluções que distribuem o movimento igualmente entre as variáveis. Uma vez que o UVMS tem dois subsistemas de dinâmicas diferentes (veículo mais lento e manipulador mais rápido) essa solução não é necessariamente a mais indicada quando os movimentos de veículos e manipuladores devem ser obtidos. O uso de um operador de pseudoinversão como o da Equação 6.1 utiliza uma matriz de valores que possibilita maior controle sobre a distribuição do movimento (Antonelli, 2006),

$$\mathbf{N}_s^\dagger = \mathbf{W}^{-1} \mathbf{N}_s^T \left(\mathbf{N}_s \mathbf{W}^{-1} \mathbf{N}_s^T \right)^{-1} \quad (6.1)$$

onde \mathbf{W} é a matriz de ponderação, que deve ser definida positiva. Os pesos da matriz de ponderação podem ser definidos por uma função objetivo que faça com que eles variem de acordo com algum critério que mude a distribuição do movimento de acordo ao longo do tempo. O uso dessa solução para a inversão pode minimizar efeitos de descontinuidade de movimento, sendo combinado com mudanças de estado.

Operadores de pseudoinversão são passíveis de implementação através da especialização da classe `WPInv` do *framework* `Kast`, cujas instâncias podem ser vinculadas à objetos `KinematicChain` para serem utilizados na execução do método `solveDavies`. Observe-se que a redefinição desse operador é mais uma variação de mudanças no modelo cinemático do sistema de intervenção que pode ser executada pelas instâncias de `Mission`.

A integração pode tanto ser implementada internamente à `Guidance` quanto empregar classes específicas para esse fim. Como o uso de métodos numéricos ocasiona o efeito de *deriva*, erros são acumulados ao longo do tempo, o que pode acarretar a abertura de cadeias fechadas como as que são aqui tratadas. A compensação desses erros pode ser feita de diferentes maneiras. O método mais conhecido é o de cinemática inversa de malha fechada (Siciliano et al., 2009; Antonelli, 2006), onde uma malha de realimentação do erro é usada junto com as referências da tarefa na definição dos valores das variáveis primárias. A outra possibilidade proposta na literatura utiliza cadeias virtuais de erro (Simas, 2008) para representar a abertura da cadeia cinemática. O uso dessas cadeias é feito por adição destas como subcadeias na cadeia cinemática do sistema. Além disso, para as novas variáveis da cadeia de erro devem ser geradas referências conforme descrito em (Simas et al., 2009).

6.3.5 Classe *Uvms*

No *framework* de intervenção, *Uvms* é a classe que encapsula as atividades que não estão diretamente relacionadas com o planejamento de movimento, mas interagem com ele. Esta é uma definição de classe do objeto *sistema* do pseudocódigo do Algoritmo 3.

A definição dessa classe dá certa liberdade de implementação do sistema, uma vez que o único método requerido para interação com os demais componentes do *framework* é o *execute*, que consome as referências de movimento calculadas por instâncias de *Guiagem* e produz em retorno um conjunto de atributos (instância de *AttributeList*), que no Algoritmo 5 é armazenado na variável *a*. Além dos valores atuais das variáveis das juntas do sistema (*qa*), esse conjunto de atributos deve conter informações relevantes para a verificação do contexto de operação por missão.

De acordo com o diagrama de blocos da Figura 33, essas informações provêm da atividade navegação. As referências resultantes da guiagem, por sua vez, são usadas para definir referência de atuação para os controladores pelas atividades pilotagem e acionamento, que gerarão o acionamento dos atuadores e propulsores do sistema de intervenção.

A Figura 36 ilustra o relacionamento dos componentes do *framework* de intervenção para o caso do UVMS planar analisado na Seção 3.3. Nessa representação do fluxo de informações, as variáveis *q* são instâncias de *State*, enquanto *a* é uma instância de *AttributeList*.

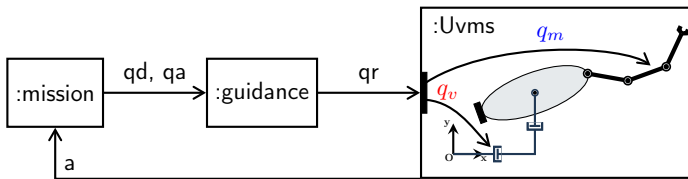


Figura 36. Fluxo de informações no sistema de intervenção planar

Embora não tenham sido discutidas neste trabalho, as acelerações são fundamentais para aplicação na dinâmica e no controle. Por esse motivo, os objetos *State* armazenam dados de aceleração nas variáveis das juntas. No caso das variáveis primárias, as acelerações devem ser fornecidas por *tarefa*. Isso pode ser definido na implementação dos interpoladores de trajetória que geram as referências da tarefa a partir das especificações do operador. Já para as variáveis secundárias, Santos (2006) propõe utilizar a derivada da Equação 3.33 para calcular as

acelerações. Outro modo de fazê-lo seria por derivação numérica.

Na implementação de classes derivadas de *Uvms*, pode-se modelar tanto a cinemática quanto a dinâmica do sistema de intervenção. Para a pilotagem, por exemplo, a dinâmica inversa é de grande importância. Ela é função das variáveis de posição, velocidade e aceleração, como descrito na Equação 6.2, em que \mathcal{D} é o símbolo que a representa (Featherstone, 2008).

$$\tau = \mathcal{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \quad (6.2)$$

A modelagem dinâmica é usualmente definida através dos formalismos de Newton-Euler ou Euler-Lagrange a partir da cinemática descrita pela notação de Denavit-Hartenberg (Fossen, 1994; Antonelli, 2006). Isso, porém, não deve gerar problemas de utilização das variáveis cinemáticas determinadas através da abordagem por helicoides, desde que sejam considerados os mesmos referenciais para ambos os modelos. Assim, uma vez definidas as posições, velocidades e acelerações do sistema, é possível aplicar estas na dinâmica e no controle. Embora não seja objeto de estudos desta tese, observa-se que uma modelagem dinâmica através da teoria dos helicoides seria de utilidade para uniformizar os modelos. Porém, este ainda é um problema em discussão na literatura, seja através dos formalismos de Newton-Euler, Euler-Lagrange ou do princípio de D'Alembert. A análise da dinâmica por helicoides em manipuladores seriais tem apresentado alguns resultados, que podem ser futuramente estendidos para o cenário da intervenção subaquática (Laus; Simoni; Martins, 2009, 2010).

6.4 SISTEMÁTICA DE DESENVOLVIMENTO DE ESTRATÉGIAS DE PLANEJAMENTO DE MOVIMENTO

Com base nos *frameworks* projetados, é possível sistematizar o desenvolvimento de estratégias de planejamento de movimento. Essa sistematização consiste em definir classes derivadas do *framework* de intervenção, cujos métodos implementarão as características necessárias para que os pseudocódigos dos Algoritmos 4 e 5 sejam executáveis. Além disso, podem ser usadas instâncias de classes do *framework* *Kast* ou mesmo especializações dessas classes.

Assim, as etapas de sistematização são definidas e descritas como segue. Algumas dessas etapas podem ainda ser divididas em atividades menores para melhor descrever as ações a serem executadas.

Definir o modelo cinemático Assumindo um contexto inicial de execução da tarefa, deve-se definir um arranjo de cadeia cinemática fechada que represente o sistema de intervenção subaquática (veículos, manipuladores) e o espaço de trabalho em que vai atuar (objetos a manipular, movimentos/restrições desejados). A componentização é uma ferramenta que pode agilizar esse processo, bem como a existência de um banco de cadeias previamente modeladas. As atividades dessa etapa seriam:

- definir o grafo de movimento da cadeia cinemática;
- identificar variáveis primárias e secundárias;
- definir as expressões para os heligiros;
- implementar o modelo cinemático.

A implementação do modelo cinemático pode ser feita através da classe `KCComposable` utilizando uma descrição textual para definir sua instância. Outra possibilidade é especializar uma classe derivada de `KinematicChain`. Pontos a se considerar para a escolha do modo de implementar são a flexibilidade da solução adotada e seu desempenho.

Definir a tarefa A identificação das partições da equação de restrição da etapa anterior é utilizada na definição da tarefa. Os movimentos definidos pelo usuário e restrições do espaço de trabalho devem ter correspondência nas variáveis primárias da cadeia cinemática. Além disso, devem ser identificados quais seriam os geradores de trajetória adequados aos movimentos/restrições. Assim, tem-se como atividades desta etapa:

- associar as variáveis primárias à definição da tarefa;
- definir os geradores de trajetórias para as variáveis;
- implementar o classe derivada de `Task`.

A classe derivada de `Task` deve conter geradores de trajetória e mapeá-los para cada variável primárias do objeto `State` vinculado à cadeia cinemática. A cada iteração da execução da intervenção, as referidas variáveis são preenchidas com os valores produzidos por esses geradores.

Definir o UVMS Essa etapa consiste na implementação do modelo do sistema de intervenção (em simulações) ou de se definir interfaces de acesso com os sistemas (em implementações reais). O

modelo pode incluir aspectos cinemáticos, dinâmicos e de controle, que foram os considerados externos aos problema de planejamento de movimento neste trabalho. Na implementação do método `execute`, deve-se mapear as variáveis da referência gerada pela guiagem (`qr`) para as respectivas entradas nas atividades de pilotagem, acionamento e controle. Além disso, após processar essas referências deve preencher a lista de atributos que será retornada com os valores atuais das variáveis de juntas do sistema (`qa`) e demais dados obtidos de sensores, o que é feito pela atividade de navegação.

Definir a missão A implementação de uma classe para gerenciar a missão envolve diferentes demandas. Além de ter a tarefa e a cadeia cinemática como seus componentes, uma classe de missão deve ter ações para a identificação do estado de execução da tarefa e modificação da cadeia e da tarefa para atender à mudanças desse estado. Assim, são relacionadas as atividades:

- Definir uma estratégia de identificação de contexto/estado de execução;
- Definir a execução de modificações no modelo cinemático de acordo com o estado;
- Definir a execução de modificações na tarefa de acordo com o estado;

Essas ações crescem em complexidade à medida em que aumenta a quantidade de dados a serem avaliados e as diferentes possibilidades de contexto de execução. Além disso, certas condições podem ser cumulativas, exigindo mudanças mais complexas do cenário.

Definir a intervenção Esta classe é de relativa simplicidade na implementação, uma vez que o método `execute`, que controla a execução da tarefa, usualmente é uma implementação do Algoritmo 5, variando pouco de acordo com as especificades do contexto de execução. Os demais métodos relevantes são `init` e `end`. O primeiro deve instanciar os objetos utilizados por `execute`, sendo que são necessários ainda instanciar anteriormente a cadeia cinemática e a tarefa para associá-los à missão e à guiagem, bem como os objetos `State` que são utilizados no fluxo de dados entre os diferentes objetos. A implementação de `end`, se necessária, deve apenas encerrar o ciclo de vida dos objetos e se for o caso passar a execução da missão para outros componentes do sistema.

Ao final desse processo, deve-se obter uma implementação compatível com a descrição da execução de tarefas de intervenção apresentada nos Algoritmos 2 e 3. Apesar de parecer complexo, observa-se que várias simplificações podem ser feitas através de reuso de resultados de sistematizações prévias, como na obtenção dos modelos cinemáticos. Além disso, implementações existentes de classes dos *frameworks*, como os geradores de trajetória e a *KCComposable* facilitam o processo. Outro recurso a ser explorado é o uso de descrições textuais.

A implementação de estratégias de planejamento de movimento de sistemas de intervenção subaquática a partir dos resultados da análise cinemática anteriormente realizada foi o objetivo deste capítulo. A sistematização proposta foi estruturada a partir de um conjunto de classes de objetos que representam os blocos de atividades identificadas como relevantes para o planejamento de movimento. Esse conjunto faz uso do *framework* *Kast* para modelar cinematicamente o sistema robótico e as tarefas a ele especificadas. Essa sistemática é usada em simulações de diferentes cenários de intervenção subaquática no próximo capítulo, a fim de ilustrar o seu uso.

7 SIMULAÇÕES E ANÁLISE

Este capítulo trata da implementação de estratégias de planejamento de movimento em simulações. São apresentados diferentes cenários de intervenção. Para cada sistema, são propostas tarefas que demandarão diferentes soluções da equação de restrição. Certos casos exigirão a mudança de estado de execução durante a missão, fazendo com que o sistema assuma diferentes configurações e soluções para a cinemática do sistema. Com isso, espera-se ilustrar os conceitos discutidos nos capítulos anteriores e a proposta de uma sistematização para a definição de planejamento de movimento.

Os modelos cinemáticos dos cenários são detalhados no Apêndice C. As simulações foram implementadas em Python (Python.org, 2011), com uso da implementação do *framework* Kast apresentada no Apêndice B. A visualização utilizou gráficos gerados pelo módulo matplotlib do Python (Hunter; Dale; Droettboom, 2011) e cenários virtuais gerados pelo software OpenRAVE (Diankov, 2010).

7.1 UVMS COM UM MANIPULADOR

O primeiro cenário de intervenção consiste em um UVMS composto por um veículo com um manipulador operando no plano horizontal, ou seja, mantendo uma profundidade constante. O manipulador é planar, com três graus de liberdade definidos por juntas rotativas. Este cenário é baseado em exemplos descritos em (Antonelli, 2006; Santos, 2006) e seu modelo é descrito na Seção C.3.1. Uma representação desse cenário é apresentada na Figura 37.

A cadeia cinemática desse modelo e a definição dos heligiros das suas juntas é descrita em um arquivo XML que possibilitará a um componente KCFactory instanciar um objeto KCComposable que o represente nas simulações.

7.1.1 Execução de uma trajetória retilínea

Esta simulação consiste na execução de uma trajetória retilínea repetitiva pelo efetuador final do UVMS, como se fosse o caminho percorrido por uma ferramenta durante uma operação de limpeza de casco de uma embarcação, por exemplo. O efetuador final se desloca de um

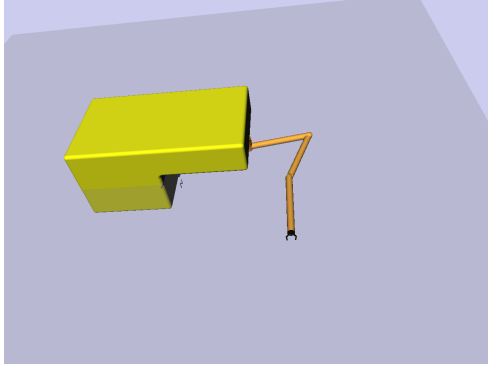


Figura 37. Cenário de simulação de um UVMS com um manipulador

ponto $\boldsymbol{\eta}_{e,1}$ a um ponto $\boldsymbol{\eta}_{e,2}$ seguindo uma trajetória retilínea paralela ao eixo \mathbf{x} do referencial inercial no tempo de $8s$, retornando após ao ponto $\boldsymbol{\eta}_{e,1}$ no mesmo período de tempo. Esse movimento é executado duas vezes, o que faz com que a tarefa leve um tempo de $32s$ para ser completada. Durante toda a operação, o efetuador final mantém a orientação constante. As posturas do efetuador no plano $\boldsymbol{\eta}_e = [x_e \ y_e \ \psi_e]^T$ são iguais a

$$\boldsymbol{\eta}_{e,1} = \begin{bmatrix} 3,935 \\ 3,682 \\ 1,571 \end{bmatrix} \quad \boldsymbol{\eta}_{e,2} = \begin{bmatrix} 6,135 \\ 3,682 \\ 1,571 \end{bmatrix} \quad (7.1)$$

onde as unidades de comprimento x_e e y_e estão em metros (m) e a unidade de ângulo ψ_e está em radianos¹.

Durante a execução da tarefa pelo efetuador final, o veículo permanece estacionário, alinhado com o referencial inercial e com a posição do referencial c do UVMS mostrado na Figura 89 coincidente com a origem do referencial inercial, ou seja, $\boldsymbol{\eta}_{v,1} = \boldsymbol{\eta}_{v,2} = [0 \ 0 \ 0]^T$.

As referências para as variáveis de junta da cadeia cinemática do sistema de intervenção q_{t_1} , q_{t_2} e q_{t_3} associadas respectivamente às componentes da postura do efetuador final x_e , y_e e ψ_e são geradas por interpolação polinomial de 5ª ordem (uma instância de TrajLinearPoly) de forma que pare ao final de cada trecho percorrido entre os dois pontos extremos da reta. As referências para as variáveis de juntas q_{v_1} , q_{v_2} e

¹Estas unidades serão as utilizadas em todas as simulações, salvo outra unidade ser explicitamente descrita.

q_{v_3} associadas à postura do veículo (x_v , y_v e ψ_v , respectivamente) são valores constantes, com velocidade nula (geradas por uma instância de `TrajConst`). Essas variáveis são definidas como primárias.

A implementação da simulação é descrita na Listagem 8 do Apêndice B. As variáveis a serem determinadas são as associadas com as juntas do manipulador q_{m_1} , q_{m_2} e q_{m_3} . Isso define uma matriz \mathbf{N}_s quadrada. O passo da simulação é 0,01s. As velocidades são integradas pelo método de Euler (usado como padrão pela classe `Guidance` após obter as velocidades pela solução da equação de restrição). Para minimizar os erros relativos à integração numérica e precisão computacional, utilizou-se uma adaptação da técnica de cinemática inversa em malha fechada (CLIK, ou *Closed-Loop Inverse Kinematics*) para as variáveis da tarefa, que é implementada na classe `Guidance`. Segundo ela, as velocidades das variáveis primárias são definidas como

$$\dot{\mathbf{q}}_p = \dot{\mathbf{q}}_d + \mathbf{K}_p \mathbf{e}_d \quad (7.2)$$

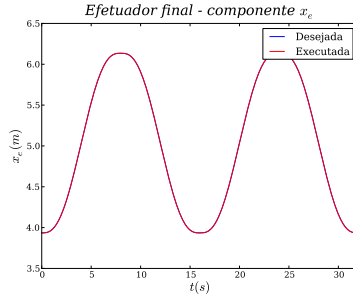
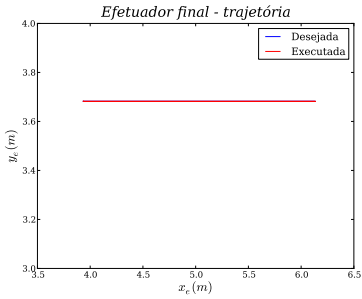
onde $\dot{\mathbf{q}}_d$ é o valor das velocidades desejadas para as variáveis primárias e \mathbf{e}_d é o erro de postura, definido como a diferença entre a postura desejada para o efetuador final e a postura definida pela cinemática direta do sistema em função das posições das juntas do manipulador e da postura do veículo (definida em uma classe derivada de `Uvms` criada para representar o sistema de intervenção), ou

$$\mathbf{e}_d = \mathbf{q}_d - \mathbf{q}_a \quad (7.3)$$

A matriz de ganhos \mathbf{K}_p foi determinada por tentativa e erro, tendo como ponto de partida os valores utilizados em (Antonelli, 2006). Chegou-se aos valores iguais a $\mathbf{K}_p = \text{diag} \{100, 100, 10\}$. Embora aqui apenas tenha sido utilizada a realimentação do erro de posição para o CLIK, pode-se estender a técnica para incluir a realimentação do erro de velocidade e a integração do erro. Essas extensões no *framework* serão implementadas em futuras versões de `Kast`.

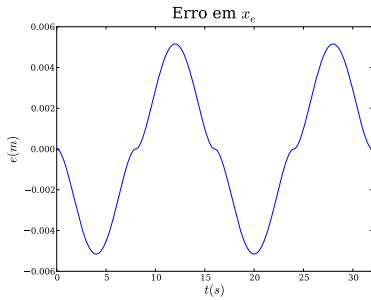
A trajetória percorrida pelo efetuador final é apresentada na Figura 38a. A variação do componente x_e ao longo do tempo é traçada na Figura 38b. As componentes y_e e ψ_e , por sua vez, não variaram seus valores. Os erros nas variáveis do efetuador devido ao uso de métodos numéricos e precisão computacional são traçados na Figura 39.

Uma composição das posturas do sistema de intervenção em diferentes instantes da execução da tarefa é mostrada na Figura 40. As posições das juntas e suas velocidades são apresentadas na Figura 41.

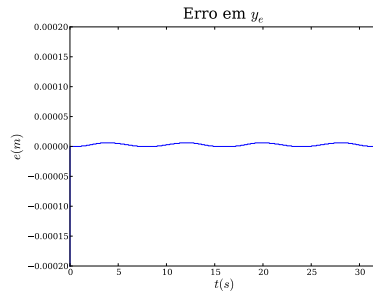


(a) Trajetória do efetuador final do UVMS (b) Componente x_e do efetuador final

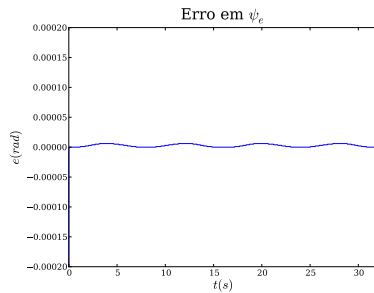
Figura 38. Movimento do efetuador final



(a)



(b)



(c)

Figura 39. Erro nas variáveis do efetuador final

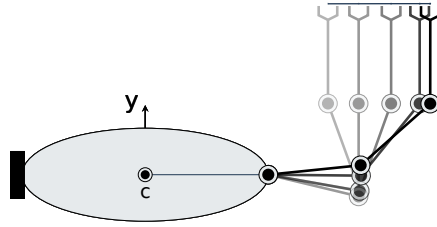


Figura 40. Movimento do UVMS durante a execução da tarefa

Esse primeiro caso de simulação foi utilizado para validar a implementação dos *frameworks* Kast e de intervenção. Além disso, por ser um caso simples permite demonstrar como utilizar os *frameworks*.

Com o mesmo objetivo de validação, uma segunda simulação da mesma tarefa foi executada. Nela, foi considerado que o veículo também pode se mover, o que acarreta um particionamento diferente do original. Nesse caso, o sistema torna-se redundante, com seis variáveis a serem determinadas para um movimento planar. O método *SolveDavies* de *KinematicChain* passa a utilizar o operador de pseudoinversão para essas situações. O reparticionamento foi definido pela atribuição da nova relação de variáveis primárias e secundárias no atributo *partitioning* da cadeia cinemática.

A solução da equação de restrição com o uso da pseudoinversa distribui o movimento da mesma forma para todas as variáveis secundárias. Isso resulta em movimentação do veículo em todos os momentos da execução da tarefa, o que é indesejável devido à dinâmica do veículo ser bastante diferente da do manipulador. Nesse caso, a pseudoinversão ponderada pode minimizar o movimento do veículo, pela distribuição de pesos para cada variável a ser calculada. Essa operação é definida na Equação 6.1.

A classe *KinematicChain* tem um atributo denominado *plnv* que pode ser utilizado para definir um objeto que execute a operação de pseudoinversão em matrizes. Uma classe de pseudoinversão ponderada denominada *WPInV* está disponível em *Kast*. Nela, pesos grandes implicam em menores valores para as variáveis associadas. Uma terceira simulação utilizou essa classe, com pesos unitários para as variáveis das juntas do manipulador e pesos de grande magnitude para as variáveis do veículo. Essa distribuição produziu resultados próximos aos da primeira simulação.

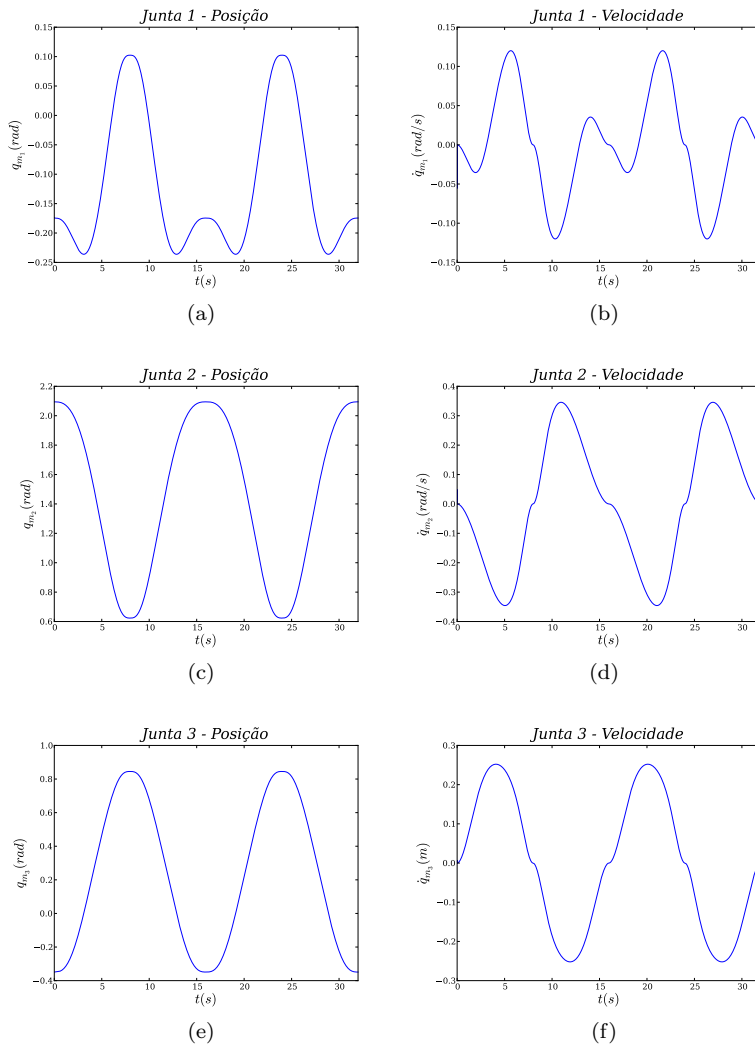


Figura 41. Variáveis do manipulador

Nas simulações em que as variáveis do veículo são consideradas secundárias, o movimento do efetuador final é o mesmo apresentado na Figura 38. O comportamento das juntas do manipulador é mostrado na Figura 42. O movimento do veículo é apresentado na Figura 43.

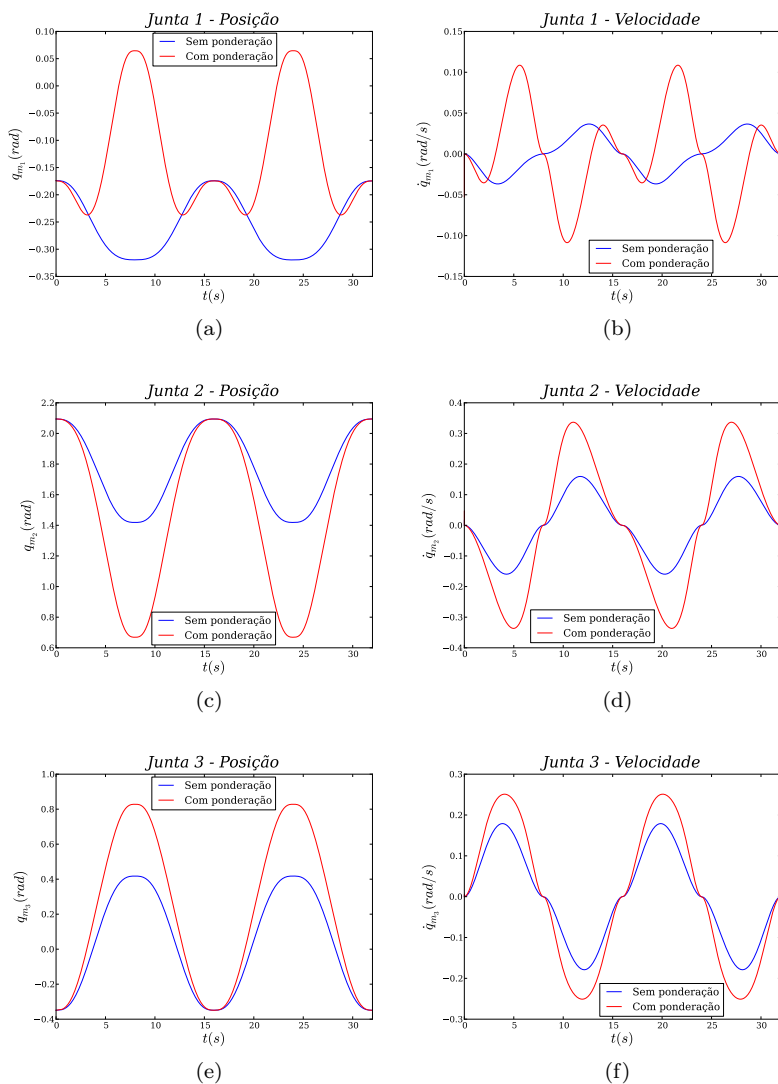


Figura 42. Variáveis do manipulador

Essas duas simulações ilustram a reusabilidade dos componentes dos *frameworks*, pois novos arranjos foram produzidos com poucas modificações de atributos.

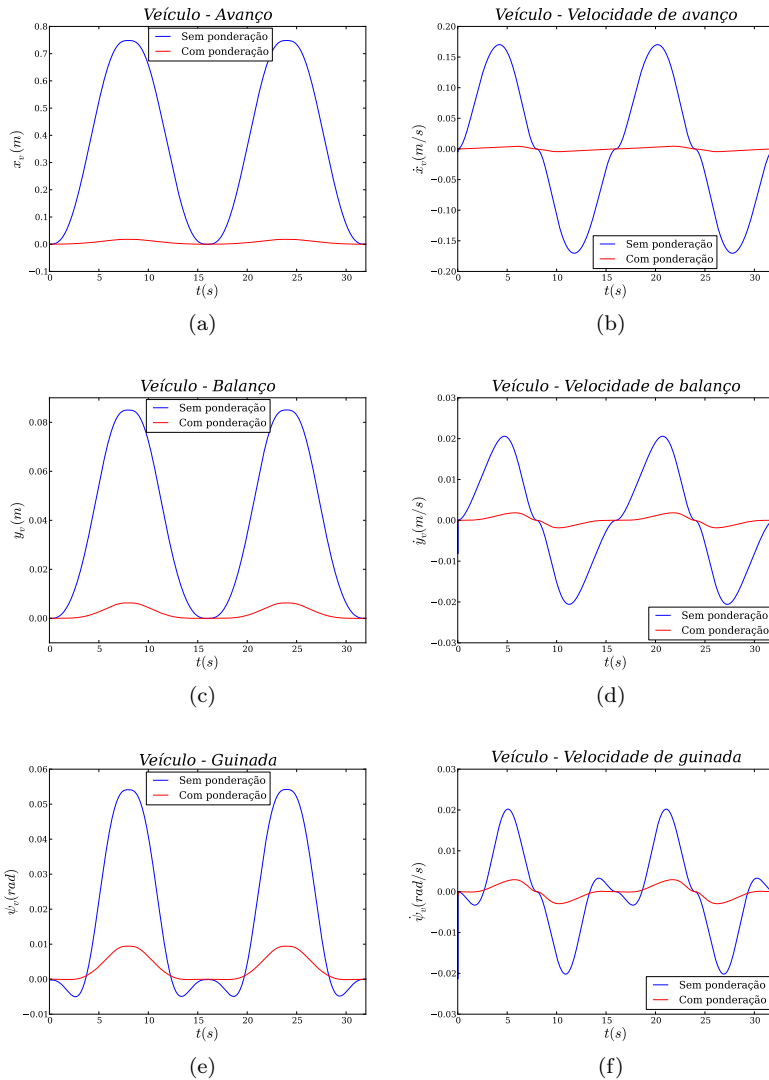


Figura 43. Variáveis do veículo

7.1.2 Reorientando o veículo durante a execução da tarefa

O caso aqui simulado consiste na reorientação do veículo enquanto o UVMS executa a mesma tarefa do primeiro caso. Esse tipo

de movimento do veículo pode ser interessante para alinhá-lo com a correnteza e assim diminuir o arrasto por ela provocado (Antonelli, 2006).

Inicialmente assume-se que o veículo deve girar 30° em um intervalo de $10s$ enquanto se mantém na posição inicial. O particionamento é o mesmo do primeiro caso simulado, uma vez que o movimento/restrrição do veículo é imposto. Assim, apenas as variáveis do manipulador compõem a partição secundária da equação de restrição. O movimento resultante nas juntas do manipulador é apresentado na Figura 44.

Quando a reorientação é feita com ângulos maiores para um mesmo intervalo de tempo, o manipulador passa por uma configuração singular ($q_{m_2} = 0$). Uma solução para esse problema é a adoção de um particionamento onde o veículo também faça parte da partição secundária e tenha seu movimento calculado, utilizando pseudoinversão ponderada na solução da equação de restrição para minimizar o movimento do veículo.

Outra possibilidade para a solução desse problema é o uso de uma abordagem por estados de operação (Santos, 2006). Nesse caso, pode-se estabelecer dois estados, um onde o veículo fica estacionário e outro onde ele pode se movimentar. No primeiro, o particionamento é o adotado na primeira simulação deste cenário. No segundo, as variáveis do veículo fazem parte da partição secundária. O evento que leva do estado estacionário para o móvel é a proximidade de uma singularidade (para o manipulador planar, q_{m_2} próximo de 0). No estado móvel, o evento que o leva ao estado estacionário é a variável q_{m_2} ter um valor que afaste o manipulador da singularidade.

As duas soluções acima listadas foram simuladas. O veículo sofre uma reorientação de $0,78rad$ no período de $10s$ enquanto deve executar a tarefa. Na primeira simulação, usando apenas pseudoinversão ponderada, as variáveis q_{v_1} e q_{v_2} juntamente com as variáveis do manipulador formam a partição secundária. Os pesos foram definidos como iguais a 100 para as variáveis do veículo e 1 para as do manipulador.

A segunda simulação utiliza um modelo híbrido de estados. Uma nova classe derivada de Mission foi criada para definir o método `verifyState`, responsável por verificar o estado de operação e, se necessário, modificar a cadeia cinemática e seu particionamento. Dois estados foram definidos, um estacionário e um móvel. A simulação inicia no estado estacionário, para o qual apenas as variáveis do manipulador devem ser determinadas. Quando o manipulador se aproxima de uma singularidade (q_{m_2} menor que um limite mínimo), `verifyState` modifica o estado para móvel. Nesse caso, é feito um reparticionamento da equa-

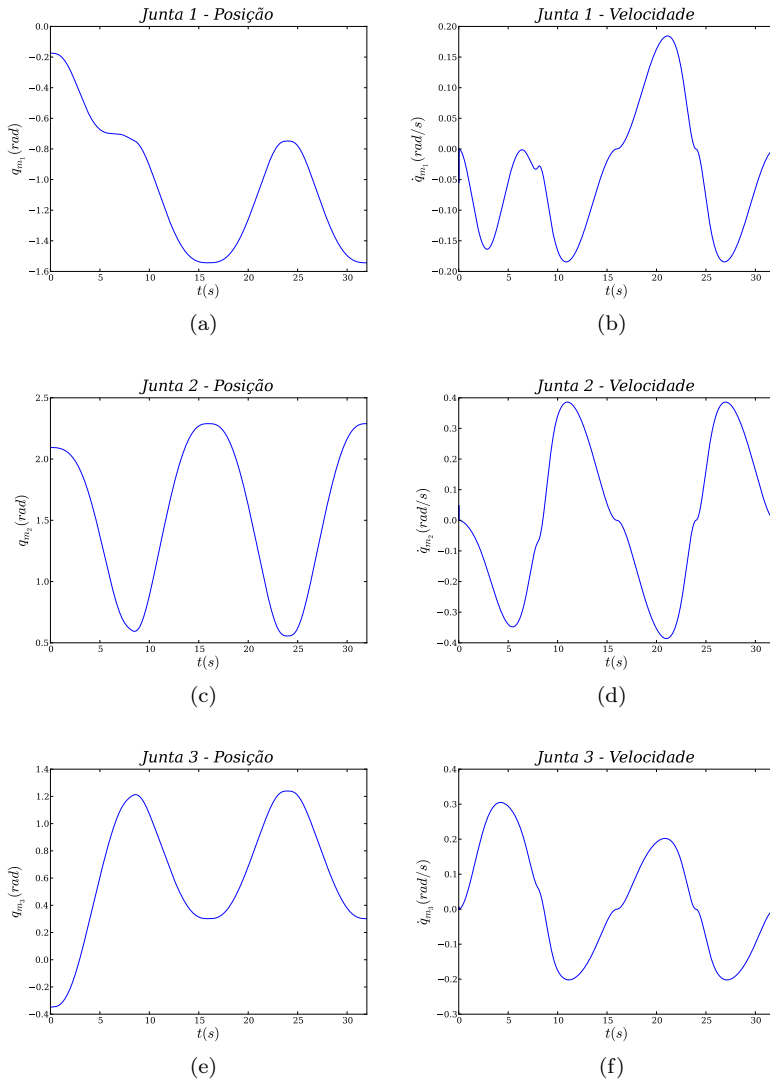


Figura 44. Variáveis do manipulador

ção de restrição, onde as variáveis q_{v_1} , q_{v_2} , q_{m_1} e q_{m_3} são consideradas secundárias. q_{m_2} passa a ser primária, e um gerador de trajetória polinomial é atribuído a ela para que seu valor passe do atual para um

valor maior que um limite máximo de segurança em um período de 1s. O sistema retorna ao estado móvel quando `verifyState` verifica que q_{m_2} tem um valor que o afaste da singularidade. Ao retornar ao estado estacionário, volta-se ao particionamento original e o gerador de trajetória associado às variáveis do manipulador são novamente utilizados, agora com as componentes de posição atualizadas. Ocorre uma transição de estacionário para móvel aos 7s da simulação, com o sistema retornando ao estado estacionário aos 7,75s.

Nas Figuras 45 e 46 são mostradas as variáveis do manipulador e do veículo, respectivamente. As duas simulações são apresentadas juntas para comparação de resultados. Observa-se que os valores de guinada são impostos, e por isso são iguais nos dois casos.

Nelas, pode-se observar que ao se usar apenas a pseudoinversa, a junta q_{m_2} do manipulador chega a atravessar a singularidade, como é destacado na Figura 45c, devido à problemas de precisão numérica. Porém, o veículo movimenta-se constantemente. O resultado produzido pelo modelo híbrido de estados restringiu o movimento do veículo a um curto período de tempo, apesar de ter algumas descontinuidades nas transições de estados. Esta última solução é mais interessante, porém, sob a ótica de minimizar o movimento do veículo.

Verifica-se nas duas simulações a flexibilidade, extensibilidade e reusabilidade dos *frameworks*. O simulação com modelo híbrido, em especial, foi facilmente implementada por reparticionamento da cadeia cinemática e vinculação de geradores de trajetória à tarefa.

7.1.3 Execução da Tarefa na Presença de Falha

Uma outra variação da primeira simulação que pode ser tratada por um modelo híbrido de estados é a ocorrência de uma falha que limite a atuação do sistema. No caso, tem-se um travamento de uma junta do manipulador do UVMS durante a execução de uma tarefa. A falha é indicada por uma variável de estado que uma implementação de classe derivada de *Uvms* lança em um determinado instante de tempo. Nesta simulação, a falha ocorre pelo travamento da junta q_{m_2} aos 19s.

Quando a instância de missão detecta a falha, o sistema passa do estado inicial estacionário para o estado móvel, o que implica no reparticionamento da equação de restrição colocando a junta q_{m_2} como primária (com uma referência constante referente à posição em que travou) e as juntas da cadeia do veículo como secundárias. O sistema passa a ter redundância, e uma instância de operador de pseudoinversão

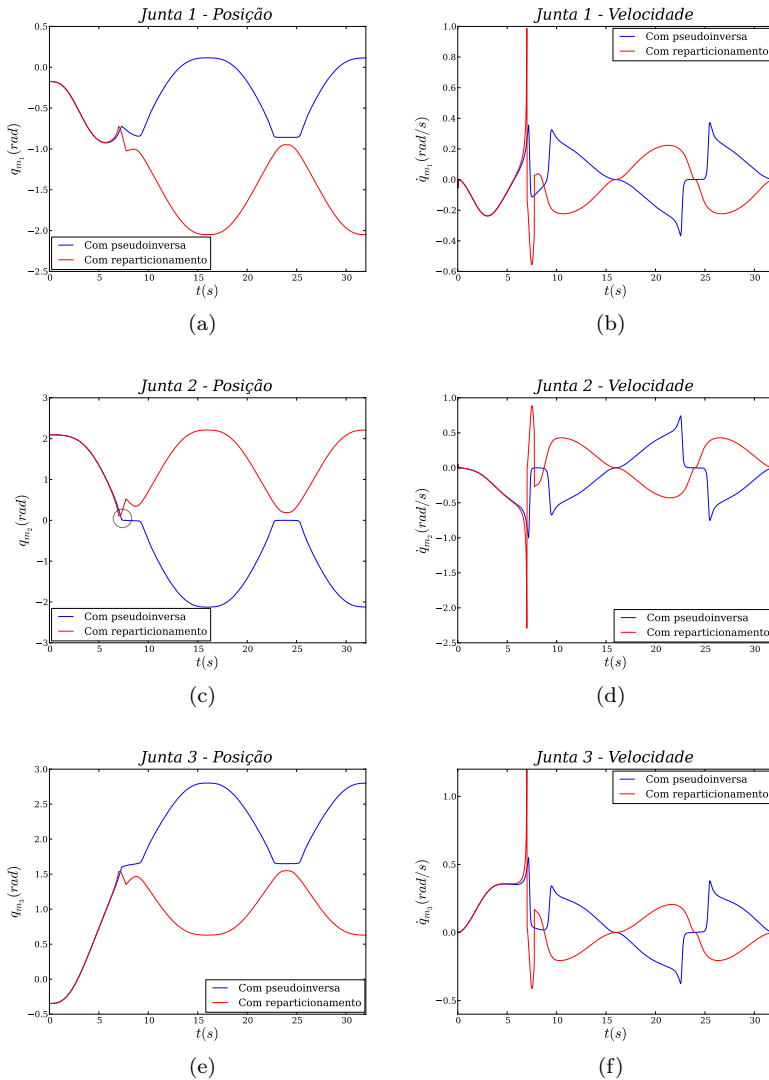


Figura 45. Variáveis do manipulador

ponderada de matrizes é alocada à cadeia cinemática. Pelo processo de tentativa e erro, chega-se a um conjunto de pesos que faça o veículo ter o menor movimento na direção de balanço (y), algum movimento na

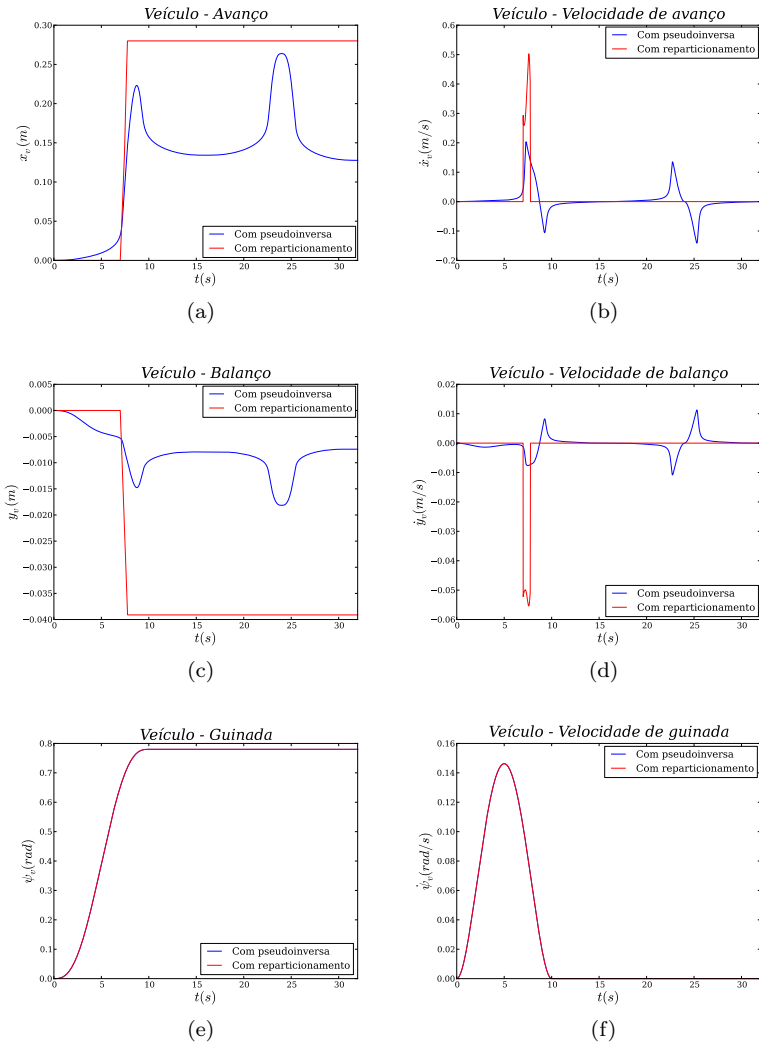


Figura 46. Variáveis do veículo

direção de avanço (\mathbf{x}) e o movimento de guinada (rotação em torno do eixo \mathbf{z}) seja equivalente ao das juntas do manipulador q_{m_1} e q_{m_3} .

O movimento do manipulador e do veículo nessa simulação são apresentados nas Figuras 47 e 48 respectivamente.

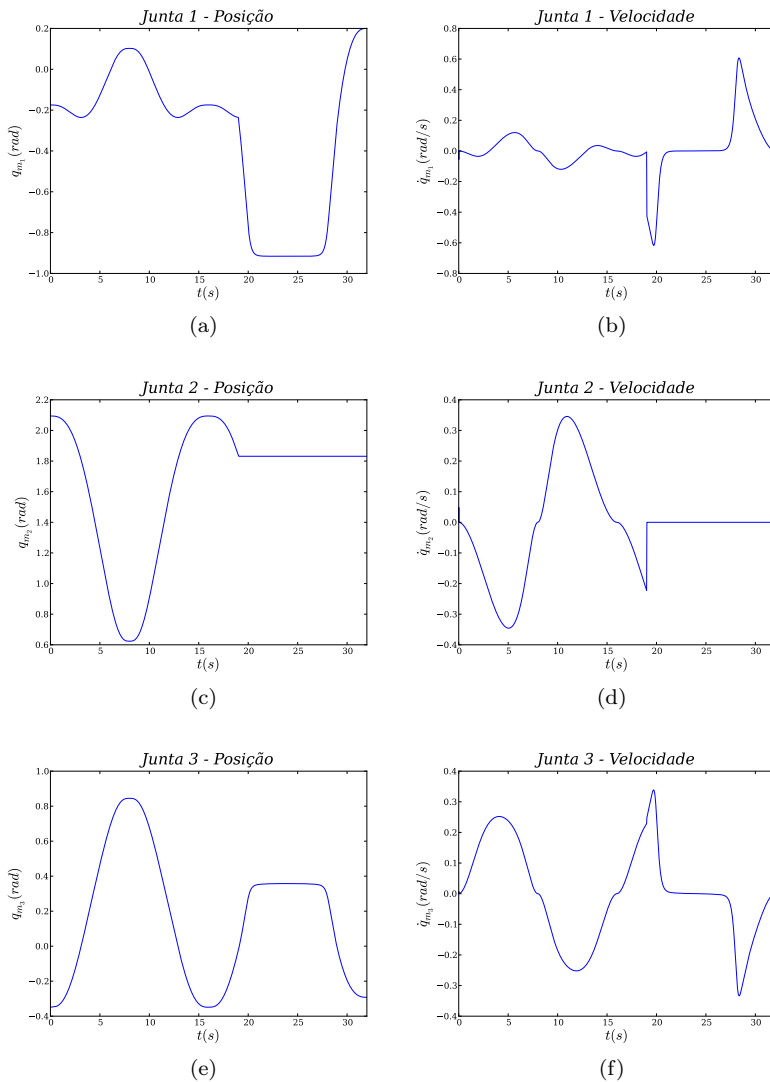


Figura 47. Variáveis do manipulador

A descontinuidade nesse caso é devida ao travamento da junta, quando q_{m_2} passa a ter um valor fixo. Apesar disso, observa-se que a tarefa consegue ser concluída, ao custo da movimentação do veículo.

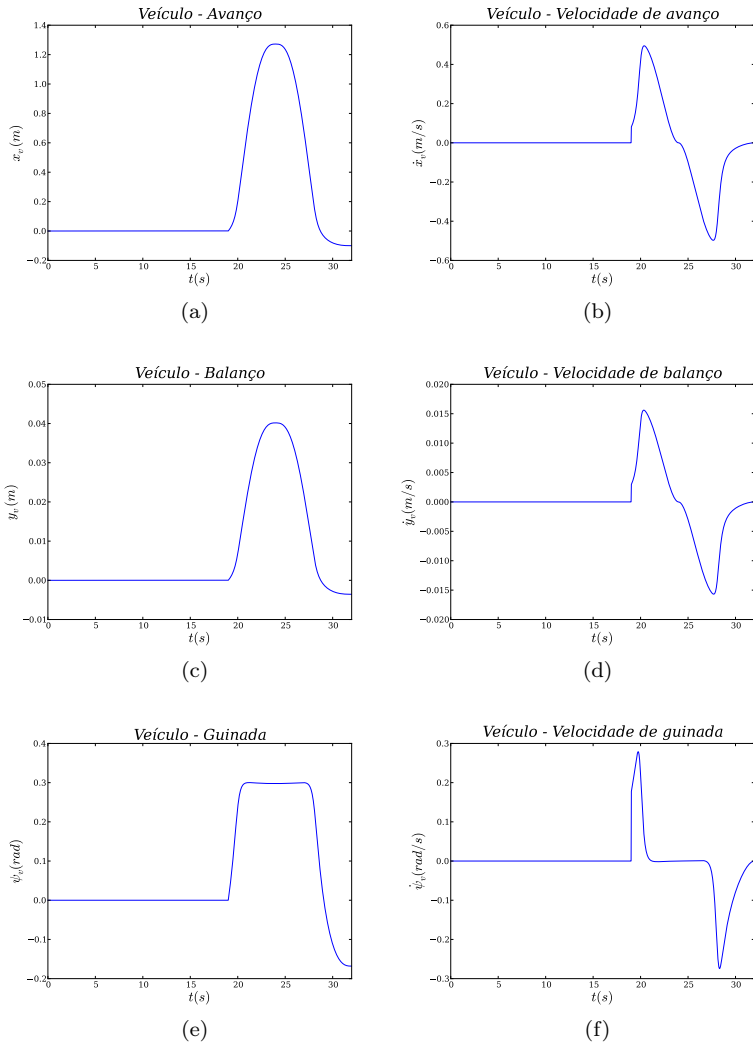


Figura 48. Variáveis do veículo

7.1.4 Execução de uma trajetória fechada

Uma última simulação utilizando o sistema de UVMS com um manipulador consiste no seguimento de uma trajetória fechada cujas

dimensões obrigarão o veículo a se mover. A trajetória é mostrada na Figura 49. Essa trajetória é percorrida em 46s. Durante todo o tempo, a orientação do efetuador final se mantém constante, com $q_{t_3} = 1,571$.

Para essa simulação, implementou-se novamente um modelo híbrido de estados cuja mudança é determinada pela proximidade ou afastamento da singularidade do manipulador (monitorando o valor da variável de junta q_{m_2}). Uma classe derivada de `Mission` foi criada para definir o método `verifyState`. O estado inicial é o estacionário, onde apenas as variáveis do manipulador estão na partição secundária. Quando o evento de proximidade de singularidade é detectado, o estado passa para móvel, onde as juntas do veículo e as juntas 1 e 3 do manipulador são da partição secundária e as demais da partição primária.

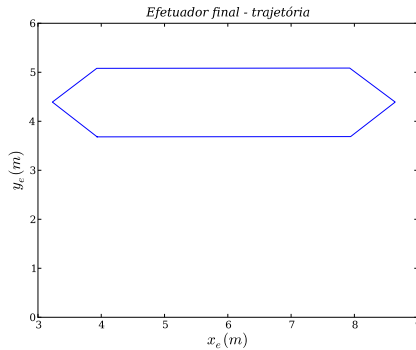


Figura 49. Trajetória a ser percorrida na execução da tarefa

Atribui-se um gerador de trajetória polinomial para q_{m_2} , de forma que seu valor passe do atual para o limite de segurança em um período de 3s. Ao passar desse limite, o sistema retorna ao estado estacionário, onde o particionamento original é restaurado e os geradores de trajetória das variáveis do veículo são atualizados.

O sistema fica no estado móvel em dois períodos de tempo. O primeiro inicia aos 5,55s e termina aos 7,98s. O segundo inicia aos 10,67s e termina aos 13,1s. Os resultados da simulação são mostrados nas Figuras 50 e 51.

A implementação dessa simulação tem os elementos discutidos nos capítulos anteriores sobre a questão da modificabilidade da cadeia cinemática e as implicações para as demais atividades que compõem um sistema de intervenção. Para cada variável que é definida como primária torna-se necessário ter um fornecedor de referências de posi-

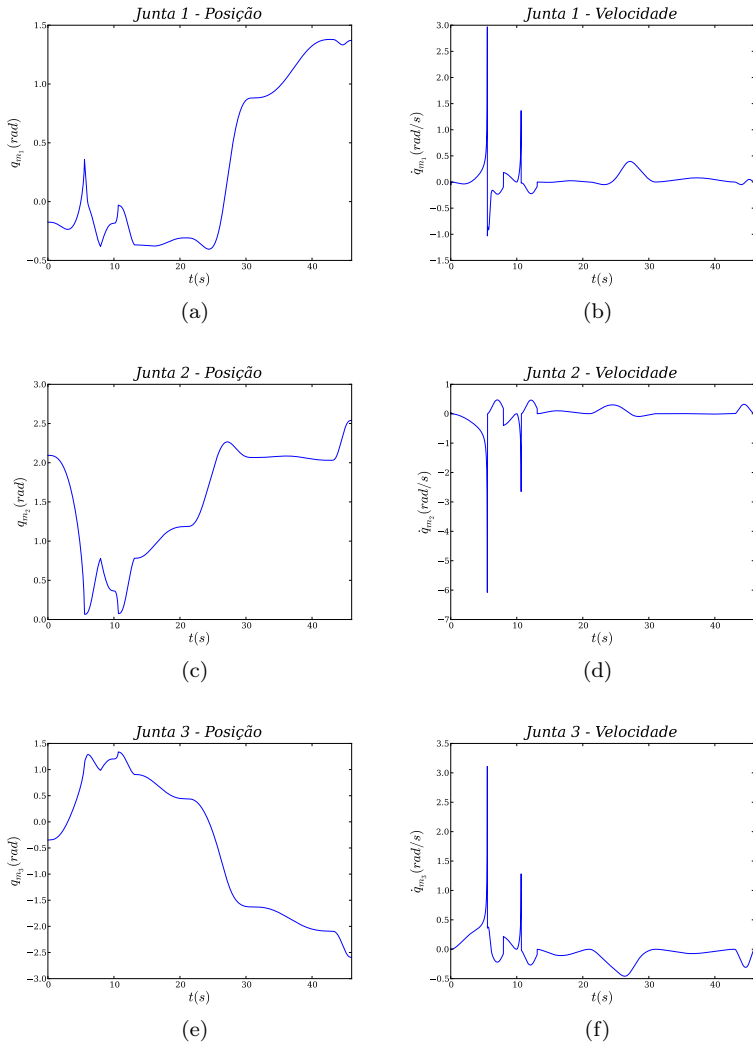


Figura 50. Variáveis do manipulador

ção e velocidade correspondente. A mudança de estados traz muitas vezes descontinuidades para o movimento das juntas, particularmente aquelas que passam de secundárias a primárias, devido à dificuldades para efetuar a transição suave entre duas referências de trajetória. Isso

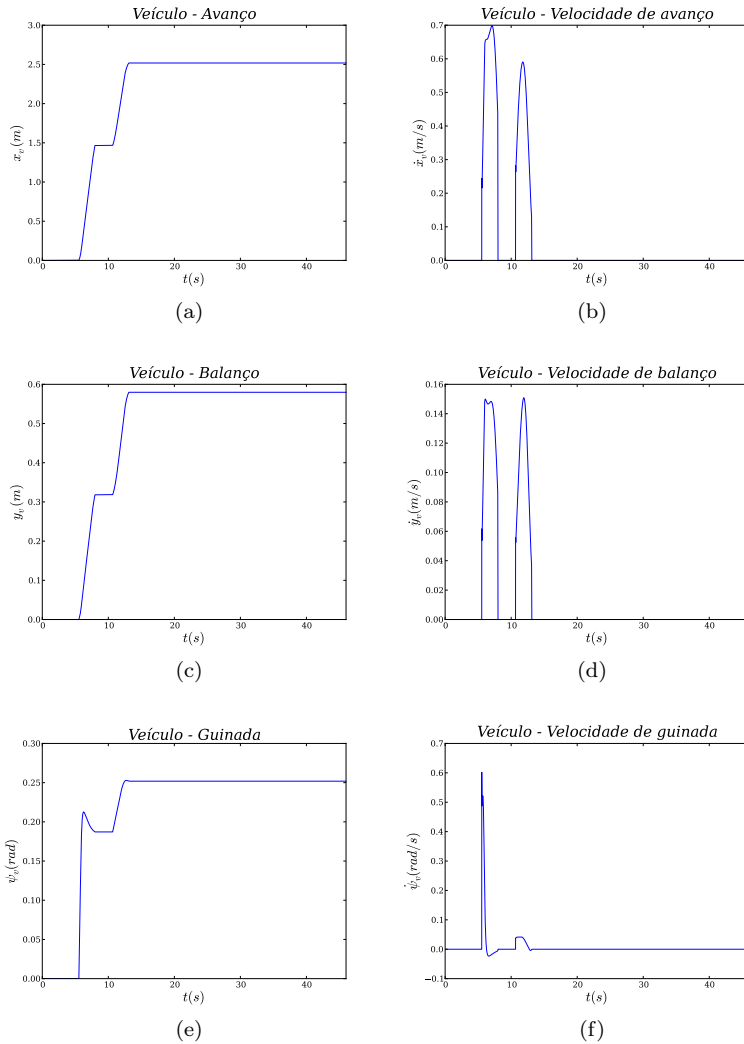


Figura 51. Variáveis do veículo

é evidente nesta simulação a cada mudança de estado do sistema de intervenção. A sistematização para o planejamento de movimento, porém, prova-se efetiva para a resolução das tarefas.

7.2 UVMS COM DOIS MANIPULADORES EM COOPERAÇÃO

O caso tratado neste cenário é de um UVMS composto por dois manipuladores planares de três graus de liberdade que operam em cooperação. A estrutura dos manipuladores é a mesma do cenário anterior. O caso de dois manipuladores cooperando em um UVMS foi analisado na extensão da cinemática por helicoides do Capítulo 4, e o modelo cinemático correspondente é apresentado na Seção C.3.2. O cenário é ilustrado na Figura 52. Como no cenário anterior, o modelo cinemático desse sistema de intervenção é descrito em um arquivo XML, que é usado para instanciação de objetos `KCComposable` que os represente nas simulações.

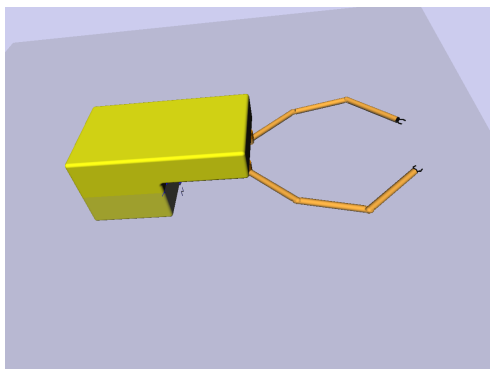


Figura 52. Cenário de simulação de um UVMS com dois manipuladores em cooperação

7.2.1 Posicionamento e encaixe de uma peça

Nesta simulação, a tarefa consiste no posicionamento de uma peça retangular e posterior encaixe em um painel. Para tanto, define-se como a tarefa em questão o movimento do centro da peça. A posição inicial desse ponto é coincidente com a origem do sistema de coordenadas inercial, e os eixos de seu referencial estão alinhados com ele, assim $\eta_{p,t=0} = [0 \ 0 \ 0]^T$.

A trajetória dessa peça é composta por diferentes movimentos. Inicialmente, ela se desloca $0,75m$ na direção de 30° em relação ao eixo x do referencial inercial em um período de $10s$. Após, ela se

desloca $1,65m$ horizontalmente em um tempo de $15s$. Nessa posição, ela rotaciona 15° em relação ao eixo x do referencial inercial em $5s$ e por fim, se desloca $0,3m$ nessa direção em um tempo de $5s$. A tarefa, então, tem a duração total de $35s$.

A peça é movimentada pelos dois manipuladores vinculados ao UVMS. Os seus efetuadores finais agarram a peça por alças de transporte, mantendo posição e orientação fixas em relação ao centro da peça. Seus valores são iguais a $\mathbf{q}_{r_1} = [-0,326 \quad -1,011 \quad 0]^T$ e $\mathbf{q}_{r_2} = [-0,326 \quad 1,011 \quad 0]^T$, onde o subscrito r_i denota a posição relativa do efetuador i em relação ao centro da peça. Durante a execução da tarefa, o veículo mantém uma postura fixa igual a $\boldsymbol{\eta}_v = [-6,3 \quad 0 \quad 0]^T$.

As variáveis de juntas da cadeia cinemática do sistema de intervenção q_{t_1} , q_{t_2} e q_{t_3} são associadas às componentes da postura da peça $\boldsymbol{\eta}_p$. A postura do veículo é representada pelas variáveis de juntas q_{v_1} , q_{v_2} e q_{v_3} . As posições relativas entre os efetuadores finais e o centro da peça são representadas por duas cadeias virtuais cujas variáveis são $q_{r_{i,1}}$, $q_{r_{i,2}}$ e $q_{r_{i,3}}$, onde $i = 1, 2$ denota o efetuador final. Este conjunto de variáveis forma a partição primária da equação de restrição.

As referências para as variáveis da peça são fornecidas por geradores de trajetória compostos (instâncias de `TrajSequence`) que agregam geradores de trajetórias lineares (`TrajLinearPoly`) e de valores constantes (`TrajConst`) para a posição e a orientação da peça. Para as demais variáveis, que devem ter um valor constante, são utilizados geradores de trajetória instanciados de `TrajConst`.

A implementação da simulação é similar à feita para o cenário anterior. As variáveis a serem determinadas são as das juntas dos dois manipuladores: $q_{m_{1,1}}$, $q_{m_{1,2}}$, $q_{m_{1,3}}$, $q_{m_{2,1}}$, $q_{m_{2,2}}$ e $q_{m_{2,3}}$. Novamente, a matriz \mathbf{N}_s é quadrada. O passo da simulação é $0,01$, e o método de integração das velocidades é o de Euler. Foi utilizado o modo de cinemática inversa de malha fechada do componente `Guidance`. As variáveis consideradas nessa realimentação são as das cadeias r_1 e r_2 , que podem ser determinadas pela cinemática direta do UVMS para os seus efetuadores finais e subsequente transformação do resultado para o referencial da peça. Em relação à essas variáveis, os ganhos da matriz \mathbf{K}_p utilizados foram 50 para as juntas translacionais e 10 para as juntas rotativas.

As trajetórias percorridas pelos efetuadores finais são traçadas no gráfico da Figura 53. Uma composição das posturas do sistema de intervenção em diferentes instantes da execução da tarefa é mostrada na Figura 54. Os gráficos da Figura 55 apresentam a variação das

componentes de postura dos efetuadores dos manipuladores 1 e 2 ao longo do tempo.

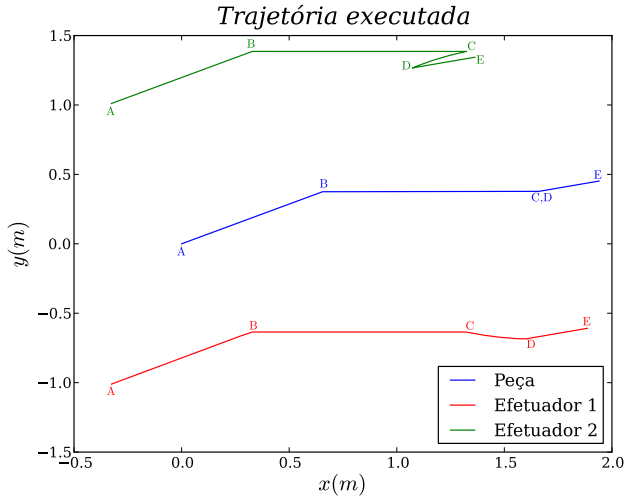


Figura 53. Movimento da peça e dos efetuadores finais

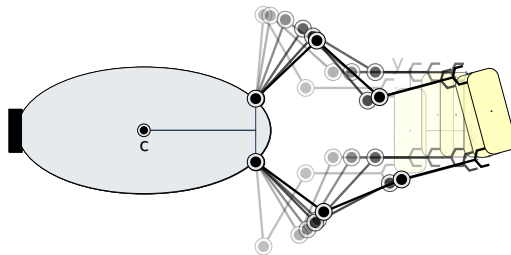


Figura 54. Movimento do UVMS durante a execução da tarefa

As posições das juntas dos manipuladores e suas velocidades são apresentadas nas Figuras 56 e 57.

Nesse cenário aparecem as cadeias de posição relativa do efetuador final em relação à peça, o que acrescenta um maior número de variáveis à equação de restrição. Além disso, o uso de mais um manipulador implicou no acréscimo de um circuito à cadeia cinemática.

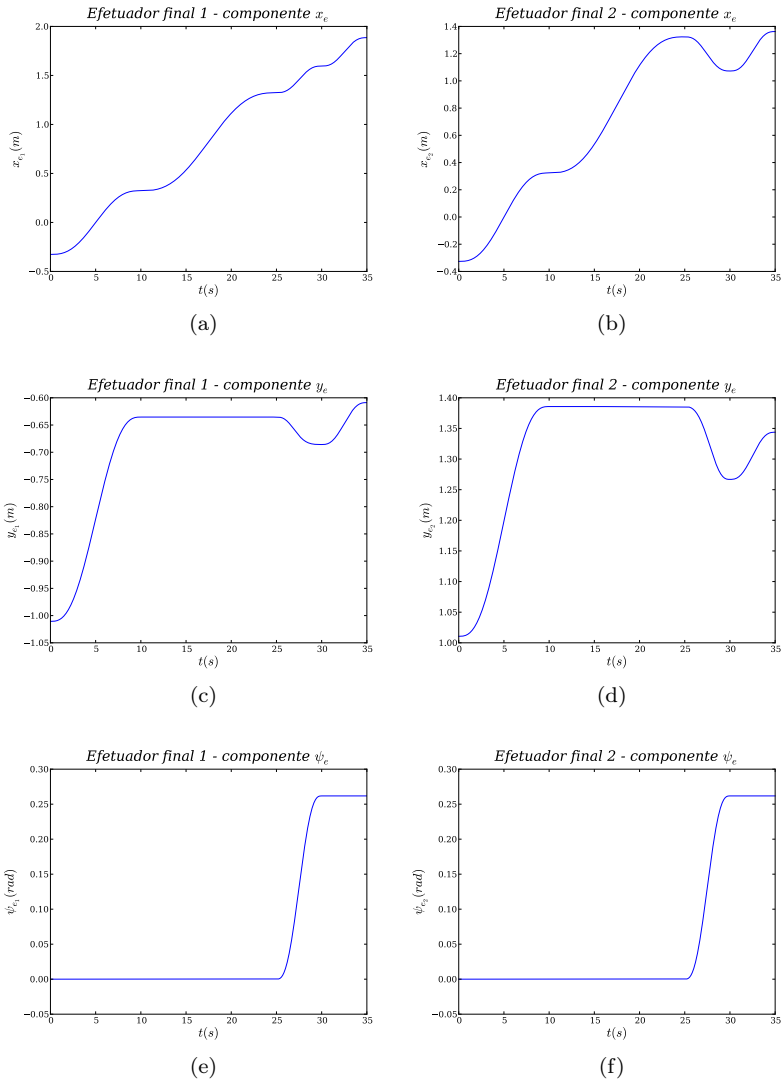


Figura 55. Postura dos efetadores finais ao longo do tempo

A implementação dessa simulação tem a mesma estrutura básica empregada na implementação da simulação do primeiro caso do primeiro cenário, modificando-se apenas a instância de *Uvms* empregada e os

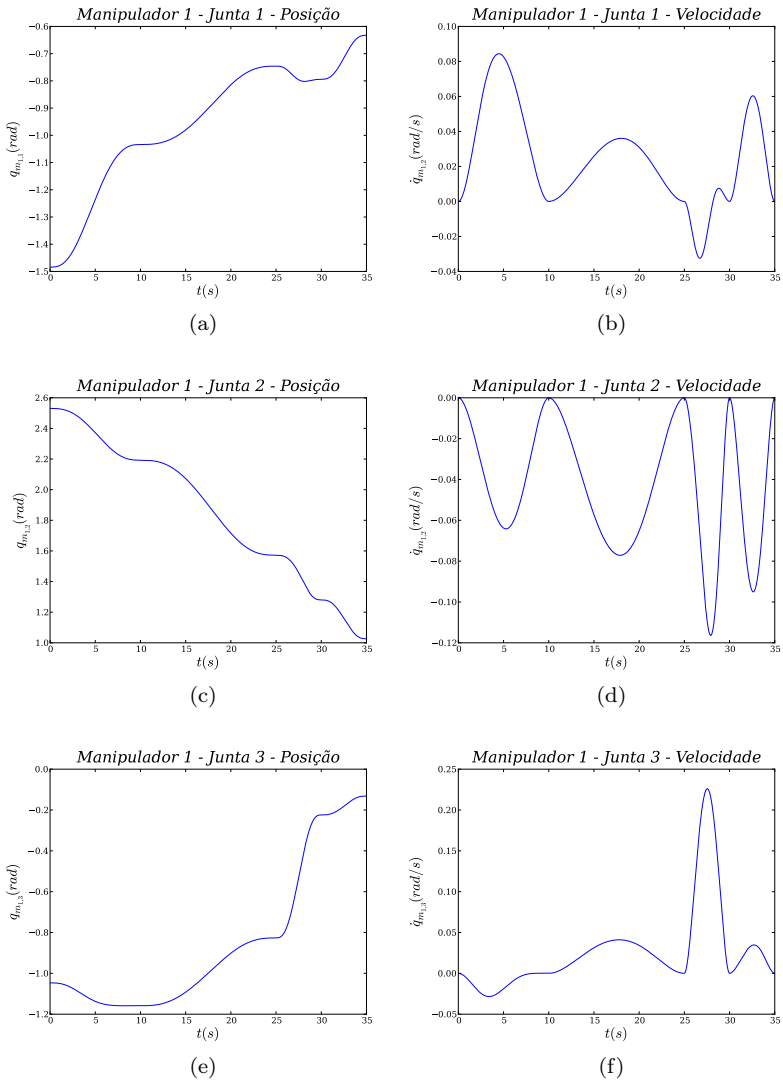


Figura 56. Variáveis do manipulador 1

arquivos de definição do modelo da cadeia cinemática e da tarefa. Assim, a sistematização definida pelos *frameworks* facilitou o reuso de componentes e o desenvolvimento de novas simulações.

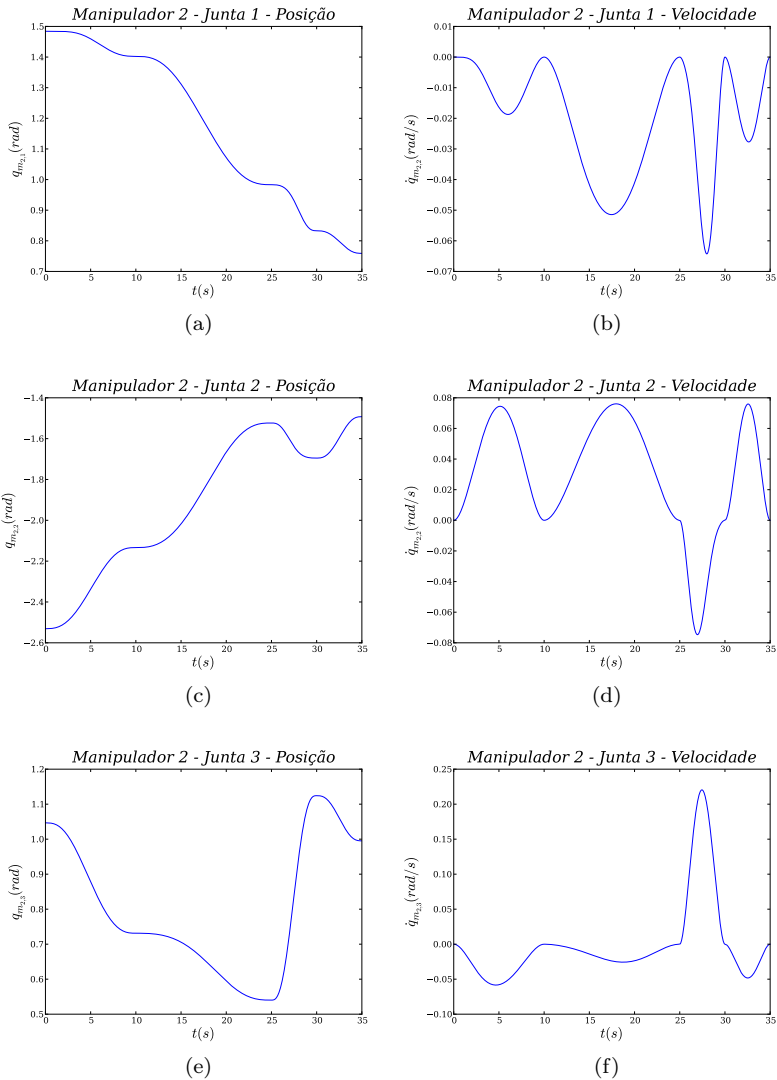


Figura 57. Variáveis do manipulador 2

Na simulação realizada, os circuitos independentes da cadeia do sistema de intervenção foram descritos em um atributo de `KCComposable` de forma que a matriz de rede fosse gerada como foi definida no

Apêndice C. Observa-se, porém, que a hierarquia de classe `KinematicChain` tem um método de determinação de circuitos independentes com base no pacote `NetworkX` de manipulação de grafos do Python (NetworkX, 2011). No caso simulado, esse método gerou uma matriz de circuitos diferente da definida na modelagem do Apêndice C, também correta. Uma simulação executada com a matriz gerada pelo método do *framework* produziu os mesmos resultados da simulação original.

7.3 DOIS UVMS EM COOPERAÇÃO

A cooperação entre UVMS é um tema ainda pouco explorado na literatura. A modelagem cinemática por helicoides de um sistema de intervenção desse tipo foi um dos casos considerados na análise do Capítulo 4. O cenário em questão utiliza dois UVMS planares com a mesma estrutura cinemática daquele tratado na Seção 7.1. O modelo deste cenário é desenvolvido na Seção C.3.3, e uma visualização deste é mostrada na Figura 58.

O modelo cinemático deste sistema foi descrito em um arquivo XML. Assim, simulações baseadas no *framework* `Kast` podem facilmente instanciar esse sistema de intervenção através do *factory method* da classe `KCFactory`.

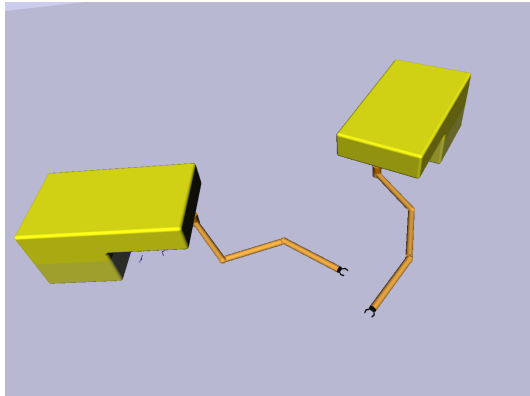


Figura 58. Cenário de simulação de dois UVMS em cooperação

7.3.1 Posicionamento de uma peça

A tarefa a ser realizada nessa simulação consiste na movimentação de uma peça retangular entre dois pontos como se fosse desconectada de um equipamento para ser reconectada em outro. A posição inicial da peça é coincidente com a origem do sistema de coordenadas inercial e seus eixos estão inclinados 30° em relação a esse sistema, ou seja $\boldsymbol{\eta}_{p,t=0} = [0 \ 0 \ 0,524]^\text{T}$. Ao final da tarefa, que deve ocorrer em $25s$, a postura da peça deverá ser $\boldsymbol{\eta}_{p,t=25} = [-0,75 \ -2 \ 0,524]^\text{T}$.

Diferentes ações são realizadas sobre a peça. Nos primeiros $5s$ ela deve rotacionar -30° , alinhando seu referencial com o referencial inercial. A seguir, ela se desloca $-2m$ verticalmente em um período de $10s$. Segue-se um movimento horizontal de $-0,75m$ em um tempo de $5s$. Por fim, ela é rotacionada 30° na posição final em $5s$.

Os efetuadores finais dos dois UVMS são posicionados ao longo do eixo x da peça. Suas posturas em relação ao referencial da peça são $\mathbf{q}_{r_1} = [-2,03 \ 0 \ 0]^\text{T}$ e $\mathbf{q}_{r_2} = [-2,015 \ 0 \ -3,142]^\text{T}$. Os veículos se mantêm estacionários durante a execução da tarefa em $\boldsymbol{\eta}_{v_1} = [-8,25 \ 0,45 \ -0,349]^\text{T}$ e $\boldsymbol{\eta}_{v_2} = [7,45 \ 2,05 \ -2,618]^\text{T}$.

A trajetória da peça é descrita pelas variáveis de juntas da cadeia cinemática do sistema de intervenção q_{t_1} , q_{t_2} e q_{t_3} . Os veículos são representados pelas variáveis $q_{v_{i,j}}$, onde $i = 1, 2$ identifica o veículo e $j = 1, 2, 3$ identifica a junta da cadeia virtual associada a ele. Esse conjunto de variáveis forma a partição primária da equação de restrição. As referências para as variáveis da peça são fornecidas por geradores de trajetória compostos (TrajSequence) agregando instâncias de geradores de trajetórias lineares (TrajLinearPoly) e de valores constantes (TrajConst). Para as demais variáveis são utilizados geradores de trajetória instanciados de TrajConst.

A implementação da simulação segue a sistematização já utilizada nos cenários anteriores. As variáveis a serem determinadas são as das juntas dos dois manipuladores: $q_{m_{i,j}}$, $i = 1, 2$ e $j = 1, 2, 3$ identificando o manipulador e a variável de junta deste, respectivamente. Novamente, a matriz \mathbf{N}_s é quadrada. O passo da simulação é $0,01$, e o método de integração das velocidades é o de Euler. Foi utilizado o modo de cinemática inversa de malha fechada do componente Guidance. Os ganhos da matriz \mathbf{K}_p foram definidos como 50 para as juntas translacionais das cadeias relativas r_i e 10 para as suas juntas rotativas.

As trajetórias percorridas pelos efetuadores finais e pela peça são mostradas no gráfico da Figura 59. Na Figura 60 apresentam-se as

posturas do sistema de intervenção ao longo da execução da tarefa. As posturas dos efetuadores finais são traçadas nos gráficos da Figura 61.

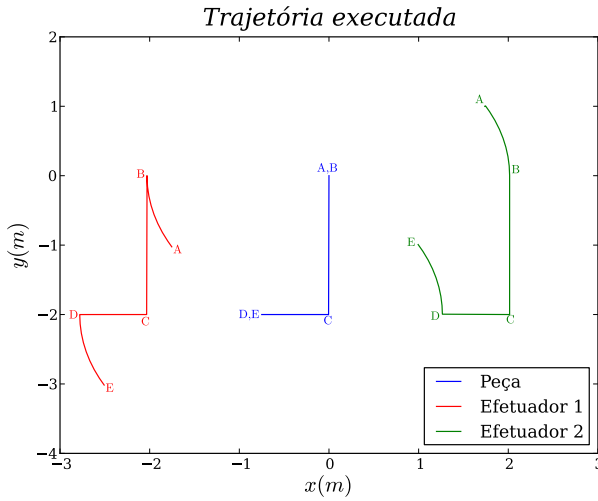


Figura 59. Movimento da peça e dos efetuadores finais

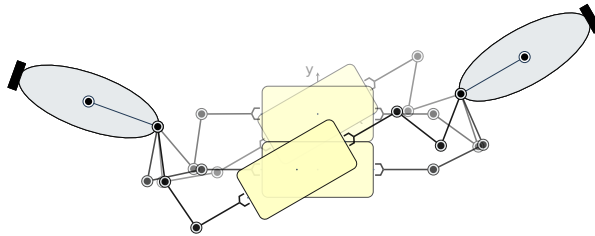


Figura 60. Movimento dos UVMS durante a execução da tarefa

As posições das juntas dos manipuladores e suas velocidades são apresentadas nas Figuras 62 e 63.

Como no cenário anterior, a cadeia forma dois circuitos independentes, porém com arranjos diferentes da cadeia daquele sistema. Novamente, a implementação da simulação se beneficiou do reuso de componentes de software, bastando derivar uma classe de *Uvms* para representar o sistema de intervenção. O modelo da cadeia cinemática e a tarefa foram descritos em um arquivo XML, como anteriormente.

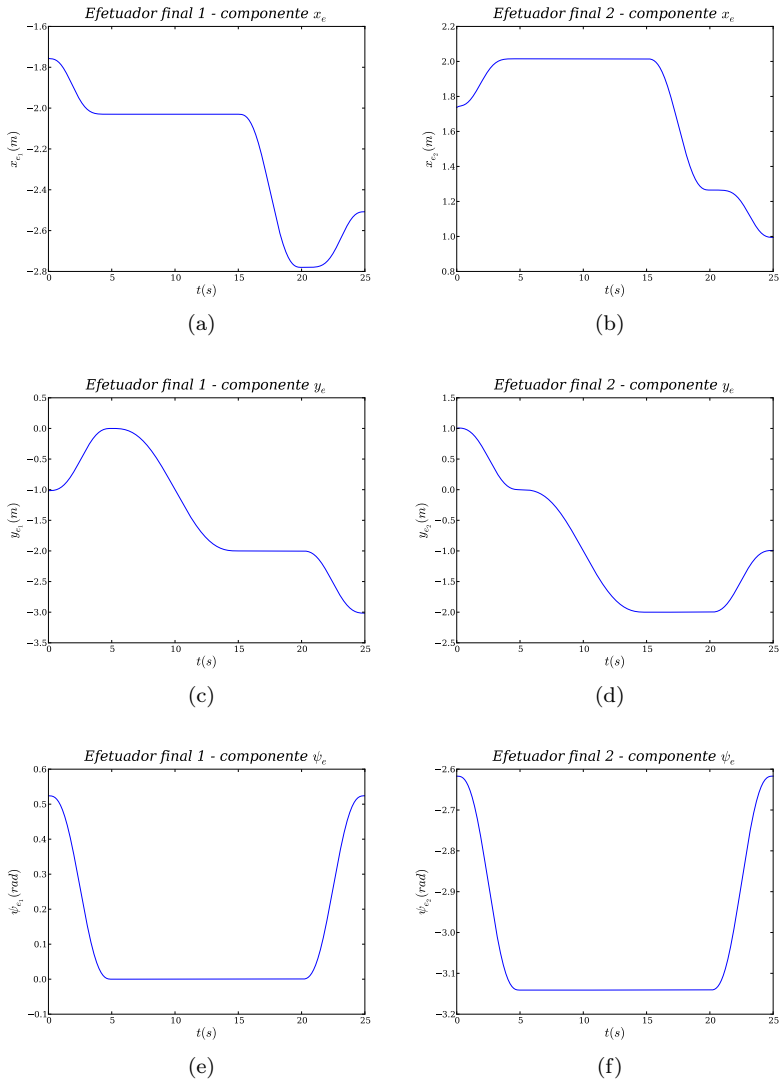


Figura 61. Postura dos efetuadores finais ao longo do tempo

Assim como no cenário anterior, os circuitos independentes da cadeia foram definidos em um atributo de `KCComposable` para que a matriz de rede fosse gerada como foi definida no Apêndice C, possibi-

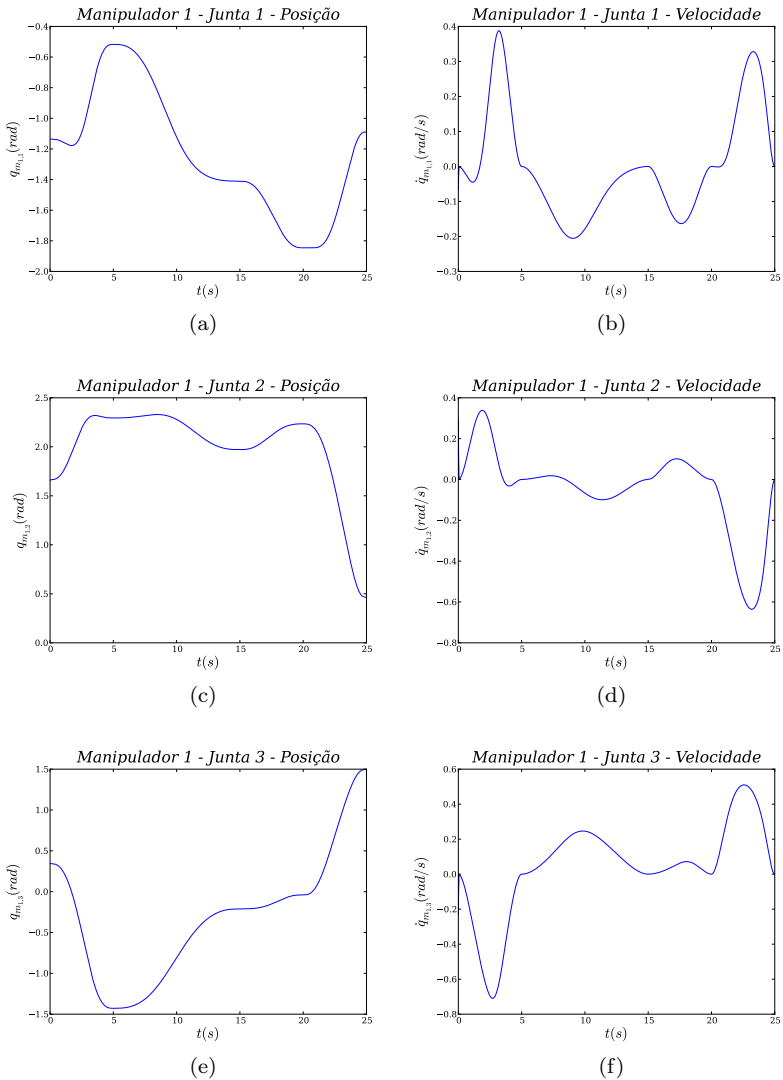


Figura 62. Variáveis do manipulador 1

litando a depuração do código e verificação de sua correção. Porém, os mesmos resultados dessa simulação foram observados quando foi utilizada a matriz de circuitos gerada pela instância de KinematicChain.

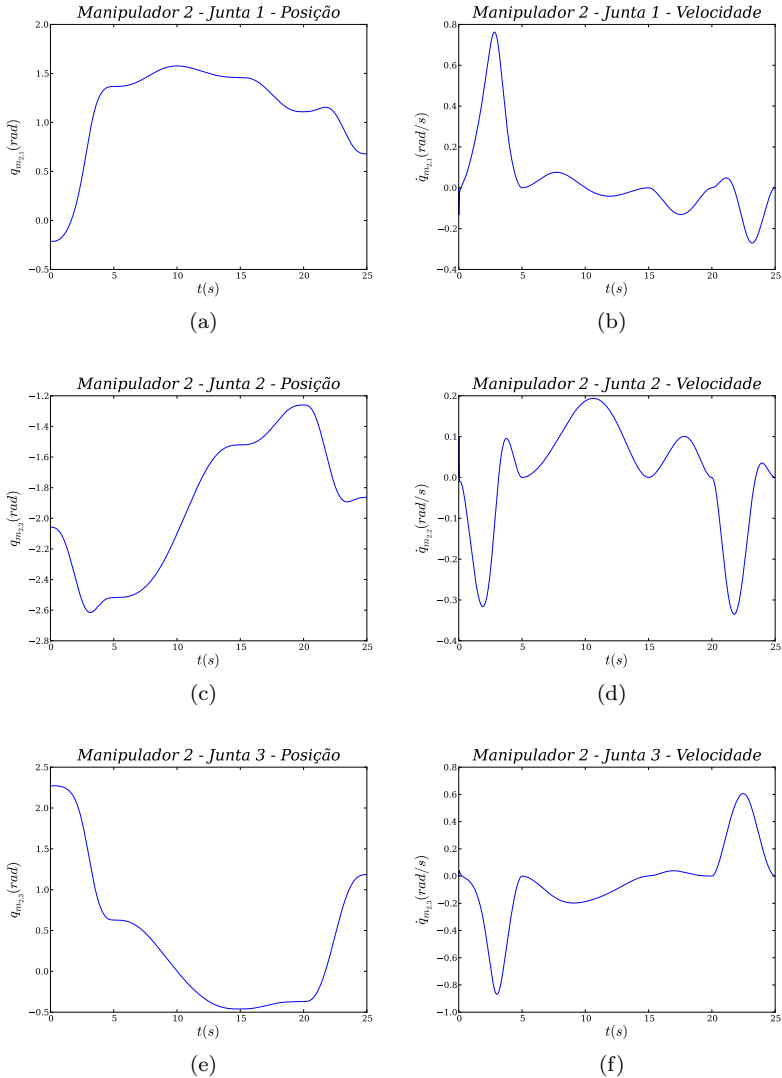


Figura 63. Variáveis do manipulador 2

7.3.2 Execução de tarefa que exige deslocamento dos UVMS

O primeiro caso desse cenário tratou de uma tarefa onde os veículos puderam manter sua posição e orientação. Um segundo caso é

definido de forma que os veículos eventualmente tivessem de abandonar a sua condição estacionária. A tarefa em questão é uma variação da primeira, onde o deslocamento vertical passa a $-3,5m$ em $17s$ e o deslocamento horizontal subsequente é de $-1,25m$ em $7s$. O tempo total de execução da tarefa passa então a $34s$. Na Figura 64 são traçados os deslocamentos da peça e dos efetuadores finais dos UVMS realizados durante a execução da tarefa.

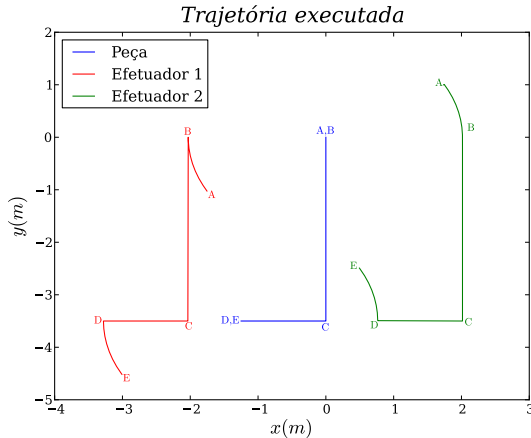


Figura 64. Movimento da peça e dos efetuadores finais

Uma classe derivada de *Mission* é criada para implementar o método *verifyUpdate* que monitore e modifique a cadeia cinemática do cenário de acordo com a necessidade. Cada UVMS tem seu próprio estado de operação, que é definido de acordo pelo método. *verifyUpdate* verifica a proximidade de singularidade das juntas $q_{m_{i,2}}$, $i = 1, 2$ do sistema, onde i identifica o UVMS ao qual o manipulador está vinculado.

Inicialmente, ambos UVMS estão em estado estacionário. Ao se detectar a proximidade de singularidade, um UVMS passa ao estado móvel, e então o reparticionamento é feito de forma que as juntas q_{v_i} , $q_{m_{i,1}}$ e $q_{m_{i,3}}$ passem a ser secundárias, enquanto $q_{m_{i,2}}$ passa a ser primária, com um gerador de trajetória *TrajLinearPoly* associado à ela para levar sua posição à um limite de segurança ($0,78rad$) em um período de $3s$. O retorno ao estado estacionário acontece ao se verificar que a junta $q_{m_{i,2}}$ está com um valor acima do limite de segurança, onde o reparticionamento de variáveis do UVMS volta ao estado original e os

geradores de referências do veículo são atualizados. A verificação dos estados é feita de forma independente para os dois UVMS.

Na execução da simulação, o UVMS 1 esteve no estado móvel durante o intervalo entre 31,32s e 33,76s. O UVMS 2 esteve no estado móvel em dois intervalos, o primeiro entre 17,81s e 20,24s e o segundo entre 25,66s e 28,09s. O movimento translacional dos veículos durante a execução da tarefa é esboçado na Figura 65.

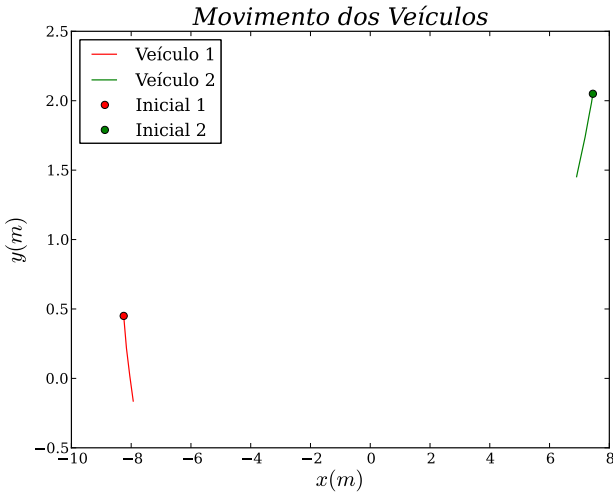


Figura 65. Translação dos veículos durante a execução da tarefa

As Figuras 66 e 67 mostram as variáveis do UVMS 1. As variáveis do UVMS 2 são apresentadas nas Figuras 68 e 69. Os picos de velocidade observados são causados pelo chaveamento de estados. Um estudo para minimizar esse comportamento na troca de estados é analisado em (Simas; Fontan; Martins, 2011).

Verifica-se pela Figura 64 que a tarefa é executada pelos efetadores finais de acordo com as definições de movimento da peça e seu relacionamento com os efetadores. Com o uso do modelo híbrido de estados, foi possível executar a tarefa fazendo os veículos se movimentarem apenas em situações onde os manipuladores a eles vinculados estivessem próximos a singularidades. Apesar das descontinuidades no movimento das juntas, observa-se que a trajetória é contínua.

Em relação à implementação, fez-se apenas a modificação do componente de missão utilizado de forma que o método `verifyUpdate`

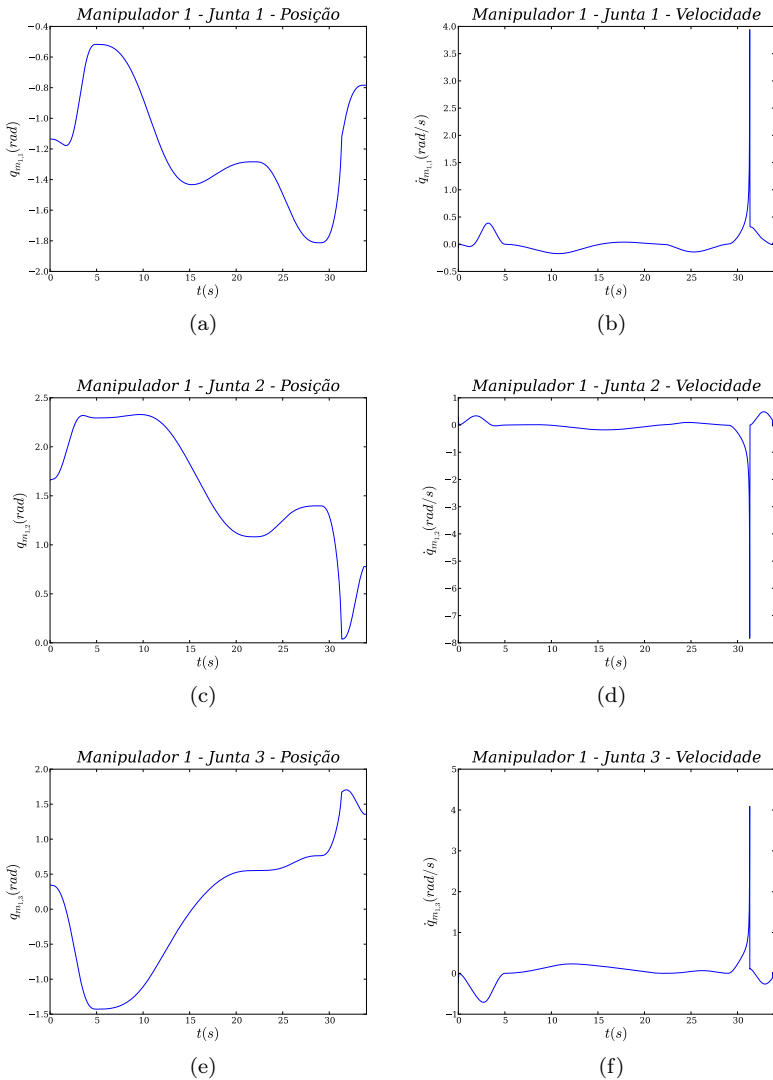


Figura 66. Variáveis do manipulador do UVMS 1

fosse definido para tratar da verificação ocorrência de eventos de mudança de estados e a adequação da cadeia cinemática e geradores de trajetória para quando o estado de cada veículo fosse modificado.

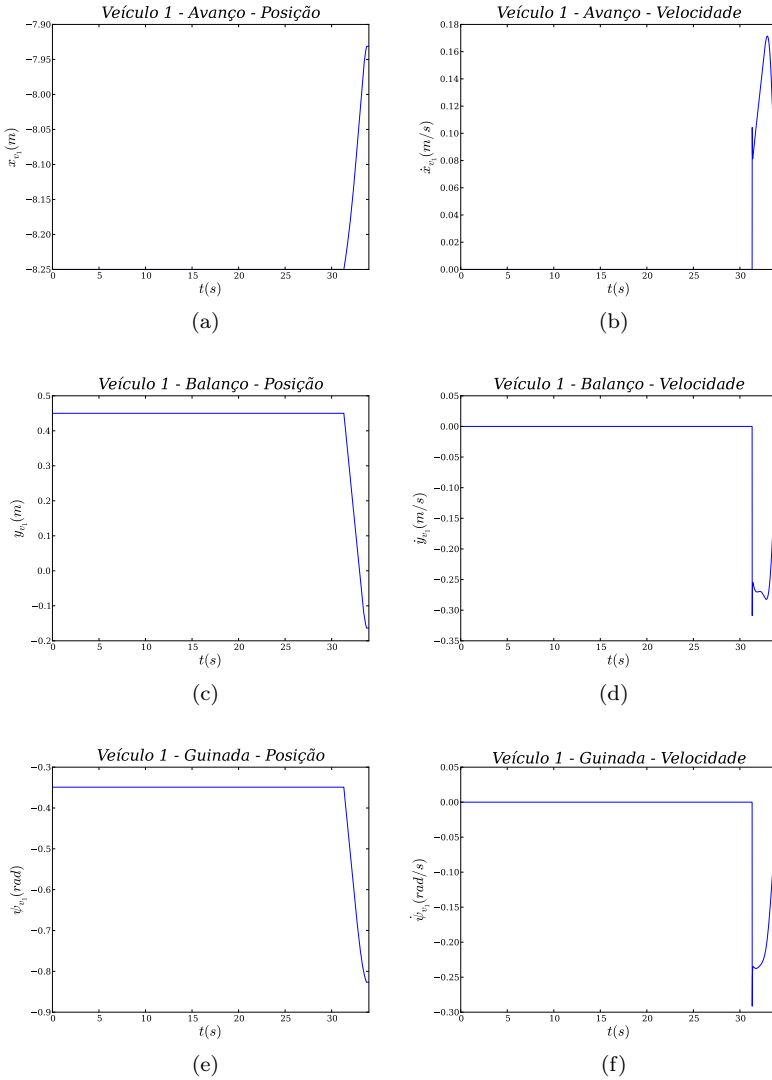


Figura 67. Variáveis do veículo do UVMS 1

7.4 DOIS UVMS COM DOIS MANIPULADORES EM COOPERAÇÃO

Em aplicações como a exploração de petróleo em águas profundas, é comum ter sistemas de intervenção com pelo menos dois mani-

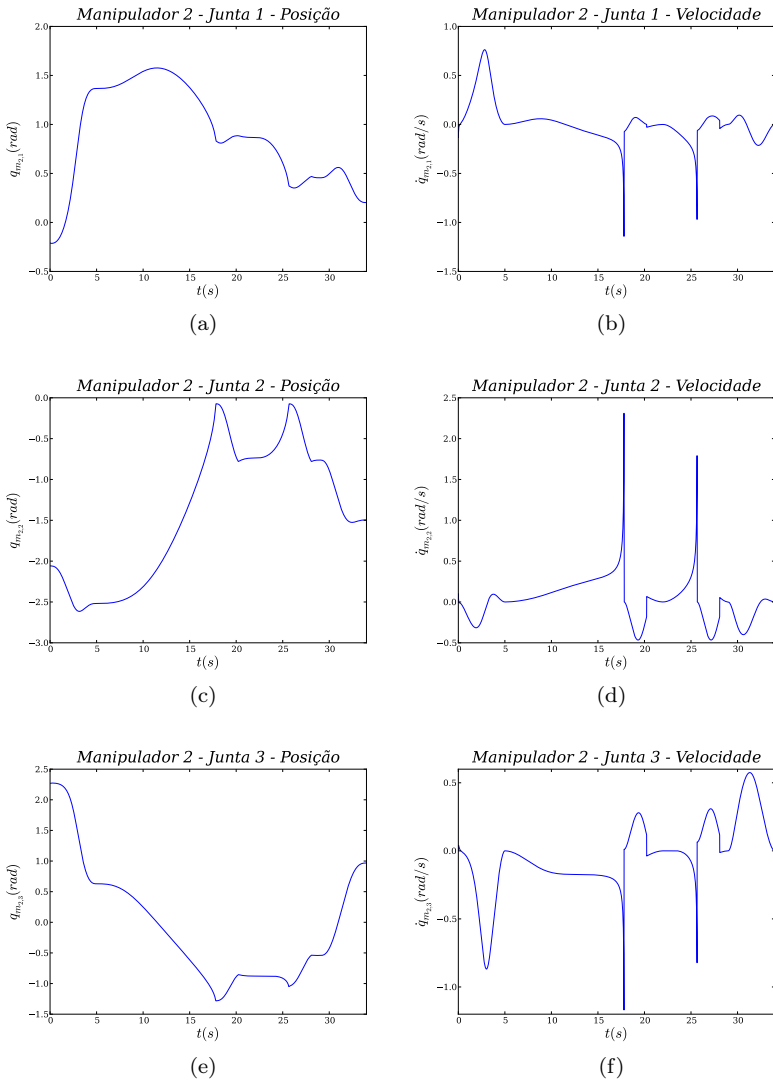


Figura 68. Variáveis do manipulador do UVMS 2

puladores. Eles usualmente trabalham de forma isolada. O desenvolvimento de autonomia de operação pode facilitar a cooperação entre esses sistemas, e com isso novas aplicações. O cenário em questão sur-

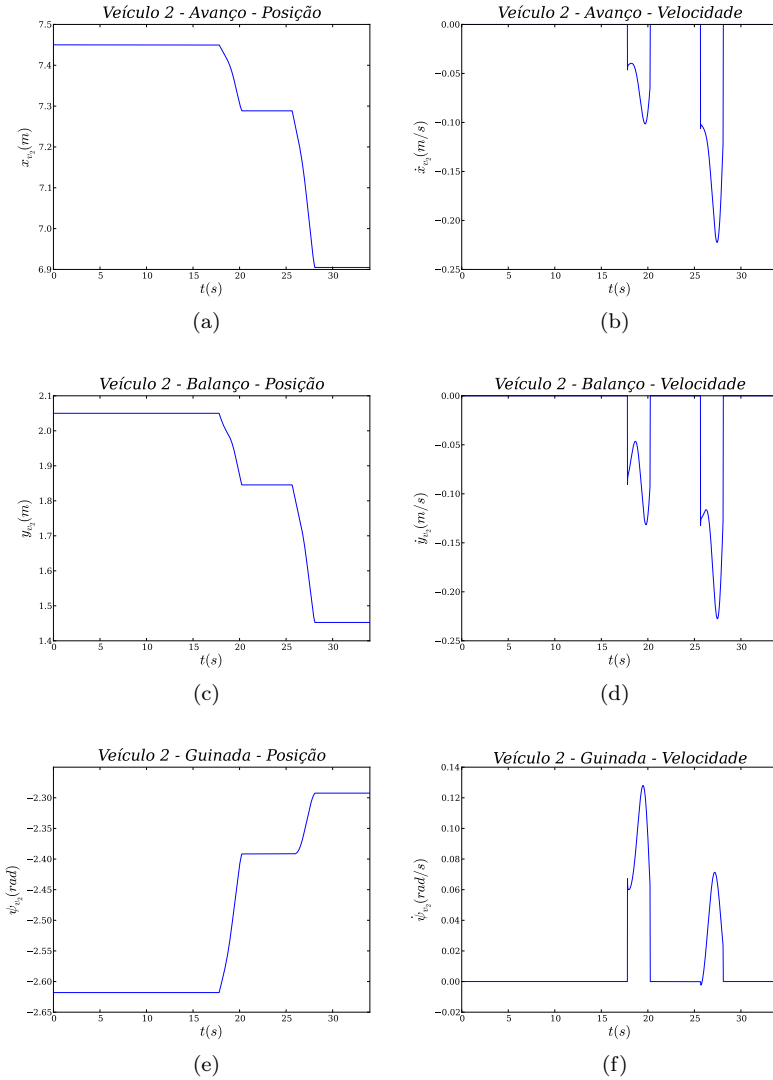


Figura 69. Variáveis do veículo do UVMS 2

giu do desenvolvimento do modelo de cooperação entre manipuladores de dois UVMS do Capítulo 4, sendo detalhado na Seção C.3.4 para um caso planar. Cada UVMS tem dois manipuladores vinculados, como é

mostrado na Figura 70.

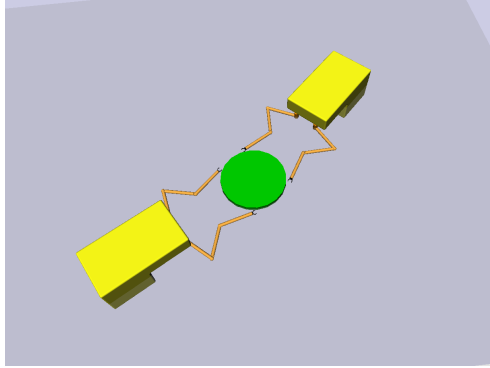


Figura 70. Cenário de simulação de dois UVMS com dois manipuladores cada em cooperação

Como nos demais cenários, a cadeia cinemática correspondente ao sistema de intervenção foi descrito em um arquivo XML para compor o banco de cadeias cinemáticas e seu uso em simulações com o Kast.

7.4.1 Posicionamento de uma estrutura cilíndrica

A tarefa desta simulação baseou-se nas operações de contenção do vazamento de petróleo causado pelo acidente da plataforma *Deepwater Horizon* (British Petroleum, 2010; Leff; Plushnick-Masti, 2010). Uma estrutura cilíndrica de contenção deve ser corretamente posicionada sobre o local do vazamento, e dois UVMS são usados para esse fim em função das dimensões da estrutura. Assume-se que a posição inicial da peça é a origem do referencial inercial. O cilindro de contenção tem raio $1,75m$, e é assumido como ponto de referência o seu centro.

A peça deve se deslocar em uma trajetória definida por diferentes pontos de passagem, levando-a da origem do referencial inercial à posição $x_p = 1,5$, $y_p = -0,25$. Durante esse deslocamento, a peça rotaciona 15° . Essa operação ocorre em um período de $30s$. Após, a peça deve ser deslocada mais $-0,25m$ na direção vertical mantendo a orientação em um período de $15s$.

Durante o período de execução da tarefa ($45s$), os veículos permanecem estacionários. As posturas são $\eta_{v_1} = [-6,9 \quad 4,0 \quad 0,524]^T$ e $\eta_{v_2} = [5,66 \quad 5,66 \quad -2,356]^T$.

Os efetuadores finais dos quatro manipuladores envolvidos devem manter posturas constantes em relação ao referencial da peça. Estas são definidas pelas cadeias r_{ij} , onde $i = 1, 2$ e $j = 1, 2$ designam o veículo e o manipulador respectivamente. Suas posturas são $\mathbf{q}_{r_{11}} = [-1, 24 \quad -1, 27 \quad 0, 75]^T$, $\mathbf{q}_{r_{12}} = [-1, 478 \quad 0, 78 \quad 0, 088]^T$, $\mathbf{q}_{r_{21}} = [-0, 217 \quad 1, 756 \quad -1, 466]^T$ e $\mathbf{q}_{r_{22}} = [1, 756 \quad -0, 004 \quad -3, 089]^T$.

A trajetória da peça, representada pelas variáveis de juntas q_{t_1} , q_{t_2} e q_{t_3} , foi gerada por duas instâncias de *TrajSequence*, uma para posição e outra para orientação. Em relação à posição, a trajetória curva foi gerada por uma B-spline cúbica a partir de pontos de passagem especificados (uma instância de *TrajSpline*). O trecho de deslocamento vertical foi gerado por um interpolador linear polinomial de 5ª ordem (*TrajLinearPoly*). Quanto à orientação, a sua variação durante os 30s iniciais foi gerada por uma instância de *TrajLinearPoly*, seguida por uma instância de *TrajConst* para gerar referências de valor constante. Para as demais variáveis primárias (as dos veículos e das cadeias relativas dos efetuadores à peça) foram utilizadas instâncias de *TrajConst*.

As variáveis das juntas dos manipuladores são identificadas como $q_{m_{vi,j}}$, onde $v = 1, 2$, $i = 1, 2$ e $j = 1, 2, 3$ correspondem respectivamente ao veículo, ao manipulador e à junta. A cadeia cinemática desse sistema forma quatro circuitos independentes, e com isso a matriz \mathbf{N}_s é quadrada para esse particionamento. O passo da simulação é 0,01, sendo usado o método default (Euler) para integração das velocidades. A instância de *Guidance* foi definida para usar a cinemática inversa de malha fechada em relação às variáveis das cadeias r_{ij} , com ganhos iguais a 50 para as juntas translacionais e 10 para as juntas rotativas.

As trajetórias percorridas pelos efetuadores finais e pela peça são mostradas na Figura 71. Na Figura 72 apresentam-se as posturas que o sistema de intervenção assume ao longo da execução da tarefa. As posturas dos efetuadores são traçadas nos gráficos das Figuras 73 e 74.

As posições das juntas dos manipuladores e suas velocidades são apresentadas nas Figuras 75 a 78. As descontinuidades observadas nas velocidades, como por exemplo no realce da Figura 75b, são causadas pela troca entre os geradores de trajetória usados para cada trecho. A continuidade no chaveamento de geradores de trajetória para cada trecho de caminho é um problema a ser abordado em versões futuras do *framework* Kast.

Apesar da cadeia cinemática mais complexa, com aumento do número de circuitos independentes (e conseqüentemente de variáveis), a implementação da simulação foi facilitada pelo reuso dos componentes dos *frameworks* e das simulação anteriores. As modificações em

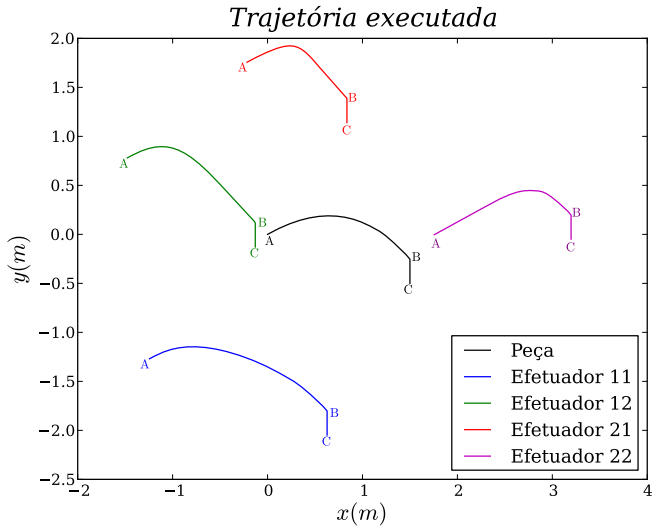


Figura 71. Movimento da peça e dos efetuidores finais

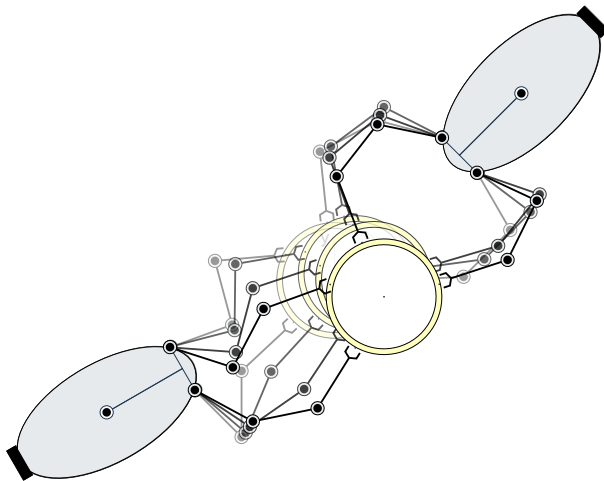


Figura 72. Movimento dos UVMS durante a execução da tarefa

relação aos cenários anteriores foram a criação de uma classe derivada de *Uvms* para determinar a cinemática direta dos manipuladores e pos-

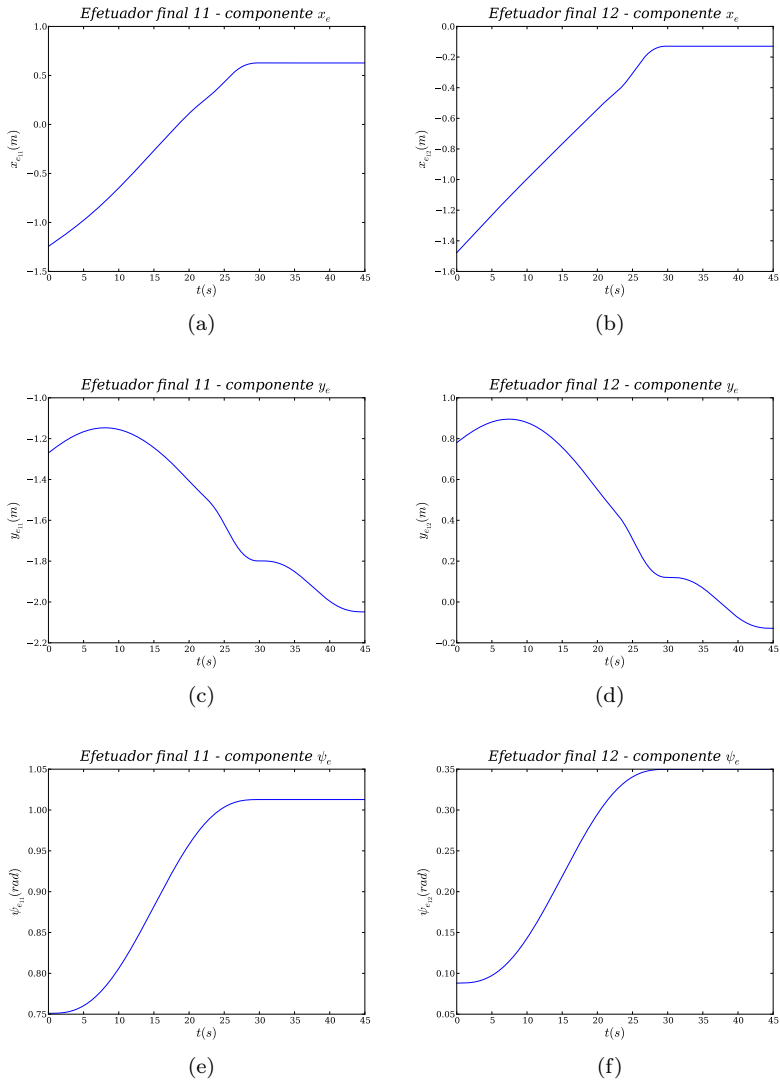


Figura 73. Postura dos efetuadores finais 11 e 12 ao longo do tempo

terior cálculo das posições relativas destes em relação à peça no referencial desta, e na descrição da cadeia cinemática em um arquivo XML. A tarefa também foi descrita em arquivo XML.

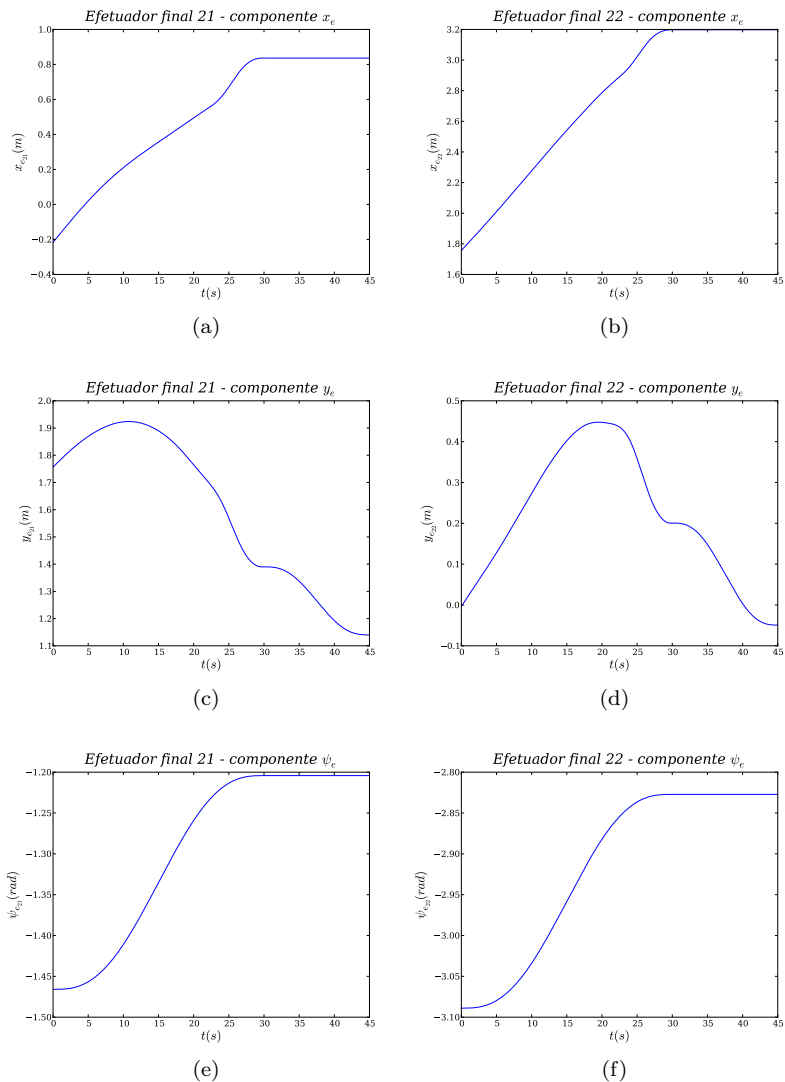


Figura 74. Postura dos efetadores finais 21 e 22 ao longo do tempo

Assim como nos cenários anteriores, os circuitos independentes da cadeia cinemática foram definidos em um atributo de `KCComposable` para que a matriz de rede fosse gerada como foi definida no Apêndice C.

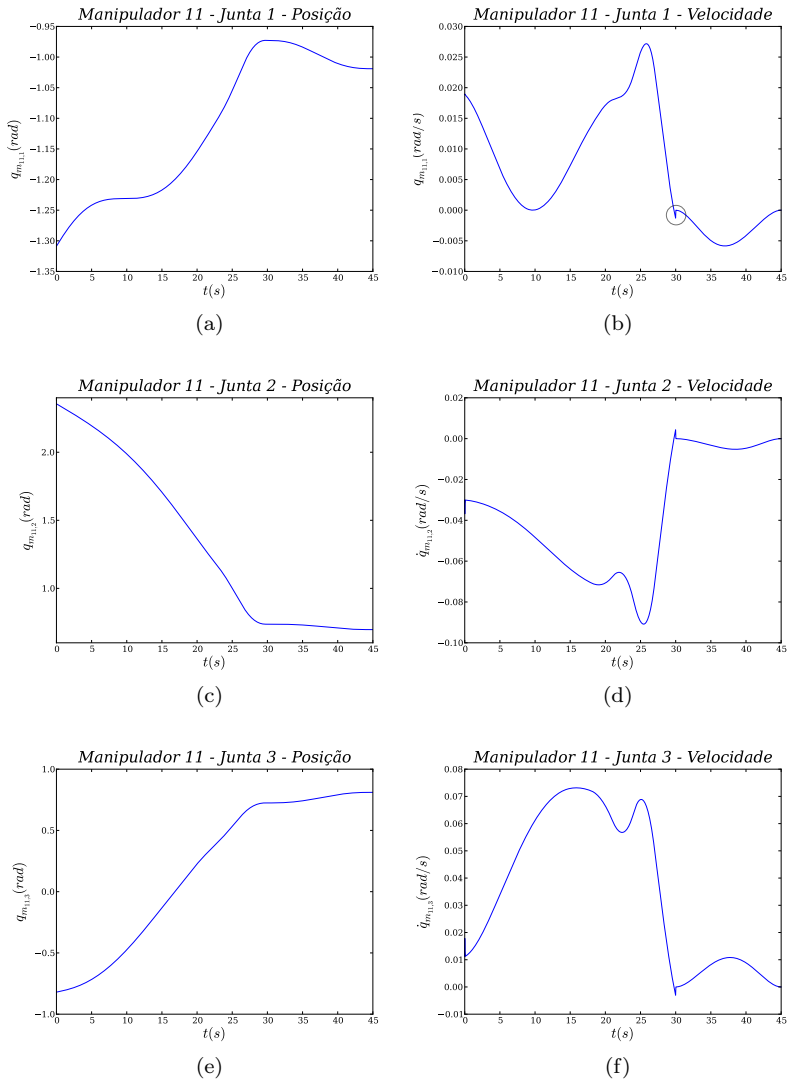


Figura 75. Variáveis do manipulador 11

Os testes com a matriz de circuitos calculada pelo Kast resultaram nos mesmos resultados, como se havia observado nas simulações anteriores.

As simulações apresentadas neste capítulo demonstraram a efe-

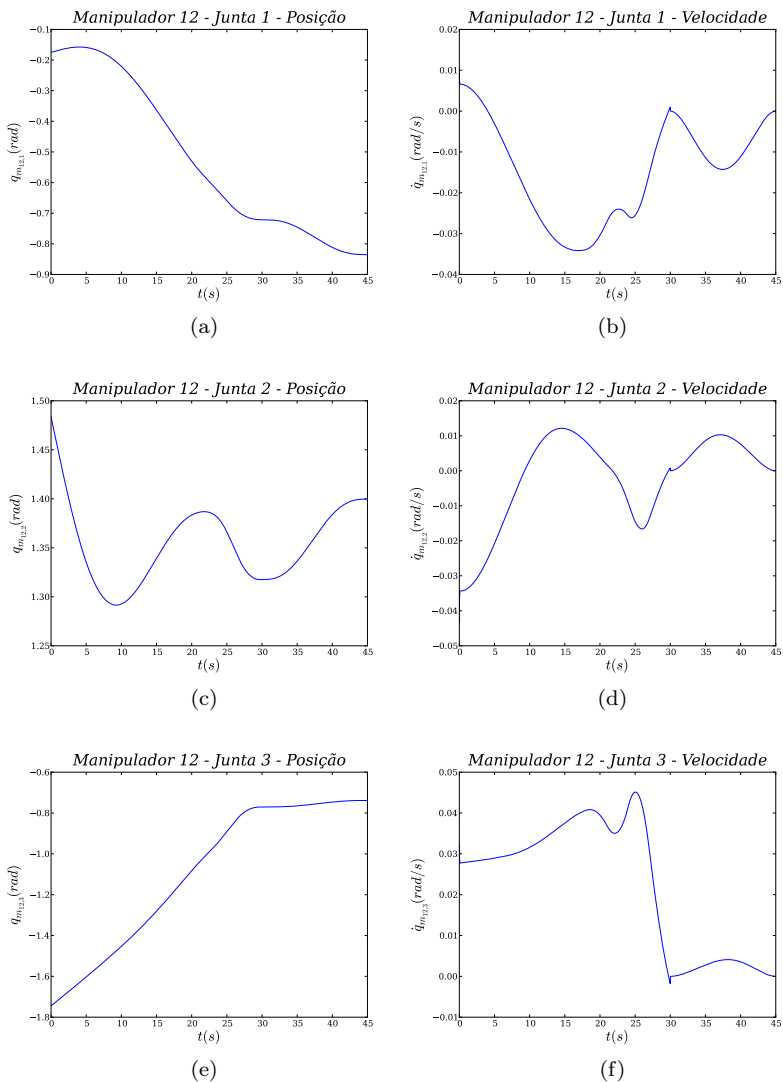


Figura 76. Variáveis do manipulador 12

tividade da sistematização proposta nesta tese. O uso dos *frameworks* projetados nos capítulos anteriores facilitou o desenvolvimento das simulações através de uma organização bem definida destas. Além disso,

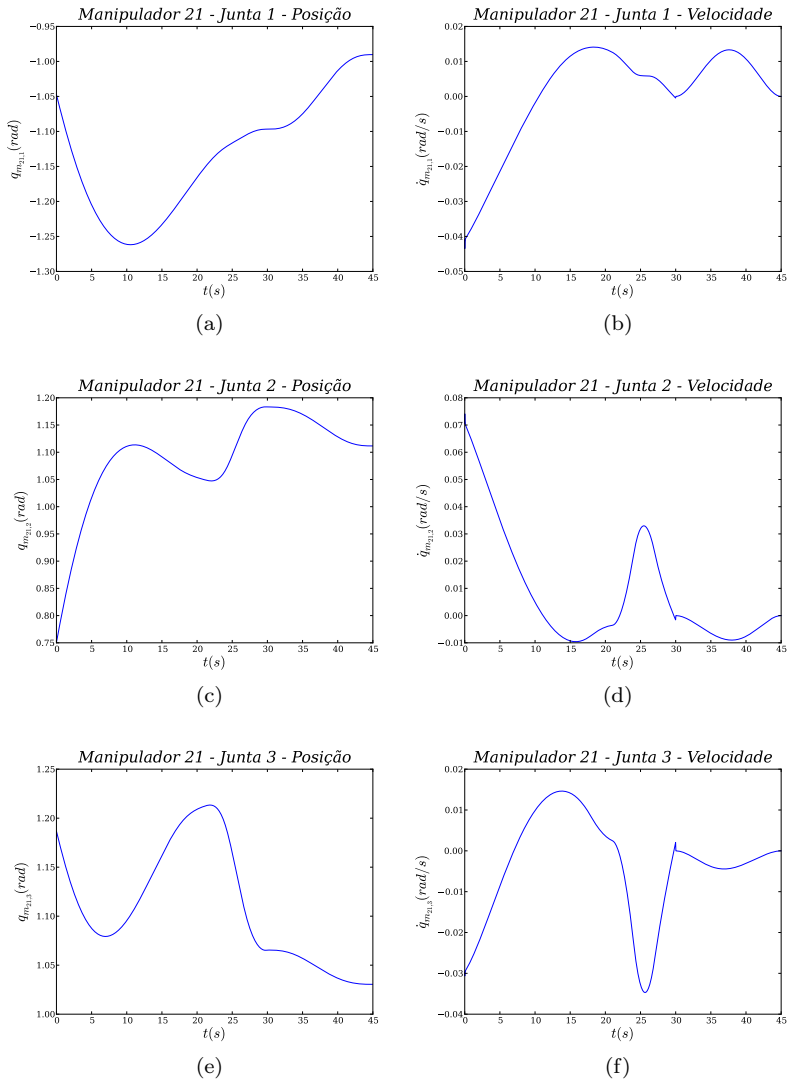


Figura 77. Variáveis do manipulador 21

ressaltam-se outras características que agilizaram o desenvolvimento, como o reuso de componentes e a extensibilidade. Os resultados das diferentes simulações ilustra que a sistematização pode ser empregada

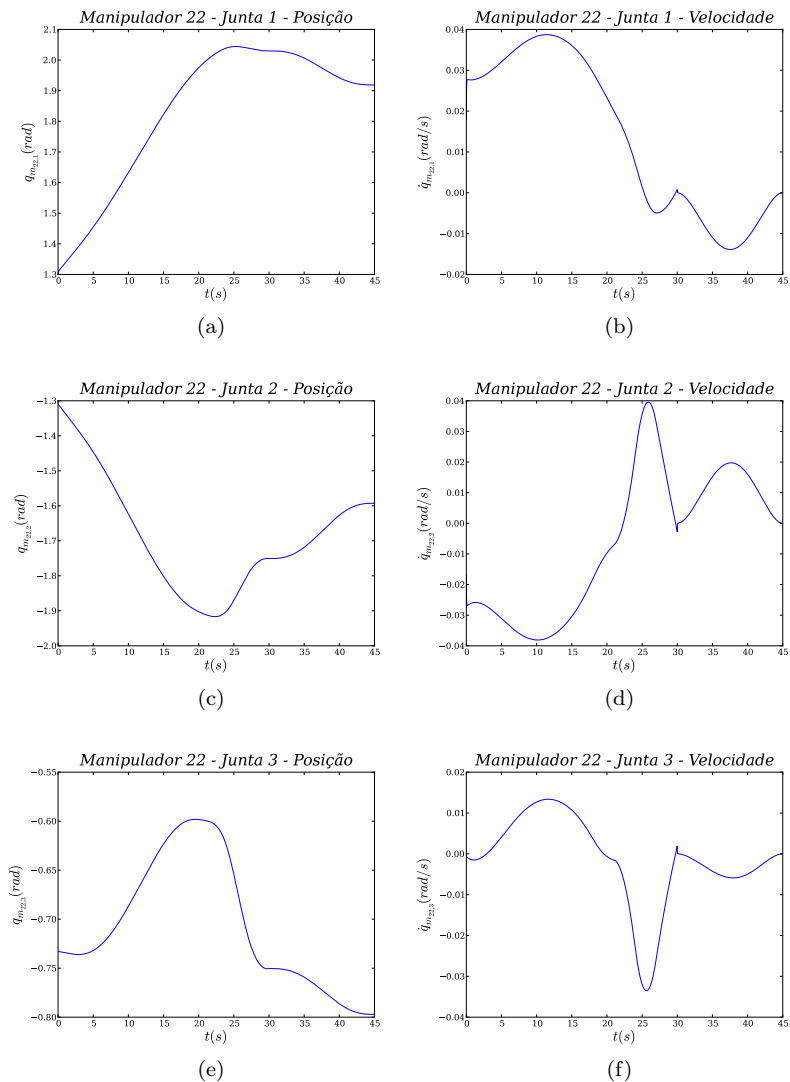


Figura 78. Variáveis do manipulador 22

em estratégias de planejamento de movimento para diferentes cenários de intervenção subaquática, seja em relação ao número de UVMS e de manipuladores envolvidos ou relação às tarefas a serem executadas.

8 CONCLUSÃO

Este trabalho procurou contribuir para a evolução dos sistemas de intervenção subaquática em direção à autonomia plena. Para tanto, fez-se uma análise desses sistemas, destacando os requisitos importantes para a autonomia. Destes, o planejamento de movimento foi definido como o objeto de pesquisa da tese, por ser considerado essencial para a autonomia de operação.

Assim, foram estudadas diferentes configurações cinemáticas de sistemas, visando estender o modelo por helicoides elaborado em trabalhos anteriores. Com base nessa extensão, foi elaborada uma sistematização para desenvolvimento e implementação de estratégias de planejamento de movimento.

As contribuições oriundas deste trabalho e as perspectivas de trabalhos futuros são relacionadas a seguir.

8.1 CONTRIBUIÇÕES DA TESE

As principais contribuições deste trabalho procuraram atender ao objetivo geral definido para a tese, qual seja, analisar a cinemática de sistemas de intervenção subaquática a fim de desenvolver estratégias de planejamento de movimento de UVMS.

A contribuição mais diretamente relacionada ao objetivo geral da tese foi a proposição de uma sistemática de desenvolvimento de estratégias de planejamento de movimento a partir de uma análise cinemática realizada para diferentes cenários de execução de operações de intervenção e da identificação de blocos de atividades do sistema robótico. A sistematização foi feita através da implementação de classes de um *framework* cujas instâncias estruturaram um algoritmo geral de execução da fase de intervenção de uma missão.

A extensão da modelagem cinemática de sistemas de intervenção subaquática baseada em helicoides para diferentes configurações desses sistemas foi outra contribuição deste trabalho. A extensão do modelo ainda trata de características adicionais, como o caso de evitamento de colisão com obstáculos presentes no espaço de trabalho. A análise de casos base para essa extensão definiu como novos manipuladores podem ser incluídos no modelo cinemático. Além disso, verificou-se como representar sistemas de intervenção subaquática onde UVMS atuam em cooperação.

Uma sistematização de obtenção de modelos cinemáticos através de componentização, ou seja, da criação de cadeias cinemáticas pela composição de cadeias mais simples previamente modeladas, foi proposta com base na análise cinemática realizada. Além de agilizar o processo de modelagem, essa abordagem permite considerar a possibilidade de automação do processo através de padrões (*templates*) de cadeias cinemáticas, e de um banco de dados para armazená-las. Além disso, acredita-se que a componentização facilita a reconfigurabilidade de cadeias cinemáticas, o que pode ser necessário em diferentes cenários de execução de missões complexas em um ambiente incerto como o subaquático. Outro resultado do estudo feito para a extensão foi identificar duas formas de modificação de cadeias cinemáticas para refletir mudanças de contextos de execução de tarefas, denominadas reparticionamento e reconfiguração, sendo esta classificação feita pela primeira vez neste trabalho.

Para facilitar a implementação de algoritmos de análise cinemática baseada em helicoides, projetou-se um *framework* computacional orientado a objetos. Sua implementação, feita em Python, é tratada no Apêndice B. O *framework* Kast será disponibilizado como software livre tão logo esteja com todos os requisitos básicos de projeto funcionais e com uma documentação mínima. Para facilitar o seu aprendizado e extensão, todas as implementações desenvolvidas nesta tese acompanharão o pacote.

O *framework de intervenção* definido na sistematização de estratégias de planejamento de movimento foi implementado e utilizado em diferentes simulações que demonstraram sua facilidade de uso. Nestas simulações, observaram-se características como reuso e extensibilidade, típicas do paradigma de orientação a objetos. Esse *framework* também será disponibilizado como software livre junto ao Kast.

As simulações também serviram para avaliar diferentes cenários de execução de tarefas e diferentes situações. Verificou-se que a sistematização proposta é efetiva para tratar de cooperação entre manipuladores e entre UVMS. Em situações excepcionais, como grandes deslocamentos e travamento de juntas, o uso de modelos híbridos de estados resolveu o planejamento de movimento, tratando de deslocar os veículos apenas quando necessário.

Dos estudos iniciais sobre o panorama da robótica subaquática, seus cenários e desafios, surgiram contribuições secundárias, como uma organização da classificação de ROV e AUV, apresentadas nas Tabelas 1, 2, 3 e 4. Além dessa organização, a própria distinção entre ROV e AUV foi considerada inadequada para o estado da arte das pesquisas em

robótica subaquática. Assim, cita-se como uma contribuição secundária a proposta de adoção de uma classificação dos veículos subaquáticos não tripulados em função das missões em que são empregados apresentada na Seção 1.2. Além disso, o desenvolvimento de um mapa conceitual para organizar e relacionar os conhecimentos relativos ao cenário da robótica subaquática atual é considerado outra contribuição derivada deste trabalho.

8.2 PERSPECTIVAS E TRABALHOS FUTUROS

A autonomia de sistemas de intervenção subaquática é um problema em aberto. As perspectivas de uma maior adoção desses sistemas à medida em que cresce o trabalho em águas profundas e ultraprofundas para a exploração de petróleo, além de outras aplicações no ambiente marinho, demandam novos usos desses sistemas e com isso aplicações de maior complexidade. Assim, há uma grande variedade de pesquisas em potencial para serem desenvolvidas. Em relação ao objeto desta tese, identificam-se algumas possibilidades de trabalhos futuros.

Inicialmente, é interessante trabalhar um número maior de cenários de sistemas de intervenção subaquática, principalmente considerando UVMS operando em três dimensões. Basear os modelos em sistemas teleoperados hoje existentes e identificar diferentes tarefas que são executadas por esses sistemas seria de grande valia para avaliar a sistematização proposta e aproximar as estratégias de planejamento de movimento da aplicação em situações reais.

Nos estudos iniciais da extensão da modelagem de UVMS por helicoides, foram identificados casos que não estavam diretamente relacionados com a ideia de intervenção subaquática, mas que poderiam ser tratados pela mesma modelagem. Os pelotões ou enxames de UUV em aplicações científicas como o monitoramento/mapeamento de grandes áreas têm gerado o interesse de pesquisadores. Acredita-se que há potencial de aplicação da sistemática de planejamento de movimento para esse tipo de aplicação a partir de similaridades com sistemas cooperativos. Ainda em relação a pelotões, a identificação de objetivos a serem alcançados/seguídos através de sensores dos UUV poderia ser integrada às estratégias de planejamento de movimento (no que se chama de *visual servoing/tracking*).

Outro potencial uso desse modelamento seria na análise da relação entre os propulsores e o movimento do veículo subaquático. A partir de um modelo relacionando o movimento do veículo com os mo-

vimentos dos propulsores, seria possível fazer uma análise da influência de cada propulsor e otimizações de projeto.

Faz-se necessária uma análise de como implementar estratégias para identificar mudanças de contextos de operação e gerir as modificações necessárias na cadeia cinemática e na tarefa. O estudo das descontinuidades geradas por modificações do modelo cinemático em função de mudanças de contexto é outro problema a ser abordado, pelos efeitos sobre a estrutura dos UVMS. Esse problema já vem sendo estudado para o caso de robôs manipuladores industriais. O uso das técnicas desenvolvidas e a avaliação de outras (como o uso de pseudoinversa ponderada com pesos variáveis e geradores de transições suaves) é um potencial campo de estudos.

No uso dos *frameworks*, observou-se várias situações em que podem ser aplicadas técnicas de otimização, como no caso dos ganhos da realimentação em malha fechada, na definição de pesos para a matriz de ponderação da pseudoinversa e até para definir estratégias de chaveamento de estados. Uma avaliação da aplicação dessas técnicas e da possibilidade de uso tanto em simulações quanto em aplicações reais é de interesse futuro.

A dinâmica e controle dos UUV é um assunto bastante estudado na literatura. Porém, não se encontraram referências usando a abordagem de helicoides nessa linha de pesquisa, embora existam trabalhos sobre dinâmica e controle usando helicoides na robótica industrial. Assim, considera-se este um tema relevante de pesquisas futuras, visando o desenvolvimento de uma modelagem com um enfoque unificado do sistema subaquático. A questão da dinâmica também é importante no estudo da interação entre o UVMS e o meio, especificação de tarefas com aplicação de esforços além de movimento e dos efeitos causados pelo ambiente imerso.

Por fim, um trabalho futuro de grande importância é a implementação dessa sistemática em UVMS reais a fim de fazer a avaliação experimental dessas técnicas. Além de validar as simulações, a experimentação em sistemas reais demandará o aprimoramento de *frameworks* para implementar essas técnicas e uma reavaliação de requisitos de projeto considerando casos reais. Para a implementação experimental, são consideradas a possibilidade de acionamento indireto do UVMS (a estratégia de planejamento de movimento substituiria os sinais gerados pelo operador de um ROV em sua estação de trabalho) ou a integração das estratégias de planejamento de movimento ao sistema de um UVMS. Para isso, seria necessário que este tivesse sido construído com características de configurabilidade total de software e

possibilidade de operação em diferentes níveis de autonomia). Foram elaborados projetos informacionais e conceituais de UUV para esse fim, que podem ser considerados uma contribuição secundária desta tese.

8.3 CONSIDERAÇÕES FINAIS

Os veículos subaquáticos não tripulados e suas aplicações formam uma área com grandes potenciais para pesquisa, ainda mais com a motivação do uso cada vez maior desses sistemas em diversas aplicações no ambiente marinho. A exploração de petróleo e gás *offshore* nacional, particularmente no caso do pré-sal, está fazendo com que muitas empresas de tecnologia subaquática venham para o Brasil. Essas empresas têm vários desafios tecnológicos a serem resolvidos para seus UUV, particularmente para os sistemas de intervenção subaquática. Isso motivará cada vez mais desenvolvimentos nesse campo, com oportunidades tanto para a pesquisa quanto para a indústria nacionais.

O presente trabalho contribuiu para esse desenvolvimento nacional, através da apresentação de inovações para o planejamento de movimento desses sistemas e de motivação para pesquisas futuras. Além disso, observa-se que os desenvolvimentos deste trabalho, em particular o *framework* Kast, podem ser empregados para outros sistemas veículo-manipulador (aéreos e terrestres) e mesmo para sistemas compostos por manipuladores apenas.

REFERÊNCIAS BIBLIOGRÁFICAS

- Ananthkrishnan, P.; Decron, S. *Dynamics of Small and Mini-Autonomous underwater Vehicles: Part I. Analysis and Simulation for Midwater Application*. Boca Raton, Florida, 2000.
- Antonelli, G. Open control problems in underwater robotics. In: IEEE. *Proc. Fourth International Workshop on RoMoCo'04 Robot Motion and Control*. Puszczkovo, 2004. p. 219–229.
- Antonelli, G. *Underwater Robots: Motion and Force Control of Vehicle-manipulator Systems*. 2. ed. Berlim: Springer, 2006. 268 p. (Springer Tracts in Advanced Robotics, v. 2).
- Antonelli, G.; Fossen, T.; Yoerger, D. Underwater Robotics. In: Siciliano, B. and Khatib, O. (Ed.). *Springer Handbook of Robotics*. Heidelberg: Springer, 2008, (Springer Tracts in Advanced Robotics). cap. 43, p. 987–1008.
- Bang-Jensen, J.; Gutin, G. *Digraphs: Theory, Algorithms and Applications*. Berlim: Springer, 2007. 754 p. (Springer Monographs in Mathematics).
- Barros, E.; Soares, F. Desenvolvimento de um robô submarino de baixo custo. In: *Congresso Brasileiro de Automática*. Natal: [s.n.], 2002. v. 14, p. 2121–2126.
- Benjamin, M. *Software Architecture and Strategic Plans for Undersea Cooperative Cueing and Intervention*. Washington, D.C., 2007.
- Bennett, P. et al. Probing the Limits of Human Deep Diving [and Discussion]. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, The Royal Society, London, v. 304, n. 1118, p. 105–117, 1984.
- Beshenov, L. *Maxima, a Computer Algebra System*. 2011. <http://maxima.sourceforge.net>. Acessado em 11/11/2011.
- Bishop, B. On the use of redundant manipulator techniques for control of platoons of cooperating robotic vehicles. *IEEE Transactions on Systems, Man and Cybernetics*, v. 33, n. 5, p. 608–615, 2003.

Bishop, B. Swarm-based object manipulation using redundant manipulator analogs. In: IEEE. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Pasadena, 2008. p. 1495–1500.

Bishop, B.; Stilwell, D. On the application of redundant manipulator techniques to the control of platoons of autonomous vehicles. In: IEEE. *Proceedings of the 2001 IEEE International Conference on Control Applications*. México, 2001. p. 823–828.

Booch, G.; Rumbaugh, J.; Jacobson, I. *The unified modeling language user guide*. Reading: Addison Wesley Longman, 1999. ISBN 0201571684.

Borges, L. E. *Python Para Desenvolvedores*. Rio de Janeiro: [s.n.], 2010. 360 p. ISBN 978-85-909451-1-6.

Botelho, S. et al. Lambdari: um robô subaquático autônomo. *Simpósio Brasileiro De Automação Inteligente-VI SBAI*, Bauru, 2003.

British Petroleum. *Gulf of Mexico Restoration - Remotely Operated Vehicles*. 2010. <http://www.bp.com/sectiongenericarticle800.do?categoryId=9036600&contentId=7067604>. Acessado em 05/06/2011.

Button, R. et al. *A Survey of Missions for Unmanned Undersea Vehicles*. Arlington: RAND Corporation, 2009. 223 p.

Campos, A. *Cinematika Diferencial de Manipuladores Empregando Cadeias Virtuais*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2004.

Campos, A.; Guenther, R.; Martins, D. Differential kinematics of serial manipulators using virtual chains. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 27, n. 4, p. 345–356, 2005.

Ceccarelli, M. Screw axis defined by Giulio Mozzi in 1763 and early studies on helicoidal motion. *Mechanism and Machine Theory*, Elsevier, v. 35, n. 6, p. 761–770, 2000.

Centeno, M. *Rovfurg-II: projeto e construção de um veículo subaquático não tripulado de baixo custo*. Dissertação (Dissertação de mestrado) — FURG - Pós-Graduação em Engenharia Oceânica, Rio Grande, 2007.

Chiaverini, S.; Oriolo, G.; Walker, I. Kinematically Redundant Manipulators. In: _____. *Springer Handbook of Robotics*. Heidelberg: Springer, 2008. cap. 11, p. 245–268.

Cui, Y.; Podder, T.; Sarkar, N. Impedance control of underwater vehicle-manipulator systems (UVMS). In: IEEE. *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999. IROS'99. Proceedings*. South Korea, 1999. v. 1, p. 148–153.

Cunha, J. *Projeto e Estudo de Simulação de um Sistema de Controle a Estrutura Variável de um Veículo Submarino de Operação Remota*. Dissertação (Dissertação de mestrado) — UFRJ - COPPE - Pós-Graduação em Engenharia Elétrica, Rio de Janeiro, 1992.

Cybernetix. *Présentation - ingénierie des systèmes robotiques - Cybernetix*. 2008. <http://www.cybernetix.fr/Presentation,90>. Acessado em 05/06/2011.

Dai, J. An historical review of the theoretical development of rigid body displacements from Rodrigues parameters to the finite twist. *Mechanism and Machine Theory*, Elsevier, v. 41, n. 1, p. 41–52, 2006.

Davies, T. Kirchhoff's circulation law applied to multi-loop kinematic chains. *Mechanism and Machine Theory*, v. 16, n. 3, p. 171–183, 1981.

Davison, J.; Hunt, K. *Robots and screw Theory: Applications of Kinematics and Statics to Robotics*. [S.l.]: Oxford, 2004.

De Novi, G. et al. New Approach for a Reconfigurable Autonomous Underwater Vehicle for Intervention. *Aerospace and Electronic Systems Magazine, IEEE*, v. 25, n. 11, p. 32–36, 2010.

De Souza, E.; Maruyama, N. Intelligent UUVs: Some issues on ROV dynamic positioning. *IEEE Transactions on Aerospace and Electronic Systems*, IEEE, v. 43, n. 1, p. 214–226, 2007.

de Wit, C.; Diaz, O.; Perrier, M. Nonlinear control of an underwater vehicle/manipulator with composite dynamics. *IEEE Transactions on Control Systems Technology*, v. 8, n. 6, p. 948–960, 2000.

Deitel, H.; Deitel, P. *Java: Como Programar*. 4. ed. Porto Alegre: Bookman, 2004. 1386 p. ISBN 85-363-0123-6.

Diankov, R. *Automated Construction of Robotic Manipulation Programs*. Tese (Doutorado) — Carnegie Mellon University, Robotics

Institute, 2010. <http://www.programmingvision.com/rosen%20diankov%20thesis.pdf>.

Diestel, R. *Graph Theory*. 3. ed. Heidelberg: Springer, 2005. 415 p. (Graduate Texts in Mathematics, v. 173).

Digiteo. *Scilab website*. 2011. <http://www.scilab.org>. Acessado em 02/02/2011.

Dominguez, R. *Simulação e Controle de um Veículo Submarino de Operação Remota*. Dissertação (Dissertação de mestrado) — UFRJ - COPPE - Pós-Graduação em Engenharia Elétrica, Rio de Janeiro, 1989.

Douglas-Westwood. *Douglas-Westwood - Energy Business Advisors*. 2011. <http://www.dw-1.com>. Acessado em 06/06/2011.

Dourado, A. *Cinemática de Robôs cooperativos*. Dissertação (Dissertação de mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2005.

Décio, O. *XML – Guia de Consulta Rápida*. [S.l.]: Novatec, 2000. 96 p. ISBN 85-85184-86-8.

Eaton, J. *Octave*. 2011. <http://www.gnu.org/software/octave>. Acessado em 02/02/2011.

El-Hawary, F. *The Ocean Engineering Handbook*. Boca Raton: CRC, 2001. 416 p. ISBN 0-8493-8598-9.

Featherstone, R. *Rigid Body Dynamics Algorithms*. 2. ed. New York: Springer, 2008. 278 p.

Floriani, B.; Dias, A.; Rocha, C. Aspectos Metodológicos No Projeto Informacional e Conceitual de Um Veículo Remotamente Operado Subaquático. In: IMECHE. *Anais do VI Congresso Luso-Moçambicano de Engenharia - CLME 2011*. Maputo: Edições Inegi, 2011.

Fontan, D. *Implementação da Cinemática Inversa de Robôs Redundantes Operando em Ambientes Confinados no Projeto Roboturb*. Dissertação (Dissertação de mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2007.

Fossen, T. *Guidance and Control of Ocean Vehicles*. [S.l.]: Wiley, 1994. 494 p.

Fowler, M. *UML Essencial*. 3. ed. Porto Alegre: Bookman, 2004. 160 p. ISBN 85-363-0454-5.

Fraisse, P. et al. Position/Force Control of an Underwater Vehicle Equipped with a Robotic Manipulator. In: . [S.l.: s.n.], 2000.

Guenther, R. et al. A New Integration Method for Differential Inverse Kinematics of Closed-Chain Robots. In: *ABCMSymposium Series in Mechatronics*. Rio de Janeiro, Brasil: ABCM, 2008, (ABCMSymposium Series, v. 3). p. 225–235.

Hsu, L. et al. Avaliação Experimental da Modelagem e Simulação da Dinâmica de Um Veículo Submarino de Operação Remota. *Controle e Automação*, v. 11, n. 2, p. 82–93, 2000.

Hunt, K. Don't cross-thread the screw. In: University of Cambridge - Trinity College. *A Symposium Commemorating The Legacy, Works and Life of Sir Robert Stawell Ball Upon the 100th Anniversary of A Treatise on The Theory of Screws*. Cambridge, United Kingdom: Cambridge University Press, 2000. p. 1–37.

Hunter, J.; Dale, D.; Droettboom, M. *matplotlib: python plotting*. 2011. <http://matplotlib.sourceforge.net>. Acessado em 05/12/2011.

Inovação Tecnológica. Submarino robótico atinge ponto mais fundo do oceano. *Inovação Tecnológica*, 2009. <http://www.inovacaotecnologica.com.br/noticias/artigo=submarino-robotico-atinge-ponto-mais-profundo-oceano>. Acessado em 01/08/2009.

Kinsey, J.; Eustice, R.; Whitcomb, L. A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges. In: IFAC. *IFAC Conference of Manoeuvring and Control of Marine Craft*. Lisboa, 2006.

Kohnen, W. 2007 MTS Overview of Manned Underwater Vehicle Activity. *Marine Technology Society Journal*, Marine Technology Society, v. 42, n. 1, p. 26–37, 2008.

Kongsberg. *Remus 6000 - Autonomous Underwater Vehicle - Kongsberg Maritime*. 2011. <http://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/481519DA1B0207CDC12574B0002A8451?OpenDocument>. Acessado em 05/06/2011.

Kuhn, V. *Controle Automático de um Veículo de Inspeção Subaquática Utilizando Sensoriamento de Baixo Custo*. Dissertação (Dissertação de mestrado) — FURG - Pós-Graduação em Engenharia Oceânica, Rio Grande, 2011.

Lane, D. et al. The AMADEUS dextrous subsea hand: design, modeling, and sensor processing. *IEEE Journal of Oceanic engineering*, v. 24, n. 1, p. 96–111, 1999.

Langtangen, H. P. *A Primer on Scientific Programming with Python*. Berlin: Springer-Verlag, 2009. 695 p. (Texts in Computational Science and Engineering). ISBN 978-3-642-02475-7.

Laus, L.; Simoni, R.; Martins, D. Progressive dynamic analysis of serial robots based on screw theory. In: *Proceedings of the 20th International Congress of Mechanical Engineering - COBEM 2009*. Gramado: [s.n.], 2009.

Laus, L.; Simoni, R.; Martins, D. Progressive Dynamic Analysis of Serial Robots Based on Screw Theory: An Extension to the Theory. In: *11th Pan-American Congress of Applied Mechanics - PACAM XI*. Foz do Iguaçu: [s.n.], 2010.

Leff, L.; Plushnick-Masti, R. Underwater Robots Attack Spill Like Superman. *msnbc.com*, Online, 2010. <http://www.msnbc.msn.com/id/37913126/ns/disaster%20in%20the%20gulf/t/underwater-robots-attack-spill-superman>. Acessado em 05/06/2011.

Lundh, F. *ElementTree Overview*. 2011. <http://effbot.org/zone/element-index.htm>. Acessado em 05/12/2011.

Marani, G.; Choi, S.; Yuh, J. Underwater autonomous manipulation for intervention missions AUVs. *Ocean Engineering*, Elsevier, v. 36, n. 1, p. 15–23, 2009.

Mathworks. *Matlab - The Language of Technical Computing*. 2011. <http://www.mathworks.com/products/matlab>. Acessado em 02/02/2011.

McLain, T.; Rock, S.; Lee, M. Experiments in the coordination of underwater manipulator and vehicle control. In: IEEE. *Oceans '95 Proceedings*. San Diego, 1995. v. 2, p. 1208–1215.

Metsker, S. J. *Padrões de Projeto em Java*. Porto Alegre: Bookman, 2004. 407 p. ISBN 85-363-0411-1.

Moraes, C. *Rovfurg-I: projeto e construção de um veículo subaquático não tripulado de baixo custo*. Dissertação (Dissertação de mestrado) — FURG - Pós-Graduação em Engenharia Oceânica, Rio Grande, 2005.

MTS. ROV Background. *MTS Remotely Operated Vehicle Committee*, Online, 2006. <http://www.rov.org/info.cfm>. Acessado em 01/08/2009.

NetworkX. *Overview – NetworkX 1.6 documentation*. 2011. <http://networkx.lanl.gov>. Acessado em 05/12/2011.

Newman, P. *The MOOS - Cross Platform Software for Robotics Research*. Oxford, 2009. Website. <http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php>. Acessado em 02/11/2009.

Numpy. *Scientific Computing Tools for Python – Numpy*. 2011. <http://numpy.scipy.org>. Acessado em 05/12/2011.

Oceaneering. *Oceaneering - Remotely Operated Vehicles (ROVs)*. 2009. <http://www.oceaneering.com/rovs>. Acessado em 05/06/2011.

Offshore Engineering. Oceaneering ROV Sets Depth Record Offshore India. *Offshore Energy Today.com*, Online, 2011. <http://www.offshoreenergytoday.com/oceaneering-rov-sets-depth-record-offshore-india>. Acessado em 02/06/2011.

Offshore Engineering. Transocean Sets New World Record in Deepwater Drilling Depth. *Offshore Energy Today.com*, Online, 2011. <http://www.offshoreenergytoday.com/transocean-sets-new-world-record-in-deepwater-drilling-depth>. Acessado em 02/06/2011.

Oliveira, A. S. d. *Análise cinemática via quatérnios duais aplicada a um sistema veículo-manipulador subaquático*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2011.

Oracle. *Java.com: Java + you*. 2011. <http://java.com/pt%20BR>. Acessado em 02/02/2011.

Padir, T. Kinematic redundancy resolution for two cooperating underwater vehicles with on-board manipulators. In: IEEE. *Proc. IEEE International Conference on Systems, Man and Cybernetics*. Waikoloa, 2005. v. 4, p. 3137–3142.

Padir, T.; Koivo, A. Modeling of two underwater vehicles with manipulators on-board. In: IEEE. *Proc. IEEE International Conference on Systems, Man and Cybernetics*. Washington, 2003. v. 2, p. 1359–1364.

Paschoa, C. AUVs in the Brazilian O&G industry. *Sea Currents - Underwater Tech and Ocean Sciences*, Online, 2010. <http://news.seadiscovery.com/post/2010/09/29/aUvs-the-brazilian-industry.aspx>. Acessado em 25/05/2011.

Petrobras. *Cada Vez Mais Fundo*. 2011. <http://www.petrobras.com.br/minisite/presal/pt/cada-vez-mais-fundo>. Acessado em 04/06/2011.

Petrobras. Descoberta de óleo leve na Bacia de Campos. *Petrobras - Relação com Investidores*, Online, 2011. <http://www.petrobras.com.br/ri/Show.aspx?id%20materia=ZhgyJ5a4fmuAoBLQChkSHw==>. Acessado em 02/06/2011.

Petrobras. Inovar Para Crescer: Entrevista com Gerente Executivo do CENPES, Carlos Tadeu Fraga. *Petrobras - Fatos e Dados*, Online, 2011. <http://fatosedados.blogspotpetrobras.com.br/2011/05/06/inovar-para-crescer-entrevista-com-gerente-executivo-do-cenpes-carlos-tadeu-fraga>. Acessado em 02/06/2011.

Phillips, J. *Freedom in Machinery*. Cambridge: Cambridge University Press, 2007. 448 p.

Phoenix. *Phoenix International Website*. 2011. <http://www.phnx-international.us>. Acessado em 25/05/2011.

Pressman, R. *Engenharia de Software*. São Paulo: Makron Books, 1995. 1056 p. ISBN 85-346-0237-9.

Python.org. *Python Programming Language - Official Website*. 2011. Available in <http://www.python.org>. Acessado em 02/02/2011.

Raybaut, P. *pythonxy: Scientific-oriented Python Distribution based on Qt and Spyder*. 2011. <http://www.pythonxy.com>. Acessado em 23/08/2011.

- Ribeiro, L. P. G.; Guenther, R.; Martins, D. Screw-Based Relative Jacobian for Manipulators Cooperating in a Task. In: *ABCMSymposium Series in Mechatronics*. Rio de Janeiro: ABCM, 2008, (ABCMSymposium Series, v. 3). p. 276–285.
- Ribeiro, L. P. G.; Martins, D. Screw-Based Relative Jacobian For Manipulators Cooperating In A Task Using Assur Virtual Chains. In: *ABCMSymposium Series in Mechatronics*. Rio de Janeiro: ABCM, 2010, (ABCMSymposium Series, v. 4). p. 729–738.
- Richardson, I.; Woodward, N.; Billingham, J. Deepwater Welding for Installation and Repair: A Viable Technology? In: The International Society of Offshore and Polar Engineers. *Int. Conf. On Offshore and Polar Engineering (ISOPE)*. Kitakyushu, 2002.
- Ridao, P. and Sanz, P. J. and Oliver, J. *Reconfigurable AUV for Intervention*. 2011. <http://www.irs.uji.es/rauvi/abouttheproject.html>. Acessado em 06/06/2011.
- Rocha, C.; Dias, A. Evitamento de Colisão de Robôs Através do Método das Restrições Cinemáticas. In: ABCM - Associação Brasileira de Ciências Mecânicas. *Proceedings do VI Congresso Nacional de Engenharia Mecânica - CONEM*. Campina Grande, 2010.
- Rocha, C.; Dias, A. Sistemas Subaquáticos de Intervenção Autônoma: Cenários e Desafios. In: Instituto de Pesquisas da Marinha. *Proceedings do VI Simpósio Brasileiro de Engenharia Inercial*. Rio de Janeiro: Instituto de Pesquisas da Marinha, 2010.
- Rocha, C. et al. Obstacle and Collision Avoidance in Unstructured and Time-Varying Environments: A Screw Theory Approach for Manipulators. In: Teesside University. *Proceedings of the 19th International Conference on Flexible Automation and Intelligent Manufacturing*. Middlesbrough: Gemini, 2009. p. 436–443.
- Rocha, C. R.; Tonetto, C.; Dias, A. A framework for kinematic modeling of cooperative robotic systems based on screw theory. In: Universidade Federal do Rio Grande do Norte. *Proceedings of the 21th International Congress of Mechanical Engineering - COBEM 2011*. Natal, 2011.
- Rocha, C. R.; Tonetto, C. P.; Dias, A. A comparison between the Denavit-Hartenberg and the screw-based methods used in kinematic modeling of robot manipulators. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 27, n. 4, p. 723–728, 2011.

Sagara, S. et al. Experiments on a floating underwater robot with a two-link manipulator. *Artificial Life and Robotics*, Springer, v. 5, n. 4, p. 215–219, 2001.

Santos, C. *Movimento Coordenado de Sistemas Veículo-Manipulador Submarinos Utilizando Técnicas de Inteligência Artificial e Sistemas Híbridos*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2006.

Santos, C. et al. Virtual kinematic chains to solve the underwater vehicle-manipulator systems redundancy. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, scielo, v. 28, p. 354 – 361, 2006.

Sarkar, N.; Podder, T. Coordinated motion planning and control of autonomous underwater vehicle-manipulator systems subject to drag optimization. *IEEE Journal of Oceanic Engineering*, v. 26, n. 2, p. 228–239, 2001.

Schjølberg, I.; Fossen, T. Modeling and control of underwater vehicle-manipulator systems. In: *Proceedings of the 3rd Conference on Marine Craft Maneuvering and Control*. Southampton, UK: [s.n.], 1994. p. 45–57.

Schmiegelow, J. *O Planeta Azul: Uma Introdução Às Ciências Marinhas*. Rio de Janeiro: Interciência, 2004. 202 p.

Scipy. *Scipy*. 2011. <http://www.scipy.org/SciPy>. Acessado em 05/12/2011.

Siciliano, B.; Khatib, O. (Ed.). *Springer Handbook of Robotics*. Heildeberg: Springer, 2008. 1375 p. (Springer Tracts in Advanced Robotics).

Siciliano, B. et al. *Robotics: Modelling, Planning and Control*. London: Springer, 2009. 632 p. (Advanced Textbooks in Control and Signal Processing).

Simas, H. *Planejamento de Trajetórias e Evitamento de Colisão em Tarefas de Manipuladores Redundantes Operando em Ambientes Confinados*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2008.

Simas, H.; Fontan, D.; Martins, D. Smooth Transitions in Trajectory Profiles for Redundant Robots Performing Secondary Tasks. In:

Universidade Federal do Rio Grande do Norte. *Proceedings of the 21th International Congress of Mechanical Engineering - COBEM 2011*. Natal, 2011.

Simas, H. et al. A new method to solve robot inverse kinematics using Assur virtual chains. *Robotica*, Cambridge Univ Press, v. 27, n. 07, p. 1017–1026, 2009.

SNAME. *Nomenclature for Treating The Motion of a Submerged Body Through a Fluid*. New Jersey, 1950.

Soylu, S.; Buckham, B.; Podhorodeski, R. Redundancy Resolution For Underwater Mobile Manipulators. *Ocean Engineering*, Elsevier, v. 37, p. 325–343, 2010.

Stone Aerospace. *ENDURANCE*. 2012. <http://www.stoneaerospace.com/products-pages/products-ENDURANCE.php>. Acessado em 15/04/2012.

Stroustrup, B. *Stroustrup: C++*. 2011. <http://www2.research.att.com/~bs/C++.html>. Acessado em 02/02/2011.

Tarn, T.; Yang, S. Modeling and control for underwater robotic manipulators – an example. In: IEEE. *Proceedings of 1997 IEEE International Conference on Robotics and Automation*. Albuquerque, 1997. v. 3, p. 2166–2171.

Tavares, A. *Um Estudo Sobre A Modelagem e O Controle De Veículos Subaquáticos Não Tripulados*. Dissertação (Dissertação de mestrado) — FURG - Pós-Graduação em Engenharia Oceânica, Rio Grande, 2003.

Terra. Conheça o Remora 6000, o Robô das Buscas do AF447. *Terra Notícias*, Online, 2011. <http://www.terra.com.br/noticias/infograficos/remora6000>. Acessado em 25/05/2011.

Tonetto, C.; Rocha, C.; Dias, A. Simulation of Multi-Robot Cooperative Systems Programming Based on a Three Environment Definition. In: IMECHE. *Proceedings of the 12th Mechatronics Forum Biennial International Conference*. Zurich: IWF - Institute of Machine Tools and Manufacturing, 2010. v. 2, p. 186–193.

Tonetto, C. P. *Uma Sistematização da Modelagem e Programação Cinemática de Sistemas Robóticos Cooperativos Para a Realização de Tarefas*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2011.

Tonjum, S.; Peterson, R.; Florio, J. Norwegian Deep Diving Trials [and Discussion]. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, The Royal Society, London, v. 304, n. 1118, p. 143–149, 1984.

Tsai, L. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. New York: Wiley-Interscience, 1999.

Tsai, L. W. *Mechanism Design: Enumeration of Kinematic Structures According to Function*. Boca Raton: CRC-Press, 2000.

UNESCO (Ed.). *Water in a Changing World: The United Nations World Water Development Report 3*. Paris: UNESCO, 2009. 430 p.

Valavanis, K. et al. Control architectures for autonomous underwater vehicles. *IEEE Control Systems Magazine*, v. 17, n. 6, p. 48–64, 1997.

W3C. *Extensible Markup Language(XML)*. 2011. <http://www.w3.org/XML>. Acessado em 09/02/2011.

Waldron, K.; Schmedeler, J. Kinematics. In: Siciliano, B.; Khatib, O. (Ed.). *Springer Handbook of Robotics*. Heidelberg: Springer, 2008, (Springer Tracts in Advanced Robotics). cap. 01, p. 9–34.

Whitcomb, L. Underwater Robotics: Out of The Research Laboratory and Into The Field. In: IEEE. *Proceedings of ICRA '00*. San Francisco, 2000. v. 1, p. 709–716.

Xu, G. et al. Motion Control and Computer Simulation for Underwater Vehicle-Manipulator Systems. In: *International Conference on Mechatronics and Automation - ICMA 2007*. Heilongjiang: [s.n.], 2007. p. 1368–1373.

Yuh, J. Modeling and control of underwater robotic vehicles. *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, v. 20, n. 6, p. 1475–1483, 1990.

Yuh, J. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, v. 8, n. 1, p. 7–24, 2000.

Yuh, J. et al. Design of a semi-autonomous underwater vehicle for intervention missions (SAUVIM). In: *Proceedings of the 1998 IEEE International Symposium on Underwater Technologies*. Tokyo: [s.n.], 1998. p. 63–68.

Ziviani, N. *Projeto de algoritmos com implementação em Java e C++*. São Paulo: Thomson Learning, 2006. 642 p.

**APÊNDICE A – Fundamentos da Análise Cinemática Por
Helicoides**

Este apêndice consiste em uma revisão das teorias e ferramentas utilizadas na análise cinemática por helicoides. Ele inicia pela representação por helicoides do movimento de corpos rígidos, seguida pela sua extensão para cadeias cinemáticas, onde a teoria dos grafos é um instrumento importante. As ferramentas usadas para a análise das cadeias cinemáticas como o método de Davies e as cadeias virtuais de Assur são apresentadas a seguir. O uso do método das restrições cinemáticas para o planejamento de movimento encerra o apêndice.

A.1 O MOVIMENTO DE CORPOS RÍGIDOS

A teoria dos helicoides é uma forma de se estudar o movimento geral de um corpo rígido no espaço, bem como o movimento relativo de corpos rígidos vinculados, formando cadeias cinemáticas. Suas origens remontam aos estudos do deslocamento helicoidal instantâneo de Mozzi em 1763 e do deslocamento finito geral de corpos rígidos de Chasles em 1830. Ball sistematizou a teoria em 1900, sendo posteriormente aplicada na análise cinemática por Hunt, Phillips, Roth e Tsai, entre outros. A evolução dos estudos do movimento helicoidal e suas aplicações é apresentada nas revisões históricas de Ceccarelli e Dai (Ceccarelli, 2000; Dai, 2006).

Um helicóide (do inglês *screw*) é um ente geométrico, como um ponto, uma reta ou um plano. Ele é formado por um *eixo* (ou reta direcionada) e por uma grandeza escalar denominada *passo*. Assim, essa entidade pode representar ao mesmo tempo uma quantidade rotacional em torno do seu eixo e uma grandeza translacional paralela a este eixo, relacionando-as pelo valor do passo (Hunt, 2000; Davison; Hunt, 2004). A teoria dos helicoides associa um significado físico a esta entidade puramente geométrica, utilizando-a para expressar tanto velocidades (angular e linear) quanto esforços (forças e momentos) em uma única representação, o que o torna conveniente para estudos em cinemática e estática (Dai, 2006).

Uma notação bastante usada para expressar helicoides é baseada na definição de uma reta pelas coordenadas de Plücker, como ilustrado na Figura 79. Seja \mathbf{S} um vetor que representa a direção e a magnitude do eixo do helicóide. A posição do eixo em relação à origem \mathbf{O} de um sistema de coordenadas de referência é descrita pelo *momento* de \mathbf{S} , definido como o produto vetorial entre o vetor posição \mathbf{s}_0 de um ponto qualquer do eixo do helicóide e o vetor \mathbf{S} .

As coordenadas de Plücker são descritas na Equação A.1. Os

três primeiros componentes correspondem ao vetor \mathbf{S} do eixo da reta, enquanto os três componentes restantes são o seu momento. Observe-se que as coordenadas de Plücker não formam um vetor, e que \mathbf{S} e seu momento são ortogonais (Hunt, 2000).

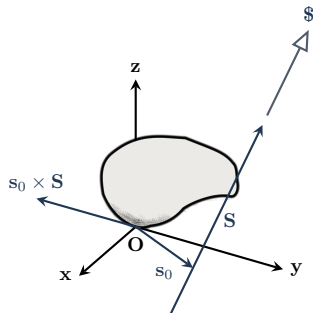


Figura 79. Definição geométrica das coordenadas de Plücker

$$\mathcal{S} = \begin{bmatrix} \mathbf{S} \\ \mathbf{s}_o \times \mathbf{S} \end{bmatrix} = [\mathcal{L} \quad \mathcal{M} \quad \mathcal{N} \mid \mathcal{P} \quad \mathcal{Q} \quad \mathcal{R}]^T \quad (\text{A.1})$$

A translação paralela ao eixo do helicóide relaciona-se à rotação em torno de \mathbf{S} pelo passo escalar h . Esta translação é adicionada às coordenadas de Plücker nas componentes de momento, resultando na Equação A.2, onde $\mathcal{P}^* = \mathcal{P} + h\mathcal{L}$, $\mathcal{Q}^* = \mathcal{Q} + h\mathcal{M}$ e $\mathcal{R}^* = \mathcal{R} + h\mathcal{N}$. A representação geométrica é ilustrada na Figura 80.

$$\mathcal{S} = \begin{bmatrix} \mathbf{S} \\ \mathbf{s}_o \times \mathbf{S} + h\mathbf{S} \end{bmatrix} = [\mathcal{L} \quad \mathcal{M} \quad \mathcal{N} \mid \mathcal{P}^* \quad \mathcal{Q}^* \quad \mathcal{R}^*]^T \quad (\text{A.2})$$

Um helicóide pode ser decomposto em um *helicóide normalizado* $\hat{\mathcal{S}}$ e uma magnitude Ψ , como na Equação A.3,

$$\mathcal{S} = \hat{\mathcal{S}}\Psi = \begin{bmatrix} \mathbf{s} \\ \mathbf{s}_o \times \mathbf{s} + h\mathbf{s} \end{bmatrix} \Psi \quad (\text{A.3})$$

onde \mathbf{s} é o vetor unitário de \mathbf{S} . Caso o helicóide normalizado represente apenas rotação ($h = 0$) ele se reduz à forma da Equação A.4. Para uma translação pura assume-se $h = \infty$ e o helicóide normalizado é expresso

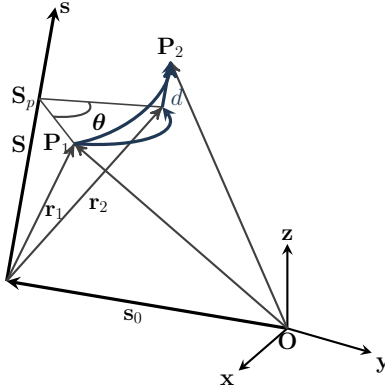


Figura 81. Deslocamento helicoidal e parâmetros de Rodrigues

$$\mathbf{R}(\theta) = \begin{bmatrix} c_\theta + s_x^2(1 - c_\theta) & s_y s_x(1 - c_\theta) - s_z s_\theta & s_z s_x(1 - c_\theta) - s_y s_\theta \\ s_y s_x(1 - c_\theta) - s_z s_\theta & c_\theta + s_y^2(1 - c_\theta) & s_y s_z(1 - c_\theta) - s_x s_\theta \\ s_z s_x(1 - c_\theta) - s_y s_\theta & s_y s_z(1 - c_\theta) - s_x s_\theta & c_\theta + s_z^2(1 - c_\theta) \end{bmatrix} \quad (\text{A.7})$$

$$\mathbf{p}(d, \theta) = ds + [\mathbf{I} - \mathbf{R}(\theta)] \mathbf{s}_0 \quad (\text{A.8})$$

Um corpo rígido pode sofrer mais de um deslocamento helicoidal, e o deslocamento total resultante pode ser representado por um único deslocamento helicoidal equivalente. Este é determinado pela premultiplicação das matrizes de transformação homogênea dos deslocamentos helicoidais originais.

A.1.2 Cinemática Diferencial

Segundo o Teorema de Mozzi, o movimento instantâneo dos pontos de um corpo pode ser decomposto em uma rotação diferencial em torno de um eixo e uma translação diferencial paralela a este eixo. Este movimento helicoidal pode ser representado por um helicóide de velocidade denominado *heligiro* (do inglês *twist*) (Campos; Guenther; Martins, 2005).

Sendo um helicóide, o heligiro é composto por uma parte rotacional e uma translacional como $\mathcal{S} = [\boldsymbol{\omega}^T; \mathbf{v}_p^T]^T$, onde $\boldsymbol{\omega}$ é a velocidade

angular do corpo rígido e \mathbf{v}_p é a velocidade linear de um ponto instantaneamente coincidente com a origem \mathbf{O} que se move solidário ao corpo. A Figura 82 ilustra essa definição. \mathbf{v}_p tem um componente normal ao eixo do heligiro ($\mathbf{s}_0 \times \boldsymbol{\omega}$) e um componente paralelo a ele ($h\boldsymbol{\omega}$).

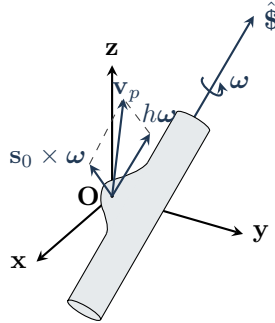


Figura 82. Heligiro em um corpo rígido

Um heligiro pode ser decomposto em um helicóide normalizado e uma magnitude, como descrito na Equação A.9, onde \dot{q} é a magnitude da velocidade. Os casos particulares anteriormente definidos para helicóides encontram significado físico nas juntas rotativas e prismáticas comumente empregadas em robótica. Para as primeiras, \dot{q} corresponde à velocidade angular ω (assim como o caso helicoidal geral). Nas últimas, \dot{q} corresponde à velocidade linear v_p .

$$\mathbb{S} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{s}_0 \times \boldsymbol{\omega} + h\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ \mathbf{s}_0 \times \mathbf{s} + h\mathbf{s} \end{bmatrix} \dot{q} = \hat{\mathbb{S}} \dot{q} \quad (\text{A.9})$$

A.2 O MOVIMENTO DE CADEIAS CINEMÁTICAS

A abordagem baseada em helicóides para modelagem cinemática de cadeias é uma alternativa à baseada na convenção de Denavit-Hartenberg, muito usada na literatura de robótica. Em relação a esta convenção, a abordagem baseada em helicóides apresenta algumas vantagens, como o fato de poder ser usada de modo uniforme para cadeias abertas e fechadas e ter flexibilidade na escolha dos sistemas de coordenadas de referência empregados (Rocha; Tonetto; Dias, 2011b).

A.2.1 Cinemática de Posição de Uma Cadeia

Em uma cadeia cinemática, elos (tratados aqui como corpos rígidos) são vinculados entre si através de juntas. O movimento de um elo i da cadeia em relação ao seu predecessor $i - 1$ é função do tipo da junta i que os conecta. Esse movimento pode ser descrito por um heligi-ro \mathcal{S}_i . Com base nesse heligi-ro, é possível definir a matriz de transformação homogênea que representa a postura do elo i em relação ao elo $i - 1$.

A postura de um elo e relativa a um elo b (por exemplo, a postura do efetuador final de um manipulador serial em relação à sua base) é então definida pelos deslocamentos helicoidais realizados pelas juntas da subcadeia definida entre b e e . O deslocamento total do elo e é obtido pela premultiplicação das matrizes de transformação ${}^{i-1}\mathbf{A}_i$ determinadas como na Equação A.6 a partir dos parâmetros de Rodrigues do deslocamento i . Este resultado é expresso na Equação A.10.

$${}^b\mathbf{A}_e = {}^b\mathbf{A}_1 {}^1\mathbf{A}_2 \cdots {}^{n-1}\mathbf{A}_n {}^n\mathbf{A}_e \quad (\text{A.10})$$

Esse processo pode ser visto como se o corpo rígido do elo e sofresse vários deslocamentos em sequência segundo os helicoides da primeira à última junta da subcadeia $b - e$. Por isso, ele é denominado *método dos deslocamentos helicoidais sucessivos* (Tsai, 1999). Ele pode ser sintetizado nos seguintes passos (Rocha; Tonetto; Dias, 2011b):

- Escolher um sistema de coordenadas fixo de referência a partir dos quais os helicoides serão definidos;
- Definir uma configuração de referência para a cadeia cinemática, a partir da qual os parâmetros \mathbf{s} e \mathbf{s}_0 dos helicoides serão determinados;
- Para cada junta, identificar os parâmetros de Rodrigues e sua variável de junta;
- Determinar a matriz de transformação homogênea relativa ao deslocamento de cada junta;
- Determinar a matriz de transformação homogênea resultante de todos os deslocamentos através da Equação A.10. A postura do elo e em relação a b é extraída dessa matriz.

Nesta síntese assume-se que o elo b está posicionado na origem do sistema de coordenadas inercial (adotado para expressar os helicoides)

e alinhado com ele. Se este não for o caso, pode-se incluir uma transformação homogênea adicional que represente a posição e orientação deste elo em relação à origem do sistema de coordenadas. Da mesma forma, é possível acrescentar uma transformação homogênea que represente a diferença de postura entre o sistema de coordenadas atribuído ao elo e e um ponto de interesse neste elo ou em um elemento a ele vinculado (por exemplo, uma ferramenta montada sobre o efetuador final de um manipulador). Dessa forma, a cinemática direta pode ser reescrita como na Equação A.11,

$${}^b\mathbf{A}_f = {}^0\mathbf{A}_b {}^b\mathbf{A}_1 {}^1\mathbf{A}_2 \cdots {}^{n-1}\mathbf{A}_n {}^n\mathbf{A}_e {}^e\mathbf{A}_f \quad (\text{A.11})$$

onde ${}^0\mathbf{A}_b$ é a matriz homogênea entre o referencial do elo b e o referencial inercial e ${}^e\mathbf{A}_f$ é a matriz homogênea entre um referencial vinculado ao ponto de interesse F e o referencial do elo e .

Quando vários helicoides são relacionados, assume-se que todos estão representados segundo um mesmo referencial. Em alguns casos, é possível que os helicoides sejam definidos segundo sistemas de referência distintos, ou mesmo que se adote um sistema de referência final diferente daquele utilizado para a definição dos helicoides das juntas. Isso pode acontecer ao se adotar um referencial que simplifique as expressões dos helicoides e das matrizes de transformação homogêneas mas que seja distinto do necessário para o estudo de postura do problema (Simas, 2008). Nesses casos, uma transformação de helicoides ${}^i\mathbf{T}_j$ deve ser aplicada. Esta pode ser definida a partir da transformação homogênea entre os sistemas de coordenadas j e i , como ilustrado na Figura 83. A transformação é expressa como na Equação A.12,

$${}^i\mathbf{T}_j = \begin{bmatrix} {}^i\mathbf{R}_j & 0 \\ \mathcal{S}({}^i\mathbf{p}_j) {}^i\mathbf{R}_j & {}^i\mathbf{R}_j \end{bmatrix} \quad (\text{A.12})$$

onde $\mathcal{S}({}^i\mathbf{p}_j)$ é a matriz antissimétrica do vetor posição entre as origens dos dois sistemas de coordenadas e ${}^i\mathbf{R}_j$ é a matriz de rotação da orientação do sistema j para o sistema i .

A.2.2 Usando Grafos Para Representar Cadeias Cinemáticas

A análise de uma cadeia pode se tornar difícil em função da complexidade da sua estrutura cinemática. Esta pode ter um grande número de juntas e elos e formar circuitos fechados ou híbridos. Assim, a modelagem acaba se tornando complexa e eventualmente específica

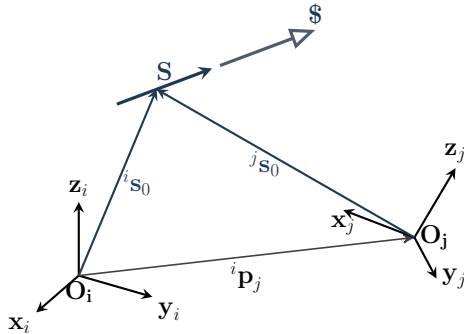


Figura 83. Transformação de helicoides

para cada estrutura cinemática, bem como a sua análise. Manipuladores paralelos, sistemas robóticos cooperativos e sistemas veículo-manipulador são casos típicos da robótica.

Para simplificar e sistematizar a análise de cadeias cinemáticas, faz-se a representação destas através da *teoria dos grafos*. Apesar de ser considerada uma área relativamente nova de pesquisa da matemática, essa teoria apresenta muitos resultados e aplicações práticas em várias áreas (Bang-Jensen; Gutin, 2007). Uma discussão detalhada do seu uso para mecanismos pode ser encontrada em (Tsai, 2000).

Um *grafo* é uma estrutura matemática utilizada para representar relações entre objetos de uma determinada coleção. Os objetos são representados por *vértices* (também chamados de nós ou pontos), enquanto as relações são definidas pelas *arestas* (também chamadas de linhas) (Bang-Jensen; Gutin, 2007). Uma relação entre dois objetos pode ser bidirecional ou apenas de um objeto para outro. Neste caso, a aresta que conecta os vértices relativos aos objetos é direcionada. Um grafo que tem esse tipo de aresta é chamado de *dígrafo* (Diestel, 2005).

Um *caminho* é um conjunto de arestas que conectam dois vértices quaisquer. Ele é *fechado* quando o vértice de partida é o mesmo vértice de chegada, formando um *circuito* ou *ciclo*.

Existem diversas estruturas de dados que podem ser usadas para representar um grafo. A *matriz de adjacências* é uma delas, onde as linhas e colunas correspondem a vértices e o conteúdo de cada célula indica se há uma aresta entre um vértice l e um vértice c . Outra representação usada é a *matriz de incidências*, onde as linhas representam os vértices e as colunas representam as arestas, e as células indicam se a aresta c incide em um vértice l .

Outra matriz utilizada no estudo de grafos é a *matriz de circuitos*, em que cada linha corresponde a um circuito do grafo e cada coluna corresponde a uma aresta. O conteúdo da célula com endereço (l, c) indica se a aresta c é parte do circuito l . No caso de dígrafos, o conteúdo da célula pode indicar se o sentido da aresta coincide ou não com o sentido do circuito. Uma representação simples para esse caso, em um circuito l , é a célula assumir o valor 0 para uma aresta que não faça parte do circuito, +1 se tiver o mesmo sentido do circuito e -1 se o seu sentido for contrário ao do circuito.

Para usar grafos na análise cinemática de cadeias, associam-se os elos aos vértices e as juntas às arestas. Um par cinemático formado por dois elos a e b conectado por uma junta 1 é representado como no grafo da Figura 84. O sentido da aresta que representa a junta 1 indica que esta gera o movimento do elo b em relação ao elo a .

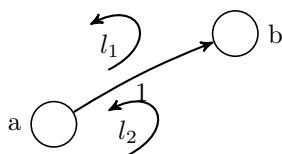


Figura 84. Representação de um par cinemático por um grafo

Ao se representar a cadeia cinemática inteira, vários circuitos podem surgir. No exemplo da Figura 84 é assumido que a junta faça parte dos circuitos l_1 e l_2 . Se o grafo da cadeia cinemática tem 3 circuitos independentes (que servem de base para construir todos os circuitos da cadeia), a coluna da matriz de circuitos \mathbf{B} correspondente à junta 1 assume os valores mostrados na Equação A.13.

$$\mathbf{B} = \begin{matrix} & \dots & 1 & \dots \\ l_1 & \left[\begin{array}{ccc} \dots & +1 & \dots \\ \dots & -1 & \dots \\ \dots & 0 & \dots \end{array} \right] & & \end{matrix} \quad (\text{A.13})$$

Existem dois tipos de dígrafos usados para a representação de cadeias cinemáticas. O *dígrafo de acoplamento* representa a relação entre os elos, ou seja, os pares cinemáticos da cadeia. O *dígrafo de movimento*, por sua vez, representa o grau de liberdade entre os pares. Este é obtido pela substituição de cada aresta correspondente a uma junta de mais de um grau de liberdade do dígrafo de acoplamento por um subgrafo com o número de arestas igual aos graus de liberdade da

junta. Caso todas as juntas da cadeia cinemática sejam de um grau de liberdade, os dígrafos de acoplamento e movimento serão idênticos. A análise cinemática é feita a partir do dígrafo de movimento.

Para exemplificar a descrição por grafos, será usado o mecanismo planar de quatro barras apresentado na Figura 85a. Todas as juntas do mecanismo são rotativas, o que faz com que o dígrafo de acoplamento e o de movimento sejam iguais. Ele é apresentado na Figura 85b. A matriz de circuitos \mathbf{B} correspondente é descrita na Equação A.14.

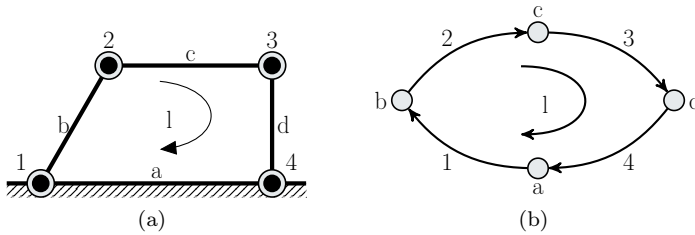


Figura 85. Mecanismo de 4 barras: (a)representação funcional; (b)grafo de movimento

$$\mathbf{B} = l \begin{bmatrix} 1 & 2 & 3 & 4 \\ +1 & +1 & +1 & +1 \end{bmatrix} \quad (\text{A.14})$$

A.2.3 Cinemática Diferencial de Uma Cadeia

Em uma cadeia cinemática, os heligiros representam o estado de velocidades dos pares cinemáticos formados por dois elos vinculados por uma junta. Um heligiro descreve o movimento instantâneo de um elo em relação ao outro elo ao qual está vinculado. O movimento instantâneo de um elo e em relação a um elo b é determinado pelo somatório dos heligiros das juntas da cadeia compreendida entre estes elos, como expresso na Equação A.15, onde o índice i corresponde à junta responsável pelo movimento relativo do elo i em relação ao seu antecessor $i - 1$.

$$\mathcal{S}_e = \sum_{i=b+1}^e \mathcal{S}_i = \begin{bmatrix} \omega_e \\ \mathbf{v}_{p_e} \end{bmatrix} = \sum_{i=b+1}^e \hat{\mathcal{S}}_i \dot{q}_i = \mathbf{J} \dot{\mathbf{q}} \quad (\text{A.15})$$

Ao se decompor os heligiros nos seus helicoides normalizados e suas magnitudes, obtém-se a matriz Jacobiana $\mathbf{J} = [\hat{\mathbf{s}}_{b+1} \cdots \hat{\mathbf{s}}_e]$ e o vetor coluna $\hat{\mathbf{q}} = [\hat{q}_{b+1} \cdots \hat{q}_e]^T$. O Jacobiano, cujas colunas são formadas pelos helicoides normalizados, relaciona as velocidades das juntas à velocidade do elo e . Os elementos de $\hat{\mathbf{q}}$, por sua vez, estão associados às colunas de \mathbf{J} e correspondem às magnitudes das velocidades das juntas (Tsai, 1999; Hunt, 2000; Campos, 2004).

A.3 ANÁLISE CINEMÁTICA

Para manipuladores seriais, a determinação do movimento dos elos é relativamente simples. Porém, à medida em que as cadeias cinemáticas tornam-se mais complexas, os grafos de movimentos passam a ser uma ferramenta importante na análise do movimento dos elos da cadeia. Em cadeias fechadas, o método de Davies é útil para relacionar as velocidades do sistema. As cadeias virtuais de Assur, por sua vez, são instrumentais para a análise do movimento de elos específicos de uma cadeia, bem como na imposição de restrições.

A.3.1 O Método de Davies

O *método de Davies* é uma adaptação da lei de circulação de Kirchhoff para o uso em cadeias cinemáticas fechadas com a finalidade de definir a sua cinemática diferencial (Campos, 2004). Davies estabelece que “a soma algébrica das velocidades relativas dos pares cinemáticos em uma cadeia fechada é igual a 0” (Davies, 1981), resultando em uma *equação de restrição* na forma expressa na Equação A.16,

$$\sum_{i=1}^n \hat{\mathbf{s}}_i = \sum_{i=1}^n \hat{\mathbf{s}}_i \hat{q}_i = \mathbf{N}\hat{\mathbf{q}} = \mathbf{0} \quad (\text{A.16})$$

onde \mathbf{N} é a *matriz de rede* que relaciona os movimentos das juntas aos circuitos independentes da cadeia cinemática. Esta é definida a partir da análise do grafo de movimento da cadeia cinemática. Ela tem dimensão $\lambda l \times F_b$, onde λ é a dimensão do espaço de helicoides, l é o número de circuitos independentes da cadeia e F_b é o seu número de graus de liberdade bruto (igual ao número de arestas do grafo de movimento). A matriz \mathbf{N} é então definida como na Equação A.17,

$$\mathbf{N} = \begin{bmatrix} \mathbf{D} \text{diag} \{ \mathbf{B}_1 \} \\ \vdots \\ \mathbf{D} \text{diag} \{ \mathbf{B}_l \} \end{bmatrix} \quad (\text{A.17})$$

onde \mathbf{D} é uma matriz cujas colunas são formadas pelos helicoides normalizados da cadeia (com dimensão $\lambda \times F_b$) e \mathbf{B} é a matriz de circuitos do grafo de movimento (com dimensão $l \times F_b$). O operador $\text{diag} \{ \mathbf{B}_i \}$ forma uma matriz diagonal com os elementos da linha i da matriz \mathbf{B} .

Pode-se afirmar que a equação de restrição estabelece λl restrições para a cadeia fechada. O número de variáveis independentes, ou seja, a *mobilidade* da cadeia cinemática, é igual a $F_N = F_b - \lambda l$. Estas correspondem às juntas atuadas da cadeia.

A equação de restrição relaciona as velocidades de todas as juntas da cadeia (Simas, 2008). Assim, é possível usá-la para calcular as velocidades de algumas juntas da cadeia em função das velocidades das demais juntas. Para tanto, a equação de restrição é particionada em dois conjuntos. A partição *primária* corresponde às juntas cujas magnitudes das velocidades são conhecidas, enquanto a partição *secundária* é formada pelas variáveis de juntas cujas magnitudes devem ser determinadas. Assim, obtém-se a Equação A.18, onde os subscritos p e s correspondem às partições primária e secundária, respectivamente.

$$\mathbf{N} \dot{\mathbf{q}} = \mathbf{N}_p \dot{\mathbf{q}}_p + \mathbf{N}_s \dot{\mathbf{q}}_s = \mathbf{0} \quad (\text{A.18})$$

Os valores das variáveis secundárias são determinados isolando-as na Equação A.18, resultando na Equação A.19.

$$\dot{\mathbf{q}}_s = -\mathbf{N}_s^{-1} \mathbf{N}_p \dot{\mathbf{q}}_p \quad (\text{A.19})$$

As magnitudes da partição secundária são obtidas se a matriz \mathbf{N}_s for inversível. Do contrário, o sistema encontra-se em alguma singularidade (Campos, 2004). Pelas definições da equação de restrição, pode-se usar o método de Davies para λl variáveis secundárias para \mathbf{N}_s ser quadrada. Em algumas situações, porém, podem haver mais variáveis secundárias que esse valor, tornando \mathbf{N}_s retangular. Nesses casos, se for possível, pode-se arbitrar valores para algumas dessas variáveis, tornado-as primárias de forma a manter \mathbf{N}_s quadrada. Outra opção é utilizar a operação pseudoinversa (Guenther et al., 2008).

A.3.2 Cadeias Virtuais de Assur

O método de Davies apresenta como limitação o fato de ser aplicável apenas a cadeias fechadas. Além disso, não é possível usá-lo para obter ou impor o movimento específico de um ponto que faz parte de um elo da cadeia cinemática. O uso de *cadeias cinemáticas virtuais* auxilia a superar tais limitações.

O conceito de cadeia cinemática virtual foi introduzido por Davies, tendo sido posteriormente analisado e desenvolvido por Campos (Campos, 2004). As cadeias virtuais podem ser consideradas ferramentas para monitoramento de cadeias cinemáticas reais ou para imposição de movimentos/restrições sobre estas (Campos, 2004; Campos; Guenther; Martins, 2005).

Como monitoramento, as cadeias virtuais são usadas de forma a se obterem informações sobre deslocamentos de elos e juntas da cadeia real. As informações obtidas através delas podem ser usadas em estratégias de evitamento de obstáculos, de verificação de limites das juntas ou de evitamento de singularidades.

Na imposição de movimentos e restrições em cadeias reais, as cadeias virtuais são usadas para especificar tarefas ou impor restrições ao movimento de determinados elementos da cadeia real. Nesse caso, cadeias virtuais são úteis para explorar a mobilidade adicional de cadeias cinematicamente redundantes.

As cadeias virtuais são formadas por elos e juntas, como cadeias cinemáticas reais, e devem ter as seguintes propriedades:

1. serem abertas;
2. terem seus helicoides normalizados relativos às juntas linearmente independentes;
3. não modificarem a mobilidade de cadeias reais quando vinculadas a estas.

Como consequência destas propriedades, a mobilidade das cadeias virtuais deve ser igual à ordem do sistema de helicoides λ das cadeias reais às quais são vinculadas.

As cadeias virtuais são classificadas como *grupos de Assur*, sendo também denominadas cadeias virtuais de Assur para evitar ambiguidades com o termo virtual (Ribeiro; Guenther; Martins, 2008), que neste contexto significa que os elos e juntas da cadeia não existem fisicamente.

Em cadeias abertas, características de manipuladores seriais, as cadeias virtuais são usadas para fechá-las, viabilizando o emprego do

método de Davies para resolver a cinemática diferencial da cadeia modificada resultante. Em cadeias fechadas, o uso de cadeias virtuais acrescenta malhas ao circuito existente, permitindo tanto a análise do movimento quanto a adição de restrições (Campos; Guenther; Martins, 2005).

Os tipos de cadeias virtuais comumente utilizados são apresentados na Figura 86. Elas são classificadas em planares e espaciais. As cadeias virtuais planares usuais são do tipo PPR e RPR por poderem representar sistemas de coordenadas cartesianos e polares, respectivamente. No caso espacial, as cadeias PPPS, RPPS e RRPS são comumente empregadas por estarem respectivamente associadas a sistemas de coordenadas cartesianos, cilíndricos e esféricos.

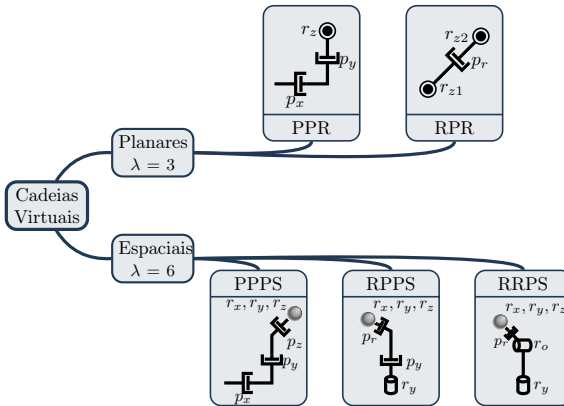


Figura 86. Cadeias virtuais de Assur

A.4 PLANEJAMENTO DE MOVIMENTO PARA A REALIZAÇÃO DE TAREFAS

As cadeias virtuais de Assur permitem impor o movimento de qualquer elo de uma cadeia cinemática real. Assim, elas podem ser utilizadas na definição de tarefas a serem executadas por um sistema robótico.

Nesse caso, uma tarefa consistiria de referências de posição e velocidade que seriam aplicadas a uma cadeia virtual para impor o movimento de um ponto do sistema em relação a um referencial. O tipo de cadeia virtual seria escolhido de acordo com o movimento a

ser imposto. A cadeia seria vinculada aos pontos da cadeia real entre os quais o movimento é desejado. Pode-se estender a especificação da tarefa para o movimento/restrrição de mais de um ponto do robô. Para isso, adicionam-se mais cadeias virtuais e descrições das suas referências de posição e velocidade.

A adição de cadeias virtuais ao sistema deve definir uma cadeia cinemática final fechada. Assim, pode-se aplicar o método de Davies para a resolução cinemática. Para isso, o particionamento da equação de restrição é feito de maneira que as variáveis cujas posições/velocidades sejam definidas na tarefa estejam na partição primária, enquanto as demais comporão a partição secundária, devendo ser calculadas na solução da equação de restrição. Obtidas as velocidades, realiza-se a integração para se obter as referências de posição.

A.4.1 O Método das Restrições Cinemáticas

Introduzido em (Santos, 2006), o método das restrições cinemáticas é uma sistematização do processo de resolução cinemática de um sistema robótico em função da tarefa que foi descrito acima.

Escolhendo adequadamente as juntas a serem atuadas (virtuais e reais) é possível definir a matriz \mathbf{N}_s como quadrada, permitindo sua inversão. Obviamente, as demais juntas devem ter seu movimento definido. No caso de manipuladores redundantes, há liberdade para adicionar restrições além das que impõem o movimento do elo considerado, permitindo explorar a maior destreza dessas cadeias cinemáticas. Isso significa que eventualmente alguma junta real da cadeia pode ser referenciada na tarefa, ao invés de alguma junta virtual. Da mesma forma, as juntas de uma cadeia virtual podem ser secundárias, como no caso em que se quer monitorar o movimento de algum ponto do sistema. Nos casos em que não é possível definir \mathbf{N}_s como quadrada, pode-se empregar uma pseudoinversão.

Simas mostra que esse método equivale ao Jacobiano estendido, para resolução de cadeias cinemáticas redundantes. Para tanto, utiliza o conceito de operador aniquilador de cadeias passivas secundárias. Tal demonstração pode ser vista em (Simas, 2008).

Em relação aos métodos mais tradicionais, usando pseudoinversa, prioridade da tarefa ou jacobiano estendido¹, o método das restrições cinemáticas apresenta como vantagem o fato de não apresentar

¹Estes métodos são apresentados e analisados em (Chiaverini; Oriolo; Walker, 2008)

inconsistências dimensionais (Santos, 2006; Fontan, 2007; Simas, 2008). Além disso, o método ainda apresenta a característica de conservação do movimento (Simas, 2008).

A.4.2 Resolução da Cinemática Inversa

As posições das juntas são obtidas através da integração das velocidades determinadas pelo método das restrições cinemáticas, em função das velocidades impostas nas juntas virtuais. A relação entre estas velocidades é estabelecida por uma matriz jacobiana. Esta, em geral, é bastante complexa, o que dificulta a obtenção de uma solução analítica, levando ao uso de métodos de integração numérica. Por serem baseados em aproximações, porém, estes métodos geram erros, em um fenômeno conhecido como *deriva*. Em cadeias abertas, isso resulta em indesejáveis erros de postura. Em cadeias fechadas o problema é mais crítico, pois estas acabam por abrir (Simas et al., 2009).

Uma solução adotada por muitos autores consiste em compensar o erro de posição nas referências da tarefa adotando uma realimentação das posições calculadas no processo conhecido como *cinemática inversa em malha fechada* ou CLIK (do inglês *Closed-Loop Inverse Kinematics* (Siciliano et al., 2009; Antonelli, 2006). Uma proposta baseada em helicoides foi desenvolvida por Simas et al. (2009).

O método de integração proposto por Simas consiste em representar o erro causado pelo método numérico de integração por *cadeias virtuais de erro* de fechamento em cada malha (Simas, 2008; Guenther et al., 2008; Fontan, 2007).

As cadeias mais convenientes para a representação do erro, são as PPR e 3P3R (caso plano e espacial, respectivamente), pois além de desacoplar os erros de posição e de orientação, evita introduzir singularidades com as cadeias de erro (Fontan, 2007). A adição destas não modifica o método de Davies, mas a matriz de rede \mathbf{N}_s passa a ser particionada levando em consideração tais cadeias. Assim, tem-se

$$\mathbf{N}_s \dot{\mathbf{q}}_s + \mathbf{N}_p \dot{\mathbf{q}}_p + \mathbf{N}_e \dot{\mathbf{q}}_e = 0 \quad (\text{A.20})$$

Para obter $\dot{\mathbf{q}}_s$, acrescenta-se um termo de erro de posição à equação A.19, resultando na equação de erro

$$\dot{\mathbf{q}}_s = -\mathbf{N}_s^{-1} (\mathbf{N}_p \dot{\mathbf{q}}_p - \mathbf{N}_e \mathbf{K}_e \mathbf{q}_e) \quad (\text{A.21})$$

Ao se substituir $\dot{\mathbf{q}}_s$ na equação A.20, e eliminando \mathbf{N}_e , obtém-se

$$\mathbf{K}_e \mathbf{q}_e + \dot{\mathbf{q}}_e = 0 \quad (\text{A.22})$$

que é assintoticamente estável, se \mathbf{K}_e for definida positiva (Simas, 2008; Simas et al., 2009). Fontan mostra em exemplos tal resultado (Fontan, 2007).

O erro é definido pelo método dos helicoides sucessivos, a partir da observação de que, em uma cadeia fechada, o produto das transformações que representam a postura de um determinado elo em relação a si mesmo é igual à identidade. Separando estas em primárias, secundárias e de erro, tem-se

$$\mathbf{A}_{p_1} \mathbf{A}_{p_2} \cdots \mathbf{A}_{p_{n_p}} \mathbf{A}_{s_1} \mathbf{A}_{s_2} \cdots \mathbf{A}_{s_{n_s}} \mathbf{E} = \mathbf{I} \quad (\text{A.23})$$

A matriz de erros é obtida, isolando-a na equação A.23, resultando em

$$\mathbf{E} = \{ \mathbf{A}_{p_1} \mathbf{A}_{p_2} \cdots \mathbf{A}_{p_{n_p}} \mathbf{A}_{s_1} \mathbf{A}_{s_2} \cdots \mathbf{A}_{s_{n_s}} \}^{-1} \quad (\text{A.24})$$

$$= \left\{ \prod_{i=1}^{n_p} \mathbf{A}_{p_i} \prod_{j=1}^{n_s} \mathbf{A}_{s_j} \right\}^{-1} = \begin{bmatrix} \mathbf{R}_e & \mathbf{p}_e \\ \mathbf{0} & 1 \end{bmatrix} \quad (\text{A.25})$$

A partir da matriz de rotação \mathbf{R}_e e do vetor de posição \mathbf{p}_e da equação A.25, extraem-se os erros de postura.

Tal solução prova-se efetiva mesmo com métodos de integração numérica simples como o método de Euler. Dado um intervalo Δt , e conhecidas as velocidades do instante t_{k-1} , as posições nas juntas podem ser calculadas por

$$\mathbf{q}_s(t_k) = \mathbf{q}_s(t_{k-1}) - \mathbf{N}_s^{-1}(t_{k-1}) (\mathbf{N}_p(t_{k-1}) \dot{\mathbf{q}}_p - \mathbf{N}_e(t_{k-1}) \mathbf{K}_e \mathbf{q}_e) \Delta t \quad (\text{A.26})$$

Apesar de apresentar estabilidade assintótica, pode-se desejar minimizar o erro ao longo de toda a trajetória. Para tanto, pode-se empregar um laço secundário no processo de integração, no qual se mantém constante a posição das juntas primárias. Neste laço, as iterações ocorrem até o erro se encontrar em uma tolerância admissível (Simas et al., 2009). Como $\dot{\mathbf{q}}_p = 0$, a Equação A.21 se reduz a

$$\dot{\mathbf{q}}_s = \mathbf{N}_s^{-1} \mathbf{N}_e \mathbf{K}_e \mathbf{q}_e \quad (\text{A.27})$$

APÊNDICE B – Implementação e Uso do *Framework* KAST

Este apêndice trata da implementação do *framework* Kast (*Kinematic Analysis by Screw Theory*). Inicia-se pela escolha da plataforma computacional usada para seu desenvolvimento e uso posterior. A seguir, descreve-se o conjunto de ferramentas computacionais empregadas na plataforma adotada (Python) e software de apoio utilizado. São discutidos aspectos de implementação do Kast. Um exemplo de uso do *framework* é então apresentado para ilustrar seu uso e possibilidades de extensão.

B.1 ESCOLHA DE UMA PLATAFORMA COMPUTACIONAL

A escolha da plataforma computacional a ser usada para a implementação do *framework* foi feita logo após a definição dos seus requisitos de projeto (Rocha; Tonetto; Dias, 2011a). Ela foi realizada em paralelo ao processo de projeto, tendo algumas revisões em função de necessidades detectadas ao longo do projeto. Observou-se que a adoção de uma plataforma orientada a objetos seria interessante para o mapeamento direto entre as entidades projetadas e a sua codificação. Porém, procurou-se manter a neutralidade na escolha de um conjunto representativo de ferramentas computacionais que são de fato empregadas para análise, simulação e implementação em robótica.

B.1.1 Requisitos Para a Plataforma

Os requisitos de escolha da plataforma computacional basearam-se nos requisitos do projeto do *framework* Kast. A plataforma deveria suportar os requisitos de modularidade e extensibilidade dos componentes do *framework* tanto para o seu desenvolvimento quanto para o seu uso. A plataforma também deveria ter possibilidade de se trabalhar no modo interativo (com uma interface de comandos/gráfica para o usuário) e também de desenvolvimento de aplicações que executassem autonomamente (*standalone* em inglês). Assim, seria possível trabalhar tanto em análise de cadeias cinemáticas quanto em simulações (usando o modo interativo) e em software a ser embarcado em sistemas reais (como aplicação *standalone*). Uma vez que o projeto utilizou o paradigma da orientação a objetos, um requisito desejável para a plataforma seria o suporte à programação orientada a objetos.

Outros requisitos foram identificados pelas particularidades da implementação das *features* descritas para o *framework*. A existência

de bibliotecas para manipulação matemática numérica de matrizes e álgebra linear foi um deles. O suporte a grafos e algoritmos para a análise destes foi outro requisito essencial. Para lidar com descrições textuais de cadeias cinemáticas e outras estruturas de dados necessárias, era necessário ter bibliotecas que gerenciassem e manipulassem diferentes tipos de dados estruturados, com destaque para o XML, que foi o tipo textual adotado para as descrições de cadeias. A questão de poder lidar com vínculos dinâmicos de diferentes componentes de software também era importante, para possibilitar a flexibilidade de uso dos componentes e a modificabilidade das cadeias cinemáticas em tempo de execução.

Como requisitos desejáveis para implementação do *Kast* estavam a disponibilidade em diferentes sistemas operacionais e dispositivos computacionais e o licenciamento como software livre/aberto. Desejava-se que o *framework* fosse passível de uso no maior número de plataformas de hardware/software utilizadas em simulações e casos reais e que fosse disponibilizado como software livre. Para isso, era importante que a plataforma fosse acessível (tanto em custo quanto em uso), com uma boa base de conhecimento disponível em literatura/Internet, e que possibilitasse o licenciamento de software nela desenvolvido como software livre.

O suporte a gráficos 2D/3D, animações, armazenamento de imagens e vídeos e capacidade de troca de informações com outros software eram também requisitos desejáveis para a plataforma, a fim de complementar o uso do *framework* com recursos usualmente necessários em simulações e implementações de sistemas reais.

B.1.2 Plataformas Avaliadas

Apesar de existirem várias plataformas computacionais em uso para simulação e implementação em engenharia, a escolha da plataforma de desenvolvimento do *framework* se deu em um conjunto de seis delas. O conjunto incluía os software comumente empregados nos Laboratórios de CAD/CAM e de Robótica da Universidade Federal de Santa Catarina, que atendiam boa parte dos requisitos. Além disso, foram consideradas características como maturidade, existência de suporte/documentação, base de usuários das plataformas e sua colaboração. Por fim, o conjunto contemplava os software com os quais o desenvolvedor do *framework* teve experiências de uso.

O conjunto consistia de três ambientes de computação numé-

rica e três plataformas de desenvolvimento de software. Os ambientes de computação numérica em questão foram o Matlab[®] (Mathworks, 2011), o Scilab (Digiteo, 2011) e o Octave (Eaton, 2011). As plataformas de desenvolvimento eram as das linguagens de programação C/C++ (Stroustrup, 2011), Java (Oracle, 2011) e Python (Python.org, 2011). As Tabelas 8 e 9 relacionam as plataformas e identificam o grau de atendimento aos requisitos.

Tabela 8. Ambientes de computação numérica avaliados para a implementação do *framework* Kast

Plataforma	Matlab [®]	Scilab	Octave
Matrizes e álgebra linear	Sim	Sim	Sim
Grafos	Sim	Sim	Sim
Gráficos 2D/3D	Sim	Sim	Sim
Animação	Sim	Sim	Sim
Extensível	Sim	Sim	Sim
Orientada a objetos	Não	Não	Não
Ambiente interativo	Sim	Sim	Sim
Aplicações autossuficientes	MCR ¹	Não	Não
Interface com aplicações	Sim	Sim	Sim
Embarcável	Sim	Não	Não
Tipo	Comercial	Software Livre	Software Livre

Dentre as opções do conjunto de plataformas, foi escolhido o Python. Embora todas atendessem os requisitos essenciais e boa parte dos desejáveis (exceto a questão da orientação a objetos), observou-se que o Python era o único que reunia características tanto de ambiente de computação numérica/simbólica quanto de plataforma de desenvolvimento. A possibilidade de criar software de execução *standalone* em várias plataformas é um diferencial em relação aos outros ambientes de computação numérica. A interação com usuário típica desses ambientes, por outro lado, é o diferencial do Python em relação às demais plataformas de desenvolvimento.

Além disso, verificou-se a possibilidade de Python empregar bibliotecas de códigos desenvolvidas em outras linguagens de programação e mesmo de fazer interface com ambientes de computação numérica (Python.org, 2011). Outros softwares também utilizam o Python como interface com terceiros, o que amplia o potencial de uso da plataforma (Diankov, 2010).

Tabela 9. Plataformas de desenvolvimento de software avaliadas para a implementação do *framework* Kast

Plataforma	C/C++	Java	Python
Matrizes e álgebra linear	Sim	Sim	Sim
Grafos	Sim	Sim	Sim
Gráficos 2D/3D	Sim	Sim	Sim
Animação	Sim	Sim	Sim
Extensível	Sim	Sim	Sim
Orientada a objetos	Sim	Sim	Sim
Ambiente interativo	Não	Não	Sim
Aplicações autossuficientes	Sim	JRE ²	Sim
Interface com aplicações	Sim	Sim	Sim
Embarcável	Sim	Sim	Sim
Tipo	Software Livre	Software Livre	Software Livre

B.2 PYTHON COMO PLATAFORMA DE DESENVOLVIMENTO

Python é uma linguagem de programação de alto nível de uso geral. Criada por Guido van Rossum para fins científicos na década de 1990, a linguagem se popularizou entre os desenvolvedores de software livre e de sistemas web. A linguagem, porém, também tem um forte nicho de aplicações no meio acadêmico e científico. Python é orientada a objetos, com características de programação dinâmica. Além disso, tem acesso a muitas bibliotecas de software já existentes e sistemas operacionais, podendo ser estendida através de módulos construídos em C/C++ (Python.org, 2011).

As funcionalidades de Python são organizadas em módulos. Esses podem consistir de interfaces para bibliotecas pré-existentes criadas em C/C++, Fortran ou outra linguagem que gere códigos binários, serem implementações completas em Python ou consistirem de interface de integração com outros tipos de linguagens/ambientes como Java e .Net. A biblioteca padrão que está embutida no interpretador Python contém muitas funcionalidades que dispensam o uso de outros módulos para aplicações simples. Dependendo da finalidade da aplicação, módulos podem ser importados para disponibilizar classes, estruturas de dados e funções específicas. Por fim, Python é disponível como software livre, assim como a maioria de seus módulos (Borges, 2010).

A versão de Python utilizada no desenvolvimento do Kast, seus

testes e uso em simulações foi a 2.7, executando em sistemas operacionais Linux (Mandriva 2010.2 e Kubuntu 11.04). A plataforma de desenvolvimento completa (Python com biblioteca padrão e demais módulos) foram instalados a partir de pacotes dessas distribuições Linux. Os códigos desenvolvidos foram também executados, sem modificações, em um sistema operacional Windows Vista com uma distribuição Python chamada Python(X,Y), voltada para fins científicos (Raybaut, 2011).

B.2.1 Módulos/Bibliotecas Empregados

Além da biblioteca padrão do Python, outros módulos se tornaram necessários para lidar com matrizes, álgebra linear, grafos e XML. Para cada uma dessas demandas, foram identificados mais de um módulo existente. A escolha dos módulos utilizados no *framework* se deu pela análise comparativa das funcionalidades existentes em cada módulo, de sua manutenção/nível de atividade, existência de documentação de suporte, de recomendações de usuários de diferentes plataformas e da facilidade de uso, comprovada pela execução de exemplos.

Para representação de matrizes, seus comportamentos e ações, bem como métodos de álgebra linear, foi adotado o *Numpy*, parte de um projeto mais abrangente de computação científica para Python denominado *Scipy*. O *Numpy* é a base de outros módulos matemáticos para Python, com capacidade de representação de matrizes, vetores e funções de álgebra linear, manipulação de polinômios, curvas, transformações de Fourier, entre outros (Numpy, 2011; Scipy, 2011).

O *NetworkX* foi o módulo utilizado para a criação e manipulação de grafos. Esse módulo tem classes para representação de grafos, onde seus nós e arestas podem conter quaisquer estruturas de dados/objetos. O conjunto de algoritmos de análise implementados nesse módulo facilitam o estudo dos grafos. De forma complementar, existem ainda facilidades para desenho automático dos grafos (NetworkX, 2011).

A construção de classes de gerenciamento de descrições de cadeias cinemáticas e outros dados de simulação demandava o desenvolvimento de analisadores de formatos de dados textuais. Os formatos escolhidos foram o XML e o CSV, por serem comuns em diferentes campos de conhecimento, aceitos por planilhas eletrônicas e outros software de processamento científico. O formato CSV, além de fácil compreensão, é gerenciado pelo módulo *Numpy*. Para o XML, foi utilizado o módulo *ElementTree*, que trata os dados organizados hierarquicamente em um arquivo XML como uma árvore (Lundh, 2011).

Apesar de não ter sido usado diretamente na implementação do *framework*, o módulo *matplotlib* auxiliou os testes dos diferentes componentes através da visualização dos resultados gerados. Ele fornece um conjunto de funcionalidades para traçados de gráficos 2D e 3D bastante sofisticado, com uma opção de uso que permite emular o comportamento de bibliotecas similares em sistemas de computação numérica como o MATLAB (Hunter; Dale; Droettboom, 2011).

B.2.2 Software de Apoio

Além do Python e seus módulos, outras ferramentas foram utilizadas nas simulações que foram implementadas com o Kast. O Maxima foi utilizado para auxiliar na obtenção das expressões dos heligiros das cadeias cinemáticas, enquanto o OpenRAVE foi utilizado para gerar a animação de modelos 3D dos cenários simulados.

O Maxima é um sistema de computação algébrica (ou CAS, do inglês *Computer Algebra System*) para manipulação de expressões numéricas e simbólicas, distribuído como software livre para diferentes sistemas operacionais. É um sistema robusto e maduro, cujo desenvolvimento vem da década de 1960 (Beshenov, 2011). Esse CAS foi empregado para definir os modelos de cadeias cinemáticas simples e sua composição. Para tanto, foram desenvolvidas funções na linguagem do Maxima que calculavam matrizes de transformação homogênea, transformações de helicoides e predefinições de cadeias simples como a PPR e a RRR. Os recursos de simplificação algébrica e trigonométrica do Maxima permitiram agilizar as definições das cadeias, bem como a escolher modos de implementar uma versão otimizada para o caso do sistema de UVMS com um manipulador planar, que é descrito na próxima seção.

O OpenRAVE é um simulador de sistemas robóticos que provê um ambiente para teste, desenvolvimento e aplicação de algoritmos de planejamento de movimento. Ele pode ser usado tanto para simulação quanto como interface com sistemas robóticos reais. Embora seu foco primário seja a cinemática e modelagem geométrica, ele possui capacidades de representar a dinâmica de sistemas e simular diferentes tipos de sensores. Ele faz interface com ambientes de computação numérica como Matlab e Octave, além de ter um módulo Python que permite seu acesso dentro de aplicações escritas nessa linguagem (Diankov, 2010). Nas simulações realizadas, o OpenRAVE foi empregado para visualizar um ambiente virtual de execução de tarefas que era animado a par-

tir dos dados resultantes das execuções das simulações. Os recursos de geração de vídeos em diferentes formatos e de imagens estáticas do ambiente virtual facilitou a apresentação de resultados e sua análise.

B.3 ASPECTOS DA IMPLEMENTAÇÃO DO *FRAMEWORK* KAST

A adoção do Python como plataforma de desenvolvimento impôs algumas mudanças na implementação do *framework* em relação à sua modelagem. Embora não tenha sido feita nenhuma modificação das especificações de métodos e classes, as suas assinaturas sofreram algumas mudanças em função das características da linguagem, como suporte a sobrecarga de operadores, possibilidade de incorporação de objetos à estruturas de controle da linguagem e a existência de propriedades. A manipulação nativa de conjuntos de dados também acrescentou facilidades de implementação, porém implicando em pequenas modificações nas estruturas de dados originalmente propostas. Além disso, o fato de lidar com tipos de dados dinâmicos permitiu agrupar métodos que realizavam operações similares sobre tipos de dados diferentes.

Uma vez que o projeto foi baseado no paradigma da orientação a objetos, procurou-se adotar as suas boas práticas, como o uso de padrões de projeto comentado no Capítulo 5. Para isso, foram criadas classes base para implementação do padrão *observer*, onde objetos podem receber notificações de modificações feitas em outros objetos. Essas são as classes *Event* (de representação de eventos), *Listener* e *Subject* (representando objetos observadores e objetos observados, respectivamente). Várias classes são derivadas de *Listener*, como as cadeias cinemáticas e helicoides. As listas de atributos e conjuntos de variáveis de juntas são derivadas de *Subject*, notificando aos seus usuários (as cadeias cinemáticas, por exemplo) quando algum dos seus dados tem seu valor modificado.

Outros padrões de projeto como o *composite*, para definição de cadeias cinemáticas, utilizaram a característica de herança inerente à orientação a objetos, como a hierarquia de classes de componentes cinemáticos (derivadas de *KCComponent*, como *Screw* e *KinematicChain*). O padrão *factory method* foi implementado nos analisadores de descrições XML para criação de objetos, como *KCFactory* e *TrajFactory*.

Apesar de uma análise do *Kast* ser bastante extensa, alguns pontos importantes para o seu uso básico são apresentados a seguir. Maiores detalhes são obtidos na leitura do código dos componentes de *Kast* e de seus exemplos, que são disponibilizados como software livre.

B.3.1 Utilização das Facilidades da Plataforma

A implementação das classes de *Kast* foi bastante facilitada pela utilização das estruturas de dados nativas do Python e de seus módulos de componentes adicionais. Listas, tuplas e dicionários são estruturas de dados nativas que foram extensamente usadas. As duas primeiras armazenam dados em conjuntos que podem ser acessados por sua posição. O último vincula chaves aos dados, que são valores usados como referência para o acesso a eles.

O módulo Numpy forneceu modos de lidar com matrizes através da classe `array`. Além de representar a estrutura de dados da matriz, essa classe permite realizar diferentes operações aritméticas além de ter métodos específicos para tratar de matrizes numéricas e álgebra linear. Uma característica dos arrays que foi muito empregada para compartilhamento de dados entre diferentes objetos foi o *slicing*, técnica que permite criar visões de seções retangulares de uma matriz que podem ser compartilhadas. Diferente de cópias, uma visão é uma referência ao conjunto de dados original, de forma que a modificação nos elementos da visão é perceptível no array original.

A partir dessas classes, foram implementadas diversos componentes das classes principais relacionadas às cadeias cinemáticas, como `AttributeList`, `State` e `StateList`.

A classe `AttributeList` armazena atributos da cadeia cinemática em uma lista acessada por um identificador textual (o nome do atributo). Ela é utilizada nos componentes cinemáticos (derivadas de `KC-Component` como helicoides, juntas e cadeias cinemáticas) para relacionar comprimentos de elos e outros dados que não costumam se modificar durante a execução de uma tarefa. Na sua implementação foram utilizados dicionários e listas.

`State` é a classe que trata das variáveis de juntas com seus valores de posição, velocidade e aceleração. Para facilitar o acesso aos dados, foram utilizados dicionários com nomes das variáveis das juntas juntamente com arrays que armazenam os seus valores. Assim, criou-se um tipo de dados matricial cujos elementos podem ser acessados tanto por posição quanto por seu nome. O fato de utilizar arrays permite trabalhar também com *slices* que podem ser tanto horizontais (todos os dados de uma variável de junta: posição, velocidade e aceleração) quanto verticais (todas as posições, por exemplo). Além disso, é possível trabalhar com blocos retangulares de dados, que podem ser compartilhados entre objetos. `StateList` nada mais é do que uma lista para gerenciamento de objetos `State`.

Essas classes foram projetadas para serem compartilhadas entre componentes cinemáticos, uma vez que é comum que eles sejam calculados a partir de um mesmo conjunto de variáveis e atributos. Tanto helicoides de forma isolada quanto cadeias cinemáticas que são compostas por juntas precisam acessar dados em comum, o que é facilitado pelas implementações feitas em *Kast*.

Além desses dados, todo componente cinemático ainda precisa ter uma matriz que armazena os valores numéricos dos componentes dos helicoides (`_num`). No caso de cadeias fechadas derivadas de *KinematicChain* ainda existe uma matriz de rede (`_netmatrix`). Essas matrizes são todas instâncias de arrays numpy, bem como outras estruturas numéricas auxiliares.

Os atributos de particionamento e sistema de helicoides são definidos por listas. No caso do particionamento, a definição de seu atributo na criação de um componente cinemático é dada por uma lista ou tupla que indica quem é variável primária (1) ou secundária (0). Os sistemas de helicoides são definidos de forma similar, indicando com valores numéricos (1) quais componentes translacionais e rotativos de um helicoide que devem ser calculados.

Por fim, as características de uma linguagem orientada a objetos dinâmica do Python simplificaram várias interfaces de métodos nas implementações. O conceito de *getters* e *setters* permitiu definir propriedades às classes, que se comportam externamente como atributos mas cujos dados são geridos por métodos. A sobrecarga de operadores evitou a criação de novos métodos para realizar operações comuns na linguagem, como comparações, operações aritméticas e de atribuição. Por fim, o fato de ter identificação dinâmica facilitou criar métodos que podiam lidar com diferentes tipos de dados para um mesmo fim, como no caso da definição dos componentes de helicoides.

Essas facilidades são nativas de Python, e sugere-se um estudo geral de suas características ao leitor interessado em estender as funcionalidades do *framework Kast*. Borges (2010) apresenta uma visão geral de várias características da linguagem. O uso de Numpy para engenharia é abordado em (Langtangen, 2009).

B.3.2 Especializando Cadeias Cinemáticas

A especialização de cadeias cinemáticas consiste na criação de classes derivadas de *KinematicChain* a fim de se desenvolver o cálculo completo dos heligiros da cadeia cinemática e preenchimento da sua

matriz de rede. Obviamente, quaisquer outros métodos podem ser reimplementados, se assim desejado.

A vantagem de se especializar é a otimização do cálculo dos valores numéricos da matriz de rede, o que em uma classe generalizada como `KCComposable` normalmente não é possível. Outra característica dessa especialização é a limitação das possibilidades de modificação da cadeia. Isso, porém, pode se tornar uma desvantagem para situações onde a reconfiguração pode ser importante para a execução de tarefas. Além disso, exige o conhecimento prévio do modelo cinemático completo do sistema, ao passo que uma instância de `KCComposable` pode ser definida em um processo iterativo.

A implementação básica de uma classe especializada consiste na definição do construtor da classe (`__init__`) e do método `_update`, que é invocado para calcular a matriz de rede.

Na inicialização, devem ser definidos o número de circuitos independentes (`_lQte`), a quantidade de heligiros (`_sQte`), os atributos como comprimento de elos (`_attr`, instância de `AttributeList`) e as variáveis de juntas (`_q`, instância de `State`). Em relação a estas variáveis, é importante observar a ordem de definição dos nomes dos heligiros, pois esta ordem é obedecida nos demais elementos da classe, como `_num` (a matriz dos heligiros) e `_netmatrix` (a matriz de rede).

O método `_update` deve utilizar os atributos e variáveis de juntas para calcular os valores numéricos dos heligiros, armazenando-os em `_num`. Após, esses valores são copiados para `_netmatrix` multiplicados por 1, -1 ou 0, de acordo com a relação deste heligiro com o circuito da matriz de rede sendo preenchido. A Listagem 1 ilustra a implementação desse método para o UVMS planar descrito na Seção C.3.1.

B.3.3 A Classe `KCComposable`

Para se ter uma implementação funcional de representação de cadeias cinemáticas, foi criada a classe `KCComposable`. Derivada de `KinematicChain`, ela utiliza as facilidades do módulo `NetworkX` de Python para gerenciar grafos de movimento das cadeias modeladas. Através de um objeto dígrafo são relacionados elos e juntas da cadeia. Essa implementação é a mais geral possível, pois a cadeia é dinamicamente montada e passível de modificações na sua estrutura.

Um objeto `KCComposable` deve ser inicialmente instanciado de acordo com a interface definida em `KinematicChain` e a seguir sua estrutura cinemática deve ser definida pela adição de elos e juntas. Os

elos são objetos da classe `Link`, que podem conter atributos como comprimento do elo e posições onde as juntas se conectam. Juntas são instâncias de `Joint`, classe derivada de `Screw` que se comporta como um heligiro, onde são definidos um helicoide normalizado e uma magnitude.

Listagem 1. Implementação de `_update`

```
def _update(self):
    if self._updateLevel > Defs.Refresh:
        self._num.fill(0.0)
        self._num[0,2]=self._num[0,3]=self._num[0,4]= 1.0
        self._num[0,5]=self._num[0,8]=self._num[1,0]= 1.0
        self._num[2,1]=self._num[1,6]=self._num[2,7]= 1.0
        self._num[1,3]= self._attr['xvm']* \
            sin(self._q['qv3',0])
        self._num[2,3]= -self._attr['xvm']* \
            cos(self._q['qv3',0])
        ang= self._q['qv3',0]+self._q['qm1',0]
        self._num[1,4]= self._num[1,3]+ \
            self._attr['lm1']*sin(ang)
        self._num[2,4]= self._num[2,3]- \
            self._attr['lm1']*cos(ang)
        ang+= self._q['qm2',0]
        self._num[1,5]= self._num[1,4]+ \
            self._attr['lm2']*sin(ang)
        self._num[2,5]= self._num[2,4]- \
            self._attr['lm2']*cos(ang)
        ang+= self._q['qm3',0]
        self._num[1,8]= self._num[1,5]+ \
            self._attr['lm3']*sin(ang)
        self._num[2,8]= self._num[2,5]- \
            self._attr['lm3']*cos(ang)
        self._netmatrix[:,0:6]= self._num[:,0:6]
        self._netmatrix[:,6:9]= -self._num[:,6:9]

    self.notify(KinematicChain.updated)
    self._updateLevel = Defs.OK
```

Elos são adicionados pelos métodos `addLink` e `addLinks`, que além do elo ainda pode ter definidos atributos lógicos para indicar se um elo é a base da cadeia ou um efetuador final. Juntas são adicionadas pelos métodos `addJoint` e `addJoints`, e além da junta deve se especificar o par de elos que estão sendo conectados por ela. Essa conexão é

dirigida, do primeiro para o segundo elo especificado, a fim de se gerar posteriormente a matriz de rede da cadeia cinemática fechada.

Como é derivada de `Screw`, as juntas tem seus heligiros normalizados definidos pelo atributo `components`. Este pode ser uma lista de componentes ou uma função que calcule todos os componentes do helicóide. As listas podem conter constantes ou referências a funções que calculem um componente. As referências de funções, sejam para a lista ou para o helicóide completo, podem ser funções normais, membros de classes ou funções lambda (tipo de função Python que é embutida no código em que é invocada).

Uma vez que os elos e juntas são instanciados separadamente, é possível que não compartilhem atributos e variáveis de juntas, o que é fundamental para o correto cálculo da equação de restrição. Assim, o método `pack` deve ser invocado quando uma cadeia `KCComposable` for completada. Esse método gera novas listas de atributos e variáveis reunindo todas as diferentes definições de dados em um conjunto e compartilhando com todos os componentes da cadeia cinemática.

A Listagem 2 contém um trecho de código que demonstra como representar uma cadeia utilizando `KCComposable`. Observa-se que embora o processo de montagem pareça trabalhoso, ele pode ser facilitado por uma interface gráfica com o usuário que gere esse código à medida que o usuário interaja com o software. Por outro lado, é possível criar uma função que instancie toda a classe para que possa ser reutilizada em diversas simulações.

B.3.4 Descrevendo cadeias cinemáticas

Para facilitar a descrição de cadeias cinemáticas e simplificar o seu reuso em implementações computacionais, foi definido um dialeto XML para a sua representação. A versão atual descreve como instâncias de `KComposable` são compostas. Porém, é possível expandir o dialeto para novas classes derivadas de `KinematicChain` à medida em que elas forem projetadas.

A classe `KCFactory` foi desenvolvida para gerar objetos derivados de `KinematicChain` de acordo com descrições fornecidas a ele, implementando o padrão de projeto *factory method*. Este corresponde ao método `kcFrom`, que a partir de um arquivo XML gera o objeto correspondente. O processamento do arquivo XML é feito através do uso dos componentes do módulo `ElementTree` de Python, que permite percorrer uma árvore de definições criada a partir do XML e acessar seu conteúdo.

Listagem 2. Exemplo de uso de KCComposable

```

# Funcoes de calculo dos valores dos helicoides
def h_m2(s):
    s._num[0, 0] = 1.0
    s._num[1, 0] = s._attr['x_vm']*sin(s.q('q_v3')) + \
        s._attr['l_m1']*sin(s.q('q_v3')+s.q('q_m1'))
    s._num[2, 0] = -s._attr['x_vm']*cos(s.q('q_v3')) - \
        s._attr['l_m1']*cos(s.q('q_v3')+s.q('q_m1'))
    vars = ['q_v1', 'q_v2', 'q_v3', 'q_m1', 'q_m2', 'q_m3',
            'q_t1', 'q_t2', 'q_t3']
    idn = Identity(1701, 'UVMS-1', u'UVMS_planar')
    stt = State(vars)
    stt.position = [0.0,0.0,0.0, -0.1745,2.0944,-0.3491,
                   3.9356,3.5321,1.5708]
# Instanciacao da cadeia cinematica
uvms = KCComposable(idn, Defs.xy_plane, state=stt,
                    partitioning=[1,1,1,0,0,0,1,1,1])
# Definicao dos elos
base = Link(Identity(0, u'Base'))
veiculo = Link(Identity(1, u'Veículo'), x_vm=2.65)
efetuador = Link(Identity(2, 'Efetuador'), l_m3=2.0)
l_v1 = Link(Identity(3, 'vx'))
l_v2 = Link(Identity(4, 'vy'))
l_m1 = Link(Identity(5, 'm1'), l_m1=2.0)
l_m2 = Link(Identity(6, 'm2'), l_m2=2.0)
l_t1 = Link(Identity(7, 'tx'))
l_t2 = Link(Identity(8, 'ty'))
uvms.addLink(base, base=True)
uvms.addLinks(veiculo, l_v1, l_v2, l_m1, l_m2, l_t1, l_t2)
uvms.addLink(efetuador, endEffector=True)
# Definicao das algumas juntas
j_v1 = Joint(Identity(11, 'v1', var='q_v1'),
             Defs.xy_plane, type=Defs.translational,
             components=[0.0,0.0,0.0,1.0,0.0,0.0])
uvms.addJoint((j_v1, base, l_v1))
j_m1 = Joint(Identity(14, 'm1', var='q_m1'),
             Defs.xy_plane, components=[0.0,0.0,1.0,
             lambda s: s.attributes['x_vm']*sin(s.q['q_v3',0]),
             lambda s:-s.attributes['x_vm']*cos(s.q['q_v3',0]),
             0.0])
j_m2 = Joint(Identity(15, 'm2', var='q_m2'),
             Defs.xy_plane, components=h_m2)
uvms.addJoints((j_m1, veiculo, l_m1), (j_m2, l_m1, l_m2))
# Integrando todos sob um mesmo conjunto de dados
uvms.pack()

```

Um exemplo de uso é mostrado na Listagem 8. Nela, pode-se ver que a instanciação da cadeia cinemática é feita em uma linha de código.

A Listagem 3 contém trechos de uma descrição de cadeia cinemática em um arquivo XML para ilustrar como utilizar essa facilidade no lugar da definição em código feita na Listagem 2.

B.3.5 Geração de Trajetórias

Os geradores de trajetórias implementados em *Kast* tem por finalidade fornecer as referências para as variáveis primárias da cadeia do sistema de intervenção a partir de descrições de tarefas desejadas para elas. Os geradores implementados utilizam caminhos definidos por pontos que as variáveis devem atingir ao longo do tempo.

Com base no módulo *Numpy* de *Python* foram criadas duas classes derivadas de *TrajectoryFromPath* denominadas *TrajLinearPoly* e *TrajSpline*. A primeira utiliza os componentes de manipulação de polinômios do *Numpy* para interpolação polinomial de 5ª ordem de uma reta entre dois pontos. Para caminhos formados por mais de 2 pontos, são interpoladas retas entre cada dois pontos sequenciais do caminho. A segunda classe utiliza os componentes que definem B-Splines de *Numpy* para interpolar uma trajetória que passe por todos os pontos do caminho especificado para ela. Embora não seja um interpolador de trajetórias, foi criada também uma classe para fornecer referências constantes para as variáveis primárias, denominada *TrajConst*.

Novos geradores podem ser implementados tanto derivados de *Trajectory* quanto de *TrajectoryFromPath*. A diferença da segunda classe é a exigência de uma lista de estados (*StateList*) como atributo adicional, que define o caminho a ser percorrido. As classes de geradores devem implementar o métodos `_update`, que é utilizado para pré-calcular os parâmetros necessários para a geração da trajetória, e `_calcPoint`, usado para calcular um ponto da trajetória em função do instante de tempo ou da iteração desejadas.

O método `_calcPoint` atualiza o atributo de estado `_q` do gerador com os valores de posição e suas derivadas primeira e segunda. Esse método é invocado por todos os métodos que fornecem referência de trajetória como `at`, `generate` e `next`. O primeiro retorna uma referência para o instante de tempo ou iteração passados como parâmetro. O segundo retorna todos os pontos da trajetória com seus instantes de tempo em um *StateList*. `next` é um método especial de *Python* que gera informações sequenciais, de forma que possa ser utilizado em estrutu-

Listagem 3. Exemplo de especificação de trajetória no formato XML

```

<?xml version="1.0" encoding="utf-8"?>
<!-- Listagem incompleta - Apenas para ilustracao -->
<KinematicChain id='1701' name='UMVS-01'
  class='KCComposable'>
  <screwsystem>xy_plane</screwsystem>
  <partitioning>1 1 1 0 0 0 1 1 1</partitioning>
  <attributes><var name='l_v3'>2.65</var>
    <var name='l_m1'>2.0</var> <var name='l_m2'>2.0</var>
    <var name='l_m3'>2.15</var></attributes>
  <state><var name='q_v1'>0.0 0.0 0.0</var>
    <var name='q_m1'>-0.174 0.0 0.0</var>
    <var name='q_m2'>2.094 0.0 0.0</var></state>
  <links><Link id='0' name='Base' base='yes'></Link>
    <Link id='1' name='Veiculo'></Link>
    <Link id='3' name='v1'></Link>
    <Link id='5' name='m1'></Link>
    <Link id='6' name='m2'></Link></links>
  <joints>
    <Joint id='11' name='v1' var='q_v1'
      type='translational'>
      <components>0.0 0.0 0.0 1.0 0.0 0.0</components>
      <linkfrom>Base</linkfrom> <linkto>v1</linkto>
    </Joint>
    <Joint id='14' name='m1' var='q_m1'>
      <components>0.0 0.0 1.0 m1_x m1_y 0.0</components>
      <linkfrom>Veiculo</linkfrom> <linkto>m1</linkto>
      <code name='m1_x' type='lambda'><![CDATA[
        s:s.attributes['l_v3']*sin(s.state['q_v3',0])
      ]]></code>
      <code name='m1_y' type='lambda'><![CDATA[
        s:-s.attributes['l_v3']*cos(s.state['q_v3',0])
      ]]></code>
    </Joint>
    <Joint id='15' name='m2' var='q_m2'>
      <components>h_m2</components>
      <linkfrom>m1</linkfrom> <linkto>m2</linkto>
      <code name='h_m2' type='method'><![CDATA[
        s._num[0,0]= 1.0
        s._num[1,0]= s._attr['l_v3']*sin(s.q('q_v3'))+
          s._attr['l_m1']*sin(s.q('q_v3')+s.q('q_m1'))
        s._num[2,0]= -s._attr['l_v3']*cos(s.q('q_v3'))-
          s._attr['l_m1']*cos(s.q('q_v3')+s.q('q_m1'))
      ]]></code>
    </Joint>
  <codes><imports>
    <import module='math'>sin cos</import>
  </imports></codes>
</KinematicChain>

```

ras de repetição da linguagem. Esses métodos já estão implementados e funcionais nas classes base, desde que existam implementações dos métodos `_calcPoint` e `_update` que possam ser invocadas.

A Listagem 4 apresenta um exemplo básico de uso de geradores de trajetória baseados em caminhos.

Listagem 4. Exemplo de uso de geradores de trajetória

```
v = ['x']
p = StateList(vars, 4)
p[0] = State(vars, array([[0.0, 0.0, 0.0]]), 0.0, 0)
p[1] = State(vars, array([[1.0, 0.0, 0.0]]), 1.5, 15)
p[2] = State(vars, array([[ -0.5, 0.0, 0.0]]), 3.5, 35)
p[3] = State(vars, array([[0.0, 0.0, 0.0]]), 4.0, 40)
trj = TrajLinearPoly(initial=0.0, final=4.0, step=0.1,
                    vars=v, path=p)
texto="t={0},_h={1};_x={2},_dx={3},_ddx={4}"
for q in trj:
    print texto.format(q.t, q.h,
                      q['x', 0], q['x', 1], q['x', 2])
```

O trecho de código descreve como instanciar um gerador derivado de `TrajectoryFromPath` (um `TrajLinearPoly`). Após instanciado, o objeto identificado como `trj` é usado em uma estrutura de repetição `for` e as referências fornecidas são processadas. Cada referência gerada é uma instância de `State`, que além de conter os valores das variáveis e suas derivadas ainda armazena o instante de tempo e o número da iteração.

Para sistematizar e simplificar a especificação de caminhos e trajetórias, foi definida uma sintaxe baseada em XML para descrição de trajetórias. Foi criada uma classe para processar essas descrições e gerar instâncias dos geradores de trajetórias especificados a partir de um *factory method*. Esta classe, denominada `TrajFactory`, processa os dados do arquivo XML passado no método `trajFrom` e retorna uma lista de geradores de trajetória correspondente aos descritos nesse arquivo. Essa classe utiliza os componentes de processamento de dados em XML do módulo `ElementTree` de Python.

A Listagem 5 contém um trecho de código que utiliza o *factory method* para instanciar geradores de trajetória e os utiliza. Na Listagem 6 há um exemplo de descrição de trajetória em arquivo XML.

Listagem 5. Exemplo de uso TrajFactory

```

trjs = TrajFactory.trajFrom('teste.traj.xml')
trj = trjs['xy']
trj._update()
texto="t={0},_h={1};_x={2},_y={3}"
for q in trj:
    print texto.format(q.t, q.h,
                       q['x'],0],q['y'],0])

```

B.4 EXEMPLO DE USO DE KAST

As Listagens 7 e 8 ilustram o uso do *framework* Kast na implementação de simulações. Elas correspondem a trechos de implementação da simulação inicial do primeiro cenário do Capítulo 7. Nesse caso, os componentes de Kast são usados na definição de classes e objetos derivados do *framework* de intervenção apresentado no Capítulo 6.

A classe `Intervention01` é derivada de `Intervention` e implementa seus métodos. O método `init` instancia os objetos necessários à simulação e define seus estados iniciais. A cadeia cinemática é representada pela variável `kc`, que é instanciada pelo *factory method* `kcFrom` da classe `KCFactory`. Ele carrega a definição da cadeia cinemática armazenada no arquivo `uvms01.kc.xml`, que corresponde a um objeto `KCComposable`. Esta cadeia é posteriormente designada como atributo dos objetos `guiagem` e `missao`, que são instâncias das classes `Guidance` e `Mission`, respectivamente.

A tarefa é uma instância de `TaskFromFile`, que cria um objeto especializado da classe `Task` do *framework* de intervenção a partir da definição de um conjunto de geradores de trajetórias especificados em um arquivo XML. Para tanto, `TaskFromFile` utiliza o *factory method* da classe `TrajFactory` de Kast.

A implementação do método `execute` é essencialmente a codificação do algoritmo desenvolvido no Capítulo 6, particularizada para a sintaxe da linguagem Python e fazendo uso de suas características.

A execução da simulação é feita no código da Listagem 7. Essencialmente consiste na instanciação de um objeto `Intervention01` e chamada dos seus métodos na ordem específica. Os dados são armazenados nos arquivos definidos para o objeto `missao`.

Listagem 6. Exemplo de especificação de trajetória no formato XML

```

<?xml version="1.0" encoding="utf-8"?>
<Trajectories id='1000' name='caso1'>
  <about author='Carlos_R_Rocha' email='cticarlo@gmail.com'
    version='20120128-1000' status='unstable' />
  <timeparams initial='0.0' final='25.0' step='0.01' />
  <Trajectory id='1001' name='xy' class='TrajSequence'>
    <timeparams initial='0' final='25' step='0.01' />
    <TrajSeqs>
      <trajseq type='TrajLinearPoly' initial='0' final='1' />
      <trajseq type='TrajConst' initial='1' final='2' />
      <trajseq type='TrajLinearPoly' initial='2' final='3' />
    </TrajSeqs>
    <vars>x y</vars>
    <Path>
      <state t='0.0' h='0'>0 0 0 0 0 0</state>
      <state t='10.0' h='1000'>.65 0 0 .375 0 0</state>
      <state t='20.0' h='2000'>1.65 0 0 .375 0 0</state>
      <state t='25.0' h='2500'>1.94 0 0 .452 0 0</state>
    </Path>
  </Trajectory>
</Trajectories>

```

Listagem 7. Implementação de `_update`

```

intervencao = Intervention01()
intervencao.init(0.0, 32.0, 0.01, comclik=True)
intervencao.execute()
intervencao.end()

```

Listagem 8. Exemplo de classe de intervenção

```

class Intervention01(Intervention):
    """
    Classe que representa a execucao da intervencao
    """
    def __init__(self):
        Intervention.__init__(self)

    def init(self, initial=None, final=None, step=None,
             comclik=True):
        # Define o uvms
        self._uvms = Uvms01()
        # Define a cadeia cinematica
        kc = KCFactory.kcFrom('uvms01.kc.xml')
        varDic = kc.state.nameDict

        # Define a atividade de guiagem
        Kp = array([[0, 0, 0, 0, 0, 0, 100, 100, 10]])
        if comclik:
            self._guiagem = Guidance(kc=kc, kp=Kp,
                                     initial=initial, final=final, step=step)
        else:
            self._guiagem = Guidance(kc=kc, kp=None,
                                     initial=initial, final=final, step=step)

        # Define a tarefa
        self._tarefa = TaskFromFile(vars=varDic,
                                    trajFile='cenario1.caso1.traj.xml')
        # Define a missao
        self._missao = Mission(initial=initial, final=final,
                               step=step, kc=kc, task=self._tarefa)
        self._missao.update(qr=None,
                            a=self._uvms.environment)
        self._missao.recordData(fnqa='atual.csv',
                                fnqd='desejado.csv', fnqr='referencia.csv')

    def end(self):
        # Garante o salvamento do buffer
        self._missao.noRecordData()

    def execute(self):
        for t in self._missao:
            self._missao.verifyState()
            qd = self._missao.taskReference
            qa = self._missao.actualState
            qr = self._guiagem.execute(t, qd, qa)
            a = self._uvms.execute(t, qr)
            self._missao.update(qr, a)
        print t

```

APÊNDICE C – Modelos Cinemáticos Utilizados

Neste apêndice são desenvolvidos os modelos cinemáticos das cadeias utilizadas ao longo da tese. São identificados os parâmetros e variáveis das cadeias junto com sua estrutura cinemática. Quando há possibilidade de composição de cadeias a partir de subcadeias previamente modeladas, segue-se o procedimento descrito no Capítulo 4. As equações de restrição das cadeias dos diferentes cenários analisados na tese são determinadas. Com este apêndice, pretende-se complementar o desenvolvimento dos capítulos anteriores, exemplificar os métodos de modelagem cinemática por helicoides e servir de apoio para a modelagem de outros sistemas de intervenção e seus cenários de utilização.

C.1 CADEIAS CINEMÁTICAS SIMPLES

Essas cadeias são formadas apenas por elos e juntas, sem nenhuma subcadeia identificável. Elas são usadas tanto como cadeias reais (como o manipulador planar RRR que é parte dos UVMS) quanto como cadeias virtuais (como a PPR planar que representa a imposição de movimento no efetuador final do manipulador).

C.1.1 Cadeia Planar PPR

Essa cadeia é formada por elos conectados por duas juntas prismáticas ortogonais entre si e uma junta rotativa. As juntas prismáticas atuam no plano da cadeia e o eixo da junta rotativa é normal a esse plano. A representação funcional do manipulador planar da Figura 87a ilustra uma cadeia desse tipo. As juntas prismáticas estão alinhadas com os eixos \mathbf{x} e \mathbf{y} do sistema de coordenadas de referência. O eixo da junta rotativa é paralelo ao eixo \mathbf{z} do referencial. As variáveis e atributos associados à cadeia são relacionados na Tabela 10.

Tabela 10. Variáveis e atributos da cadeia PPR

Nome	Tipo	Descrição
q_1	Variável	Translação da junta 1
q_2	Variável	Translação da junta 2
q_3	Variável	Rotação da junta 3
l_3	Atributo	Comprimento do elo do efetuador final

Adotando uma configuração de referência como a da Figura 87b, onde as variáveis das juntas assumem os valores $q_1 = 0$, $q_2 = 0$, $q_3 = 0$,

Os helicoides normalizados são calculados como definido na Equação A.3, resultando nas expressões da Equação C.7,

$$\hat{\mathbf{S}}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \hat{\mathbf{S}}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \hat{\mathbf{S}}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ q_2 \\ -q_1 \\ 0 \end{bmatrix} \quad (\text{C.7})$$

que formam a matriz de helicoides normalizados $\hat{\mathbf{S}}_{PPR}$ da Equação C.8 após a exclusão das linhas triviais.

$$\hat{\mathbf{S}}_{PPR} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & q_2 \\ 0 & 1 & -q_1 \end{bmatrix} \quad (\text{C.8})$$

C.1.2 Cadeia Planar RRR

Essa cadeia contém três juntas rotativas que atuam em um mesmo plano. As variáveis e atributos da cadeia são listados na Tabela 13 e uma representação funcional é mostrada na Figura 88a, relativa a um manipulador no plano \mathbf{xy} com os eixos das juntas paralelos ao eixo \mathbf{z} .

Tabela 13. Variáveis e atributos da cadeia RRR

Nome	Tipo	Descrição
q_1	Variável	Rotação da junta 1
q_2	Variável	Rotação da junta 2
q_3	Variável	Rotação da junta 3
l_1	Atributo	Comprimento do elo 1
l_2	Atributo	Comprimento do elo 2
l_3	Atributo	Comprimento do elo do efetuador final

A configuração de referência adotada para a determinação da cinemática de posição é a da Figura 88b, para a qual $q_1 = q_2 = q_3 = 0$. Os parâmetros de Rodrigues correspondentes são listados na Tabela 14.

As matrizes de transformação homogênea das juntas são mostradas nas Equações C.9, C.10 e C.11. A transformação do referencial do ponto \mathbf{e} do efetuador final em relação a um referencial situado na junta 3 é apresentada na Equação C.12. A premultiplicação destas ma-

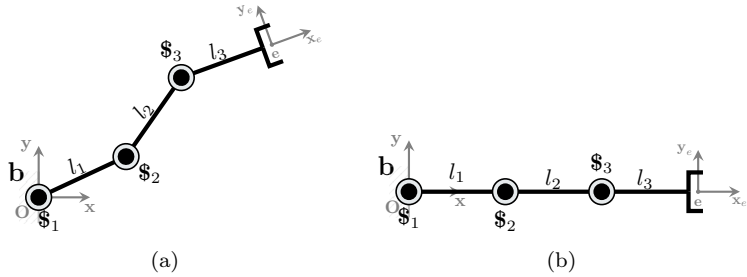


Figura 88. Manipulador planar RRR: (a)representação funcional; (b)configuração de referência

Tabela 14. Parâmetros de Rodrigues

Junta	s	s_0	t	θ
1	$(0, 0, 1)$	$(0, 0, 0)$	—	q_1
2	$(0, 0, 1)$	$(l_1, 0, 0)$	—	q_2
3	$(0, 0, 1)$	$(l_1 + l_2, 0, 0)$	—	q_3

trizes resulta na transformação homogênea entre o referencial da base e o referencial e mostrada na Equação C.13¹.

$$\mathbf{A}_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.9})$$

$$\mathbf{A}_2 = \begin{bmatrix} c_2 & -s_2 & 0 & -l_1(c_2 - 1) \\ s_2 & c_2 & 0 & -l_1 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.10})$$

$$\mathbf{A}_3 = \begin{bmatrix} c_3 & -s_3 & 0 & -(l_1 + l_2)(c_3 - 1) \\ s_3 & c_3 & 0 & -(l_1 + l_2) s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.11})$$

¹Onde c_{123} e s_{123} são as funções cosseno e seno da soma dos ângulos de juntas q_1 , q_2 e q_3 . Essa convenção será utilizada no restante do texto.

$$\mathbf{A}_e = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.12})$$

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3 \mathbf{A}_e \\ &= \begin{bmatrix} c_{123} & -s_{123} & 0 & (l_3 - l_1 - l_2) c_{123} + l_2 c_{12} + l_1 c_1 \\ s_{123} & c_{123} & 0 & (l_3 - l_1 - l_2) s_{123} + l_2 s_{12} + l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (\text{C.13})$$

A transformação de helicoides ${}^b\mathbf{T}_e$ entre os referenciais do efetuador final e da base é então determinada como na Equação C.14,

$${}^b\mathbf{T}_e = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ 0 & 0 & T_{43} \\ 0 & 0 & T_{53} & \mathbf{R} \\ T_{61} & T_{62} & 0 \end{bmatrix} \quad (\text{C.14})$$

$$\mathbf{R} = \begin{bmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

onde os componentes T_{ij} são iguais a

$$T_{43} = (l_3 - l_2 - l_1) s_{123} + l_2 s_{12} + l_1 s_1 \quad (\text{C.15})$$

$$T_{53} = -((l_3 - l_2 - l_1) c_{123} + l_2 c_{12} + l_1 c_1) \quad (\text{C.16})$$

$$T_{61} = (l_1 c_1 + l_2 c_{12}) s_{123} - (l_1 s_1 + l_2 s_{12}) c_{123} \quad (\text{C.17})$$

$$T_{62} = (l_1 s_1 + l_2 s_{12}) s_{123} + (l_1 c_1 + l_2 c_{12}) c_{123} + l_3 - l_2 - l_1 \quad (\text{C.18})$$

Para determinar a cinemática diferencial são utilizados os valores dos vetores \mathbf{s} e \mathbf{s}_0 para qualquer postura da cadeia listados na Tabela 15 junto com o passo do helicóide.

A matriz de helicoides normalizados $\hat{\mathbf{S}}_{RRR}$ da cadeia é mostrada na Equação C.19. Ela é formada pelos helicoides normalizados das juntas da cadeia, tendo suas linhas triviais excluídas.

Tabela 15. Parâmetros dos helicoides normalizados

Junta	\mathbf{s}	\mathbf{s}_0	h
1	(0, 0, 1)	(0, 0, 0)	0
2	(0, 0, 1)	($l_1 c_1, l_1 s_1, 0$)	0
3	(0, 0, 1)	($l_1 c_1 + l_2 c_{12}, l_1 s_1 + l_2 s_{12}, 0$)	0

$$\hat{\mathbf{S}}_{RRR} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & l_1 s_1 & l_1 s_1 + l_2 s_{12} \\ 0 & -l_1 c_1 & -(l_1 c_1 + l_2 c_{12}) \end{bmatrix} \quad (\text{C.19})$$

C.2 UVMS COM CADEIAS CINEMÁTICAS COMPOSTAS

A representação de um sistema veículo-manipuladores subaquáticos por cadeias cinemáticas pode ser feita por composição. O movimento do veículo pode ser associado com o do efetuador final de uma cadeia PPR (no caso planar) ou de uma cadeia PPPS (no caso espacial), alinhadas com os eixos de um referencial inercial. Os manipuladores, por sua vez, são naturalmente representados por cadeias cinemáticas reais de acordo com seus arranjos cinemáticos.

C.2.1 UVMS Planar Com Um Manipulador RRR

Esse sistema de intervenção subaquática é formado por um veículo ao qual está vinculado um manipulador planar de três graus de liberdade. Assume-se que as tarefas são executadas em uma profundidade constante, o que define um plano horizontal de atuação. A Figura 89 mostra uma representação funcional do sistema.

O movimento do veículo é representado por uma cadeia virtual planar PPR denominada v , enquanto a cadeia cinemática m , do manipulador, é do tipo RRR. A composição das definições prévias das cadeias cinemáticas PPR e RRR resulta na Equação C.20, expressa no referencial da base da cadeia v . Assume-se que o referencial do manipulador coincide com o efetuador final da cadeia do veículo.

$$\mathbf{S}_u = \mathbf{S}_v + {}^I\mathbf{T}_m \mathbf{S}_m = \begin{bmatrix} \hat{\mathbf{S}}_v & {}^I\mathbf{T}_m \hat{\mathbf{S}}_m \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_v \\ \dot{\mathbf{q}}_m \end{bmatrix} = \hat{\mathbf{S}}_u \dot{\mathbf{q}}_u \quad (\text{C.20})$$

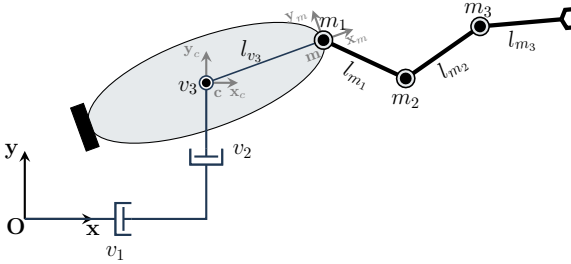


Figura 89. UVMS planar com 1 manipulador

As variáveis das cadeias são renomeadas de forma a incorporar em seus subscritos os identificadores da subcadeia da qual fazem parte, como é mostrado na Tabela 16, para evitar conflitos. O referencial do veículo coincide com o referencial inercial, motivo pelo qual se utiliza o superscrito I na transformação de helicoides. Esta transformação é definida na Equação C.6.

Tabela 16. Variáveis e atributos do UVMS planar com um manipulador

Nome	Tipo	Descrição
q_{v_1}	Variável	Translação da junta 1 do veículo
q_{v_2}	Variável	Translação da junta 2 do veículo
q_{v_3}	Variável	Rotação da junta 3 do veículo
l_{v_3}	Atributo	Comprimento do elo final do veículo
q_{m_1}	Variável	Rotação da junta 1 do manipulador
q_{m_2}	Variável	Rotação da junta 2 do manipulador
q_{m_3}	Variável	Rotação da junta 3 do manipulador
l_{m_1}	Atributo	Comprimento do elo 1 do manipulador
l_{m_2}	Atributo	Comprimento do elo 2 do manipulador
l_{m_3}	Atributo	Comprimento do elo 3 do manipulador

É possível simplificar as expressões dos componentes dos heligiros com a escolha de um referencial diferente. O referencial c em questão situa-se no centro de massa do veículo (onde a junta rotativa de v está posicionada) estando porém alinhado com o referencial inercial. A transformação entre os referenciais I e c é determinada pela translação entre as origens desses referenciais, como mostrado na Equação C.21.

$${}^c\mathbf{T}_I = \begin{bmatrix} & \mathbf{I}_3 & & \mathbf{0} \\ 0 & 0 & -q_{v_2} & \\ 0 & 0 & q_{v_1} & \mathbf{I}_3 \\ q_{v_2} & -q_{v_1} & 0 & \end{bmatrix} \quad (\text{C.21})$$

Aplicando essa transformação na matriz de helicoides normalizados na Equação C.20, e após simplificações, obtém-se a matriz de helicoides normalizados expressos segundo o referencial c descrito na Equação C.22,

$$\begin{aligned} {}^c\hat{\mathbf{\$}}_u &= {}^c\mathbf{T}_I \hat{\mathbf{\$}}_u \\ &= \begin{bmatrix} {}^c\hat{\mathbf{\$}}_{v_1} & {}^c\hat{\mathbf{\$}}_{v_2} & {}^c\hat{\mathbf{\$}}_{v_3} & {}^c\hat{\mathbf{\$}}_{m_1} & {}^c\hat{\mathbf{\$}}_{m_2} & {}^c\hat{\mathbf{\$}}_{m_3} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & m_{1_x} & m_{2_x} & m_{3_x} \\ 0 & 1 & 0 & m_{1_y} & m_{2_y} & m_{3_y} \end{bmatrix} \end{aligned} \quad (\text{C.22})$$

onde

$$m_{1_x} = l_{v_3} s_{v_3} \quad (\text{C.23})$$

$$m_{1_y} = -l_{v_3} c_{v_3} \quad (\text{C.24})$$

$$m_{2_x} = l_{v_3} s_{v_3} + l_{m_1} s_{v_3 m_1} \quad (\text{C.25})$$

$$m_{2_y} = -l_{v_3} c_{v_3} - l_{m_1} c_{v_3 m_1} \quad (\text{C.26})$$

$$m_{3_x} = l_{v_3} s_{v_3} + l_{m_1} s_{v_3 m_1} + l_{m_2} s_{v_3 m_{12}} \quad (\text{C.27})$$

$$m_{3_y} = -l_{v_3} c_{v_3} - l_{m_1} c_{v_3 m_1} - l_{m_2} c_{v_3 m_{12}} \quad (\text{C.28})$$

C.2.2 UVMS Planar Com Dois Manipuladores RRR

Da mesma forma que o sistema anterior, assume-se que as tarefas são executadas em um plano horizontal. O fato de ter dois manipuladores leva ao arranjo apresentado na Figura 90. Nele, os manipuladores estão afastados de uma distância y_{m_i} em relação ao efetuador final da cadeia virtual v , onde $i = 1, 2$ é o número do manipulador. As variáveis e atributos da cadeia cinemática são listados na Tabela 17.

A definição dos heligiros da cadeia é similar ao caso com um manipulador. Porém, deve-se observar que cada manipulador tem um sistema de referência diferente do sistema do efetuador final da cadeia

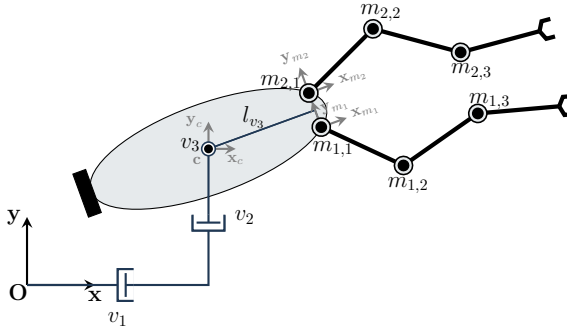


Figura 90. UVMS planar com 2 manipuladores

v , o que faz com que os heligiros de ambas as subcadeias sofram uma transformação de helicoides correspondente à translação de origens de sistemas de coordenadas. Assim, o estado de velocidades da cadeia segundo o referencial c assume a forma da Equação C.29, onde ${}^c\mathbf{T}_{m_1}$ e ${}^c\mathbf{T}_{m_2}$ são definidas nas Equações C.30 e C.31 respectivamente.

$$\begin{aligned} {}^c\mathcal{S}_u &= {}^c\mathcal{S}_v + {}^c\mathbf{T}_{m_1}\mathcal{S}_{m_1} + {}^c\mathbf{T}_{m_2}\mathcal{S}_{m_2} \\ &= \begin{bmatrix} {}^c\hat{\mathcal{S}}_v & {}^c\mathbf{T}_{m_1}\hat{\mathcal{S}}_{m_1} & {}^c\mathbf{T}_{m_2}\hat{\mathcal{S}}_{m_2} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_v \\ \dot{\mathbf{q}}_{m_1} \\ \dot{\mathbf{q}}_{m_2} \end{bmatrix} = \hat{\mathcal{S}}_u\dot{\mathbf{q}}_u \quad (\text{C.29}) \end{aligned}$$

$${}^c\mathbf{T}_{m_1} = \begin{bmatrix} c_{v_3} & -s_{v_3} & 0 & 0 & 0 & 0 \\ s_{v_3} & c_{v_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & y_{m_1}c_{v_3} + l_{v_3}s_{v_3} & c_{v_3} & -s_{v_3} & 0 \\ 0 & 0 & y_{m_1}s_{v_3} - l_{v_3}c_{v_3} & s_{v_3} & c_{v_3} & 0 \\ -y_{m_1} & l_{v_3} & 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.30})$$

$${}^c\mathbf{T}_{m_2} = \begin{bmatrix} c_{v_3} & -s_{v_3} & 0 & 0 & 0 & 0 \\ s_{v_3} & c_{v_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & y_{m_2}c_{v_3} + l_{v_3}s_{v_3} & c_{v_3} & -s_{v_3} & 0 \\ 0 & 0 & y_{m_2}s_{v_3} - l_{v_3}c_{v_3} & s_{v_3} & c_{v_3} & 0 \\ -y_{m_2} & l_{v_3} & 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{C.31})$$

A matriz de helicoides normalizados obtida a partir da Equação C.29 é apresentada na Equação C.32,

Tabela 17. Variáveis e atributos do UVMS planar com dois manipuladores

Nome	Tipo	Descrição
q_{v_1}	Variável	Translação da junta 1 do veículo
q_{v_2}	Variável	Translação da junta 2 do veículo
q_{v_3}	Variável	Rotação da junta 3 do veículo
l_{v_3}	Atributo	Comprimento do elo final do veículo
y_{m_1}	Atributo	Distância da base do manipulador 1 ao eixo de avanço do veículo
y_{m_2}	Atributo	Distância da base do manipulador 2 ao eixo de avanço do veículo
$q_{m_{1,1}}$	Variável	Rotação da junta 1 do manipulador 1
$q_{m_{1,2}}$	Variável	Rotação da junta 2 do manipulador 1
$q_{m_{1,3}}$	Variável	Rotação da junta 3 do manipulador 1
$l_{m_{1,1}}$	Atributo	Comprimento do elo 1 do manipulador 1
$l_{m_{1,2}}$	Atributo	Comprimento do elo 2 do manipulador 1
$l_{m_{1,3}}$	Atributo	Comprimento do elo 3 do manipulador 1
$q_{m_{2,1}}$	Variável	Rotação da junta 1 do manipulador 2
$q_{m_{2,2}}$	Variável	Rotação da junta 2 do manipulador 2
$q_{m_{2,3}}$	Variável	Rotação da junta 3 do manipulador 2
$l_{m_{2,1}}$	Atributo	Comprimento do elo 1 do manipulador 2
$l_{m_{2,2}}$	Atributo	Comprimento do elo 2 do manipulador 2
$l_{m_{2,3}}$	Atributo	Comprimento do elo 3 do manipulador 2

$$\begin{aligned}
c\hat{\mathcal{S}}_u &= \left[c\hat{\mathcal{S}}_{v_1} \quad c\hat{\mathcal{S}}_{v_2} \quad c\hat{\mathcal{S}}_{v_3} \quad c\hat{\mathcal{S}}_{m_{1,1}} \quad c\hat{\mathcal{S}}_{m_{1,2}} \quad c\hat{\mathcal{S}}_{m_{1,3}} \quad c\hat{\mathcal{S}}_{m_{2,1}} \quad c\hat{\mathcal{S}}_{m_{2,2}} \quad c\hat{\mathcal{S}}_{m_{2,3}} \right] \\
&= \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & m_{1,1x} & m_{1,2x} & m_{1,3x} & m_{2,1x} & m_{2,2x} & m_{2,3x} \\ 0 & 1 & 0 & m_{1,1y} & m_{1,2y} & m_{1,3y} & m_{2,1y} & m_{2,2y} & m_{2,3y} \end{bmatrix}
\end{aligned} \tag{C.32}$$

onde

$$m_{1,1x} = l_{v_3} s_{v_3} + y_{m_1} c_{v_3} \tag{C.33}$$

$$m_{1,1y} = -l_{v_3} c_{v_3} + y_{m_1} s_{v_3} \tag{C.34}$$

$$m_{1,2x} = m_{1,1x} + l_{m_{1,1}} s_{v_3} m_{1,1} \tag{C.35}$$

$$m_{1,2y} = m_{1,1y} - l_{m_{1,1}} c_{v_3} m_{1,1} \tag{C.36}$$

$$m_{1,3x} = m_{1,2x} + l_{m_{1,2}} s_{v_3} m_{1,12} \quad (\text{C.37})$$

$$m_{1,3y} = m_{1,2x} - l_{m_{1,2}} c_{v_3} m_{1,12} \quad (\text{C.38})$$

$$m_{2,1x} = l_{v_3} s_{v_3} + y_{m_2} c_{v_3} \quad (\text{C.39})$$

$$m_{2,1y} = -l_{v_3} c_{v_3} + y_{m_2} s_{v_3} \quad (\text{C.40})$$

$$m_{2,2x} = m_{2,1x} + l_{m_{2,1}} s_{v_3} m_{2,1} \quad (\text{C.41})$$

$$m_{2,2y} = m_{2,1y} - l_{m_{2,1}} c_{v_3} m_{2,1} \quad (\text{C.42})$$

$$m_{2,3x} = m_{2,2x} + l_{m_{2,2}} s_{v_3} m_{2,12} \quad (\text{C.43})$$

$$m_{2,3y} = m_{2,2x} - l_{m_{2,2}} c_{v_3} m_{2,12} \quad (\text{C.44})$$

C.3 CENÁRIOS DE OPERAÇÃO

Assim como nos modelos cinemáticos de UVMS, as cadeias cinemáticas dos cenários onde eles são empregados podem ser formadas por composição. No caso, pelas cadeias dos próprios UVMS e de cadeias virtuais que impõem movimentos e restrições que são definidos pelas tarefas e pelo espaço de trabalho.

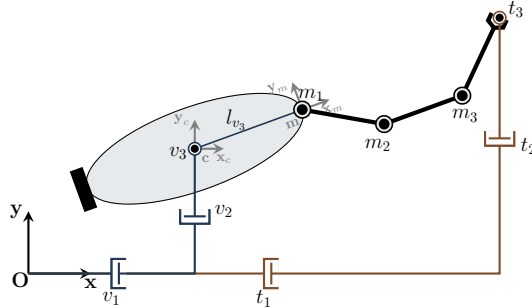
C.3.1 UVMS Planar Com Um Manipulador

Para uma operação de intervenção com apenas um manipulador, utiliza-se o modelo de UVMS descrito na Subseção C.2.1, onde o movimento do efetuador final é representado por uma cadeia virtual do tipo PPR. Essa cadeia, denominada t , é usada para impor o movimento ao efetuador final, fechando o circuito e possibilitando o uso do método de Davies para a resolução da cinemática. A Figura 91a mostra uma representação funcional do sistema, enquanto o grafo de movimento correspondente é mostrado na Figura 91b.

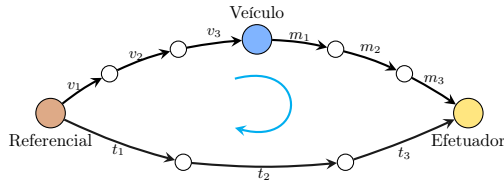
Ao se incluir a cadeia virtual t ao sistema, acrescentam-se três novos helicoides normalizados à matriz definida na Equação C.22. A definição destes é similar à da cadeia v por terem a mesma estrutura². Segundo o referencial c , os heligiros da cadeia t formam a matriz apresentada na Equação C.45.

²Outra visão seria a dela estar vinculada ao efetuador final do manipulador, o que implicaria em diferentes transformações de helicoides.

$${}^c\hat{\mathbf{s}}_t = {}^c\mathbf{T}_I\hat{\mathbf{s}}_t = \begin{bmatrix} \hat{\mathbf{s}}_{t_1} & \hat{\mathbf{s}}_{t_2} & \hat{\mathbf{s}}_{t_3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & qt_2 - q_{v_2} \\ 0 & 1 & -qt_1 + q_{v_1} \end{bmatrix} \quad (\text{C.45})$$



(a)



(b)

Figura 91. Execução de tarefas com um UVMS planar com um manipulador: (a)representação funcional; (b)grafo de movimento

É possível relacionar geometricamente as variáveis de posição da cadeia t com as das cadeias v e m , como mostrado na Equação C.46.

$$\mathbf{q}_t = \begin{bmatrix} q_{t_1} \\ q_{t_2} \\ q_{t_3} \end{bmatrix} = \begin{bmatrix} q_{v_1} + l_{v_3}c_{v_3} + l_{m_1}c_{v_3m_1} + l_{m_2}c_{v_3m_{12}} + l_{m_3}c_{v_3m_{123}} \\ q_{v_2} + l_{v_3}s_{v_3} + l_{m_1}s_{v_3m_1} + l_{m_2}s_{v_3m_{12}} + l_{m_3}s_{v_3m_{123}} \\ q_{v_3} + q_{m_1} + q_{m_2} + q_{m_3} \end{bmatrix} \quad (\text{C.46})$$

Substituindo essas relações na Equação C.45, definem-se os helicoides normalizados de t em função do mesmo conjunto de variáveis que

os das cadeias v e m . A matriz de helicoides normalizados do sistema completo é mostrada na Equação C.47,

$$\mathbf{D} = \begin{bmatrix} c\hat{\mathcal{S}}_{v_1} & c\hat{\mathcal{S}}_{v_2} & c\hat{\mathcal{S}}_{v_3} & c\hat{\mathcal{S}}_{m_1} & c\hat{\mathcal{S}}_{m_2} & c\hat{\mathcal{S}}_{m_3} & c\hat{\mathcal{S}}_{t_1} & c\hat{\mathcal{S}}_{t_2} & c\hat{\mathcal{S}}_{t_3} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & m_{1_x} & m_{2_x} & m_{3_x} & 1 & 0 & t_{3_x} \\ 0 & 1 & 0 & m_{1_y} & m_{2_y} & m_{3_y} & 0 & 1 & t_{3_y} \end{bmatrix} \quad (\text{C.47})$$

onde

$$t_{3_x} = l_{v_3}s_{v_3} + l_{m_1}s_{v_3m_1} + l_{m_2}s_{v_3m_{12}} + l_{m_3}s_{v_3m_{123}} \quad (\text{C.48})$$

$$t_{3_y} = -l_{v_3}c_{v_3} - l_{m_1}c_{v_3m_1} - l_{m_2}c_{v_3m_{12}} - l_{m_3}c_{v_3m_{123}} \quad (\text{C.49})$$

A partir do grafo de movimento da Figura 91b, determina-se a matriz de rede \mathbf{B} ,

$$\mathbf{B} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad -1] \quad (\text{C.50})$$

a qual, aplicada na Equação A.17 juntamente com a matriz de helicoides normalizados, resulta na matriz de rede \mathbf{N} descrita na Equação C.51.

$$\mathbf{N} = [\hat{\mathcal{S}}_{v_1} \quad \hat{\mathcal{S}}_{v_2} \quad \hat{\mathcal{S}}_{v_3} \quad \hat{\mathcal{S}}_{m_1} \quad \hat{\mathcal{S}}_{m_2} \quad \hat{\mathcal{S}}_{m_3} \quad -\hat{\mathcal{S}}_{t_1} \quad -\hat{\mathcal{S}}_{t_2} \quad -\hat{\mathcal{S}}_{t_3}] \quad (\text{C.51})$$

Para executar operações em um espaço livre de obstáculos onde o veículo permanece estacionário (ou tem algum movimento previamente determinado), pode-se usar o particionamento da equação de restrição da Equação C.52 para resolver a cinemática para o manipulador a partir das especificações da tarefa pela cadeia t e as restrições do veículo impostas pela cadeia v .

$$\mathbf{N}\dot{\mathbf{q}} = \left[\hat{\mathcal{S}}_{v_1} \quad \hat{\mathcal{S}}_{v_2} \quad \hat{\mathcal{S}}_{v_3} \quad \hat{\mathcal{S}}_{m_1} \quad \hat{\mathcal{S}}_{m_2} \quad \hat{\mathcal{S}}_{m_3} \quad -\hat{\mathcal{S}}_{t_1} \quad -\hat{\mathcal{S}}_{t_2} \quad -\hat{\mathcal{S}}_{t_3} \right] \begin{bmatrix} \dot{q}_{v_1} \\ \dot{q}_{v_2} \\ \dot{q}_{v_3} \\ \dot{q}_{m_1} \\ \dot{q}_{m_2} \\ \dot{q}_{m_3} \\ \dot{q}_{t_1} \\ \dot{q}_{t_2} \\ \dot{q}_{t_3} \end{bmatrix} = \mathbf{0} \quad (\text{C.52})$$

C.3.2 UVMS Planar Com Dois Manipuladores Em Cooperação

Nesse tipo de tarefa onde dois manipuladores operam em cooperação, a cadeia cinemática do sistema tem complexidade maior que a do caso anterior. Além do modelo cinemático do UVMS descrito na Subseção C.2.2, são acrescentadas 3 cadeias virtuais. A cadeia t representa o movimento da peça sendo manipulada, enquanto as cadeias r_1 e r_2 representam a posição relativa dos efetuadores finais do UVMS em relação ao ponto da peça vinculado à cadeia t . A Figura 92a mostra uma representação funcional do sistema, enquanto o grafo de movimento correspondente é mostrado na Figura 92b. Ele é uma expansão do grafo contraído da Figura 23.

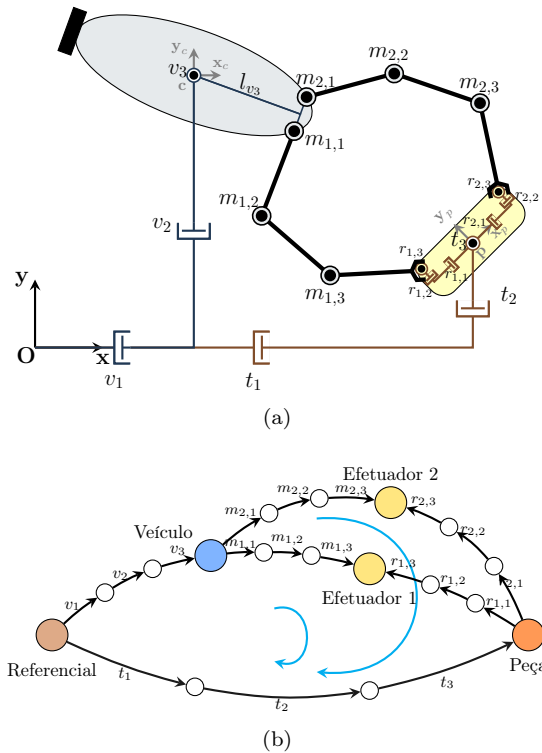


Figura 92. UVMS planar com dois manipuladores trabalhando em cooperação: (a) representação funcional; (b) grafo de movimento

A matriz de helicoides normalizados do sistema \mathbf{D} é então formada pelos helicoides do UVMS e pelos helicoides das subcadeias t , r_1 e r_2 , todas do tipo PPR. O referencial adotado para o sistema é o c . Para expressar os heligiros normalizados de acordo com ele são usadas as transformações de helicoides mostradas na Equação C.53,

$$\mathbf{D} = \left[\begin{array}{cccc} {}^c\hat{\mathbf{S}}_u & {}^c\hat{\mathbf{S}}_t & {}^c\mathbf{T}_p^p\hat{\mathbf{S}}_{r_1} & {}^c\mathbf{T}_p^p\hat{\mathbf{S}}_{r_2} \end{array} \right] \quad (\text{C.53})$$

onde ${}^c\hat{\mathbf{S}}_u$ é o conjunto de helicoides normalizados do UVMS definidos na Equação C.32 e ${}^c\hat{\mathbf{S}}_t$ tem a mesma definição da Equação C.45. A transformação de helicoides ${}^c\mathbf{T}_p$ ocorre do referencial solidário à peça em manipulação, com origem no ponto em que a cadeia t se conecta a ele (que é no qual os heligiros de r_1 e r_2 estão definidos), para o referencial da cadeia c . Ela é definida como ${}^c\mathbf{T}_p = {}^c\mathbf{T}_I^I\mathbf{T}_p$, onde a transformação de helicoides ${}^I\mathbf{T}_p$ é definida como na Equação C.54.

$${}^I\mathbf{T}_p = \left[\begin{array}{cccccc} c_{t_3} & -s_{t_3} & 0 & & & \\ s_{t_3} & c_{t_3} & 0 & & \mathbf{0} & \\ 0 & 0 & 1 & & & \\ 0 & 0 & q_{t_2} & c_{t_3} & -s_{t_3} & 0 \\ 0 & 0 & -q_{t_1} & s_{t_3} & c_{t_3} & 0 \\ q_{t_1}s_{t_3} - q_{t_2}c_{t_3} & q_{t_1}c_{t_3} + q_{t_2}s_{t_3} & 0 & 0 & 0 & 1 \end{array} \right] \quad (\text{C.54})$$

Assim, a transformação ${}^c\mathbf{T}_p$ é igual a

$${}^c\mathbf{T}_p = \left[\begin{array}{cccccc} c_{t_3} & -s_{t_3} & 0 & & & \\ s_{t_3} & c_{t_3} & 0 & & \mathbf{0} & \\ 0 & 0 & 1 & & & \\ 0 & 0 & q_{t_2} - q_{v_2} & c_{t_3} & -s_{t_3} & 0 \\ 0 & 0 & -q_{t_1} + q_{v_1} & s_{t_3} & c_{t_3} & 0 \\ T_{6,1} & T_{6,2} & 0 & 0 & 0 & 1 \end{array} \right] \quad (\text{C.55})$$

$$T_{6,1} = (q_{t_1} - q_{v_1})s_{t_3} - (q_{t_2} - q_{v_2})c_{t_3} \quad (\text{C.56})$$

$$T_{6,2} = (q_{t_1} - q_{v_1})c_{t_3} + (q_{t_2} - q_{v_2})s_{t_3} \quad (\text{C.57})$$

Com isso, ${}^c\hat{\mathbf{S}}_{r_1}$ e ${}^c\hat{\mathbf{S}}_{r_2}$ são iguais a

$${}^c\hat{\mathbb{S}}_{r_1} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{r_{1,1}}s_{t_3} + q_{r_{1,2}}c_{t_3} + q_{t_2} - q_{v_2} \\ s_{t_3} & c_{t_3} & -q_{r_{1,1}}c_{t_3} + q_{r_{1,2}}s_{t_3} - q_{t_1} + q_{v_1} \end{bmatrix} \quad (\text{C.58})$$

$${}^c\hat{\mathbb{S}}_{r_2} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{r_{2,1}}s_{t_3} + q_{r_{2,2}}c_{t_3} + q_{t_2} - q_{v_2} \\ s_{t_3} & c_{t_3} & -q_{r_{2,1}}c_{t_3} + q_{r_{2,2}}s_{t_3} - q_{t_1} + q_{v_1} \end{bmatrix} \quad (\text{C.59})$$

A partir do grafo de movimento da Figura 92b, e seguindo a ordem dos helicoides normalizados da matriz \mathbf{D} , determina-se a matriz de rede \mathbf{B} ,

$$\mathbf{B} = \begin{bmatrix} \mathbf{1}_3 & \mathbf{1}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & -\mathbf{1}_3 & \mathbf{0}_3 \\ \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{1}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & -\mathbf{1}_3 \end{bmatrix} \quad (\text{C.60})$$

onde $\mathbf{1}_c$ é um vetor de 1's de dimensão $[1 \times c]$ e $\mathbf{0}_c$ é um vetor de 0's de dimensão $[1 \times c]$. A matriz de rede resultante da substituição de \mathbf{D} e \mathbf{B} na Equação A.17 é mostrada na Equação C.61.

$$\mathbf{N} = \begin{bmatrix} {}^c\hat{\mathbb{S}}_v & {}^c\hat{\mathbb{S}}_{m_1} & \mathbf{0} & -{}^c\hat{\mathbb{S}}_t & -{}^c\hat{\mathbb{S}}_{r_1} & \mathbf{0} \\ {}^c\hat{\mathbb{S}}_v & \mathbf{0} & {}^c\hat{\mathbb{S}}_{m_2} & -{}^c\hat{\mathbb{S}}_t & \mathbf{0} & -{}^c\hat{\mathbb{S}}_{r_2} \end{bmatrix} \quad (\text{C.61})$$

Para executar operações em um espaço livre de obstáculos onde o veículo permanece estacionário (ou tem algum movimento previamente determinado), pode-se usar o particionamento da equação de restrição da Equação C.62 para resolver a cinemática para os manipuladores a partir das especificações da tarefa pelas cadeias t , r_1 e r_2 e as restrições do veículo impostas pela cadeia v .

$$\mathbf{N}\dot{\mathbf{q}} = \begin{bmatrix} \hat{\mathbb{S}}_v & \hat{\mathbb{S}}_{m_1} & \mathbf{0} & -\hat{\mathbb{S}}_t & -\hat{\mathbb{S}}_{r_1} & \mathbf{0} \\ \hat{\mathbb{S}}_v & \mathbf{0} & \hat{\mathbb{S}}_{m_2} & -\hat{\mathbb{S}}_t & \mathbf{0} & -\hat{\mathbb{S}}_{r_2} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_v \\ \dot{\mathbf{q}}_{m_1} \\ \dot{\mathbf{q}}_{m_2} \\ \dot{\mathbf{q}}_t \\ \dot{\mathbf{q}}_{r_1} \\ \dot{\mathbf{q}}_{r_2} \end{bmatrix} = \mathbf{0} \quad (\text{C.62})$$

C.3.3 Dois UVMS Planares Com Um Manipulador Cada Em Cooperação

Um cenário de operação cooperativa entre UVMS é apresentado na Figura 93a. Nesse caso, cada veículo tem um manipulador planar

acoplado, formando o UVMS descrito na subseção C.2.1. A descrição de uma tarefa consiste na definição do movimento desejado para o objeto a ser manipulado através da cadeia virtual t e das posturas dos efetadores finais dos UVMS em relação ao objeto representadas pelas cadeias virtuais r_1 e r_2 . O grafo de movimento desse cenário é mostrado na Figura 93b.

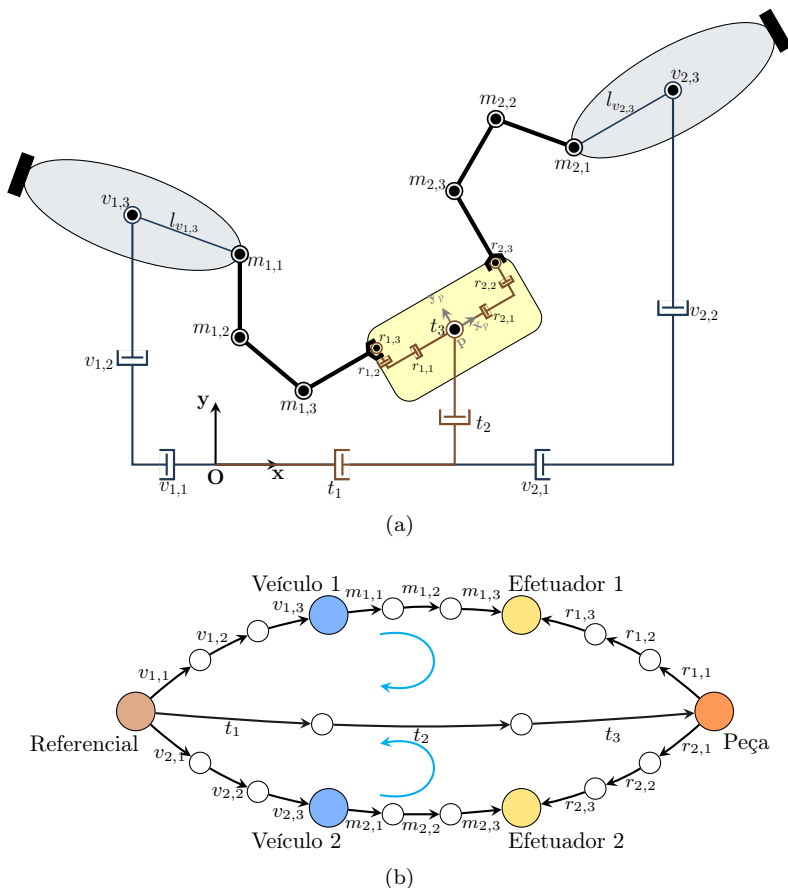


Figura 93. Execução de tarefa cooperativa entre dois UVMS planares: (a)representação funcional; (b)grafo de movimento

A matriz de helicoides normalizados dessa cadeia cinemática é obtida pela composição das cadeias dos dois UVMS (u_1 e u_2) além das subcadeias t , r_1 e r_2 . O referencial adotado nesse cenário é o inercial,

visto ser comum aos dois UVMS e ao objeto sendo manipulado. Assim, a matriz \mathbf{D} assume a forma da Equação C.63,

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{S}}_{u_1} & \hat{\mathbf{S}}_{u_2} & \hat{\mathbf{S}}_t & {}^I\mathbf{T}_p{}^p\hat{\mathbf{S}}_{r_1} & {}^I\mathbf{T}_p{}^p\hat{\mathbf{S}}_{r_2} \end{bmatrix} \quad (\text{C.63})$$

para ${}^I\mathbf{T}_p$ definida na Equação C.54.

A partir do grafo de movimento determina-se a matriz de circuitos \mathbf{B} que é descrita na Equação C.64,

$$\mathbf{B} = \begin{bmatrix} \mathbf{1}_6 & \mathbf{0}_6 & -\mathbf{1}_3 & -\mathbf{1}_3 & \mathbf{0}_3 \\ \mathbf{0}_6 & \mathbf{1}_6 & -\mathbf{1}_3 & \mathbf{0}_3 & -\mathbf{1}_3 \end{bmatrix} \quad (\text{C.64})$$

Com \mathbf{D} e \mathbf{B} determinados, calcula-se a matriz de rede pela Equação A.17, que resulta na Equação C.65.

$$\mathbf{N} = \begin{bmatrix} \hat{\mathbf{S}}_{u_1} & \mathbf{0} & -\hat{\mathbf{S}}_t & -\hat{\mathbf{S}}_{r_1} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{S}}_{u_2} & -\hat{\mathbf{S}}_t & \mathbf{0} & -\hat{\mathbf{S}}_{r_2} \end{bmatrix} \quad (\text{C.65})$$

Os componentes da matriz de rede são expressos segundo o referencial inercial como

$$\hat{\mathbf{S}}_{u_1} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & q_{v_{1,2}} & m_{1,1x} & m_{1,2x} & m_{1,3x} \\ 0 & 1 & -q_{v_{1,1}} & m_{1,1y} & m_{1,2y} & m_{1,3y} \end{bmatrix} \quad (\text{C.66})$$

$$m_{1,1x} = q_{v_{1,2}} + l_{v_{1,3}}s_{v_{1,3}} \quad (\text{C.67})$$

$$m_{1,1y} = -q_{v_{1,1}} - l_{v_{1,3}}c_{v_{1,3}} \quad (\text{C.68})$$

$$m_{1,2x} = m_{1,1x} + l_{m_{1,1}}s_{v_{1,3}}m_{1,1} \quad (\text{C.69})$$

$$m_{1,2y} = m_{1,1y} - l_{m_{1,1}}c_{v_{1,3}}m_{1,1} \quad (\text{C.70})$$

$$m_{1,3x} = m_{1,2x} + l_{m_{1,2}}s_{v_{1,3}}m_{1,12} \quad (\text{C.71})$$

$$m_{1,3y} = m_{1,2y} - l_{m_{1,2}}c_{v_{1,3}}m_{1,12} \quad (\text{C.72})$$

$$\hat{\mathbf{S}}_{u_2} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & q_{v_{2,2}} & m_{2,1_x} & m_{2,2_x} & m_{2,3_x} \\ 0 & 1 & -q_{v_{2,1}} & m_{2,1_y} & m_{2,2_y} & m_{2,3_y} \end{bmatrix} \quad (\text{C.73})$$

$$m_{2,1_x} = q_{v_{2,2}} + l_{v_{2,3}} s_{v_{2,3}} \quad (\text{C.74})$$

$$m_{2,1_y} = -q_{v_{2,1}} - l_{v_{2,3}} c_{v_{2,3}} \quad (\text{C.75})$$

$$m_{2,2_x} = m_{2,1_x} + l_{m_{2,1}} s_{v_{2,3}} m_{2,1} \quad (\text{C.76})$$

$$m_{2,2_y} = m_{2,1_y} - l_{m_{2,1}} c_{v_{2,3}} m_{2,1} \quad (\text{C.77})$$

$$m_{2,3_x} = m_{2,2_x} + l_{m_{2,2}} s_{v_{2,3}} m_{2,12} \quad (\text{C.78})$$

$$m_{2,3_y} = m_{2,2_y} - l_{m_{2,2}} c_{v_{2,3}} m_{2,12} \quad (\text{C.79})$$

$$\hat{\mathbf{S}}_t = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & q_{t_2} \\ 0 & 1 & -q_{t_1} \end{bmatrix} \quad (\text{C.80})$$

$$\hat{\mathbf{S}}_{r_1} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{1,1}} s_{t_3} + q_{r_{1,2}} c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{1,1}} c_{t_3} + q_{r_{1,2}} s_{t_3} \end{bmatrix} \quad (\text{C.81})$$

$$\hat{\mathbf{S}}_{r_2} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{2,1}} s_{t_3} + q_{r_{2,2}} c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{2,1}} c_{t_3} + q_{r_{2,2}} s_{t_3} \end{bmatrix} \quad (\text{C.82})$$

Um particionamento para o sistema operando livre de obstáculos e considerando que os veículos permanecem estacionários é descrito na Equação C.83. Com isso, resolve-se a cinemática para os manipuladores a partir da restrição de movimento dos veículos, da especificação do movimento desejado para o objeto e das posições relativas entre os manipuladores e a referência do objeto.

$$\mathbf{N}\dot{\mathbf{q}} = \begin{bmatrix} \hat{\mathbf{S}}_{v_1} & \hat{\mathbf{S}}_{m_1} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{S}}_t & -\hat{\mathbf{S}}_{r_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{S}}_{v_2} & \hat{\mathbf{S}}_{m_2} & -\hat{\mathbf{S}}_t & \mathbf{0} & -\hat{\mathbf{S}}_{r_2} \end{bmatrix} \begin{bmatrix} \dot{q}_{v_1} \\ \dot{q}_{m_1} \\ \dot{q}_{v_2} \\ \dot{q}_{m_2} \\ \dot{q}_t \\ \dot{q}_{r_1} \\ \dot{q}_{r_2} \end{bmatrix} = \mathbf{0} \quad (\text{C.83})$$

O particionamento pode ser diferente de acordo com as exigências da tarefa, como a possibilidade de algum dos manipuladores estar próximo de uma singularidade ou de limites de juntas, por exemplo, o que implica na necessidade de movimentar o veículo ao qual ele está vinculado.

C.3.4 Dois UVMS Planares Com Dois Manipuladores Cada Em Cooperação

Um segundo cenário de operação cooperativa entre UVMS envolve UVMS com dois manipuladores. O sistema, formado por dois UVMS com a configuração cinemática apresentada na Subseção C.2.2, é apresentado na Figura 94. A descrição de uma tarefa consiste na definição do movimento desejado para o objeto a ser manipulado através da cadeia virtual t e das posturas dos efetadores finais dos UVMS em relação ao objeto representadas pelas cadeias virtuais r_{11} , r_{12} , r_{21} e r_{22} . O grafo de movimento desse cenário é mostrado na Figura 95.

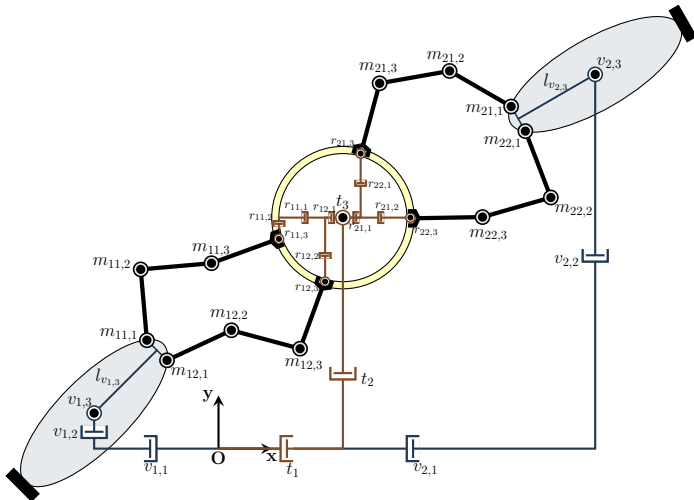


Figura 94. Representação funcional do cenário de cooperação entre dois UVMS planares com dois manipuladores cada

A matriz de helicoides normalizados dessa cadeia cinemática é obtida pela composição das cadeias dos dois UVMS (v_1 , m_{11} , m_{12} , v_2 , m_{21} e m_{22}) e das subcadeias t , r_{11} , r_{12} , r_{21} e r_{22} . O referencial adotado

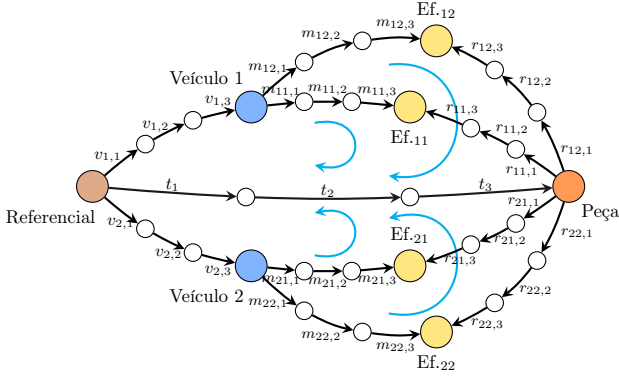


Figura 95. Grafo de movimento do cenário de cooperação entre dois UVMS planares com dois manipuladores cada

nesse cenário é o inercial, por ser comum aos dois UVMS e ao objeto manipulado. Assim, a matriz \mathbf{D} assume a forma da Equação C.84,

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{s}}_{v_1} & \hat{\mathbf{s}}_{m_{11}} & \hat{\mathbf{s}}_{m_{12}} & \hat{\mathbf{s}}_{v_2} & \hat{\mathbf{s}}_{m_{21}} & \hat{\mathbf{s}}_{m_{22}} & \cdots \\ \cdots & \hat{\mathbf{s}}_t & {}^I\mathbf{T}_p^p \hat{\mathbf{s}}_{r_{11}} & {}^I\mathbf{T}_p^p \hat{\mathbf{s}}_{r_{12}} & {}^I\mathbf{T}_p^p \hat{\mathbf{s}}_{r_{21}} & {}^I\mathbf{T}_p^p \hat{\mathbf{s}}_{r_{22}} \end{bmatrix} \quad (\text{C.84})$$

onde a transformação de helicoides ${}^I\mathbf{T}_p$ é definida na Equação C.54.

A partir do grafo de movimento determina-se a matriz de circuitos \mathbf{B} que é descrita na Equação C.85.

$$\mathbf{B} = \begin{bmatrix} \mathbf{1}_3 & \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{1}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{1}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{1}_3 & -\mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{1}_3 \end{bmatrix} \quad (\text{C.85})$$

Assim, a matriz de rede assume a forma da Equação C.86.

$$\mathbf{N} = \begin{bmatrix} \hat{\mathbf{s}}_{v_1} & \hat{\mathbf{s}}_{m_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{s}}_t & -\hat{\mathbf{s}}_{r_{11}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hat{\mathbf{s}}_{v_1} & \mathbf{0} & \hat{\mathbf{s}}_{m_{12}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{s}}_t & \mathbf{0} & -\hat{\mathbf{s}}_{r_{12}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{s}}_{v_2} & \hat{\mathbf{s}}_{m_{21}} & \mathbf{0} & -\hat{\mathbf{s}}_t & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{s}}_{r_{21}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{s}}_{v_2} & \mathbf{0} & \hat{\mathbf{s}}_{m_{22}} & -\hat{\mathbf{s}}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\hat{\mathbf{s}}_{r_{22}} \end{bmatrix} \quad (\text{C.86})$$

Os componentes da matriz de rede são expressos como

$$\hat{\$}_{v_1} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & q_{v_1,2} \\ 0 & 1 & -q_{v_1,1} \end{bmatrix} \quad (\text{C.87})$$

$$\hat{\$}_{m_{11}} = \begin{bmatrix} 1 & 1 & 1 \\ m_{11,1x} & m_{11,2x} & m_{11,3x} \\ m_{11,1y} & m_{11,2y} & m_{11,3y} \end{bmatrix} \quad (\text{C.88})$$

$$m_{11,1x} = y_{m_{11}} c_{v_1,3} + ef_{v_1,y} \quad (\text{C.89})$$

$$m_{11,1y} = y_{m_{11}} s_{v_1,3} - ef_{v_1,x} \quad (\text{C.90})$$

$$m_{11,2x} = m_{11,1x} + l_{m_{11,1}} s_{v_1,3} m_{11,1} \quad (\text{C.91})$$

$$m_{11,2y} = m_{11,1y} - l_{m_{11,1}} c_{v_1,3} m_{11,1} \quad (\text{C.92})$$

$$m_{11,3x} = m_{11,2x} + l_{m_{11,2}} s_{v_1,3} m_{11,12} \quad (\text{C.93})$$

$$m_{11,3y} = m_{11,2y} - l_{m_{11,2}} c_{v_1,3} m_{11,12} \quad (\text{C.94})$$

$$ef_{v_1,x} = q_{v_1,1} + l_{v_1,3} c_{v_1,3} \quad (\text{C.95})$$

$$ef_{v_1,y} = q_{v_1,2} + l_{v_1,3} s_{v_1,3} \quad (\text{C.96})$$

$$\hat{\$}_{m_{12}} = \begin{bmatrix} 1 & 1 & 1 \\ m_{12,1x} & m_{12,2x} & m_{12,3x} \\ m_{12,1y} & m_{12,2y} & m_{12,3y} \end{bmatrix} \quad (\text{C.97})$$

$$m_{12,1x} = y_{m_{12}} c_{v_1,3} + ef_{v_1,y} \quad (\text{C.98})$$

$$m_{12,1y} = y_{m_{12}} s_{v_1,3} - ef_{v_1,x} \quad (\text{C.99})$$

$$m_{12,2x} = m_{12,1x} + l_{m_{12,1}} s_{v_1,3} m_{12,1} \quad (\text{C.100})$$

$$m_{12,2y} = m_{12,1y} - l_{m_{12,1}} c_{v_1,3} m_{12,1} \quad (\text{C.101})$$

$$m_{12,3x} = m_{12,2x} + l_{m_{12,2}} s_{v_1,3} m_{12,12} \quad (\text{C.102})$$

$$m_{12,3y} = m_{12,2y} - l_{m_{12,2}} c_{v_1,3} m_{12,12} \quad (\text{C.103})$$

$$\hat{\$}_{v_2} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & q_{v_2,2} \\ 0 & 1 & -q_{v_2,1} \end{bmatrix} \quad (\text{C.104})$$

$$\hat{\mathfrak{S}}_{m_{21}} = \begin{bmatrix} 1 & 1 & 1 \\ m_{21,1x} & m_{21,2x} & m_{21,3x} \\ m_{21,1y} & m_{21,2y} & m_{21,3y} \end{bmatrix} \quad (\text{C.105})$$

$$m_{21,1x} = y_{m_{21}} c_{v_{2,3}} + ef_{v_{2,y}} \quad (\text{C.106})$$

$$m_{21,1y} = y_{m_{21}} s_{v_{2,3}} - ef_{v_{2,x}} \quad (\text{C.107})$$

$$m_{21,2x} = m_{21,1x} + l_{m_{21,1}} s_{v_{2,3}} m_{21,1} \quad (\text{C.108})$$

$$m_{21,2y} = m_{21,1y} - l_{m_{21,1}} c_{v_{2,3}} m_{21,1} \quad (\text{C.109})$$

$$m_{21,3x} = m_{21,2x} + l_{m_{21,2}} s_{v_{2,3}} m_{21,12} \quad (\text{C.110})$$

$$m_{21,3y} = m_{21,2y} - l_{m_{21,2}} c_{v_{2,3}} m_{21,12} \quad (\text{C.111})$$

$$ef_{v_{2,x}} = q_{v_{2,1}} + l_{v_{2,3}} c_{v_{2,3}} \quad (\text{C.112})$$

$$ef_{v_{2,y}} = q_{v_{2,2}} + l_{v_{2,3}} s_{v_{2,3}} \quad (\text{C.113})$$

$$\hat{\mathfrak{S}}_{m_{22}} = \begin{bmatrix} 1 & 1 & 1 \\ m_{22,1x} & m_{22,2x} & m_{22,3x} \\ m_{22,1y} & m_{22,2y} & m_{22,3y} \end{bmatrix} \quad (\text{C.114})$$

$$m_{22,1x} = y_{m_{22}} c_{v_{2,3}} + ef_{v_{2,y}} \quad (\text{C.115})$$

$$m_{22,1y} = y_{m_{22}} s_{v_{2,3}} - ef_{v_{2,x}} \quad (\text{C.116})$$

$$m_{22,2x} = m_{22,1x} + l_{m_{22,1}} s_{v_{2,3}} m_{22,1} \quad (\text{C.117})$$

$$m_{22,2y} = m_{22,1y} - l_{m_{22,1}} c_{v_{2,3}} m_{22,1} \quad (\text{C.118})$$

$$m_{22,3x} = m_{22,2x} + l_{m_{22,2}} s_{v_{2,3}} m_{22,12} \quad (\text{C.119})$$

$$m_{22,3y} = m_{22,2y} - l_{m_{22,2}} c_{v_{2,3}} m_{22,12} \quad (\text{C.120})$$

$$\hat{\mathfrak{S}}_t = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & q_{t_2} \\ 0 & 1 & -q_{t_1} \end{bmatrix} \quad (\text{C.121})$$

$$\hat{\mathbf{S}}_{r_{11}} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{11,1}}s_{t_3} + q_{r_{11,2}}c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{11,1}}c_{t_3} + q_{r_{11,2}}s_{t_3} \end{bmatrix} \quad (\text{C.122})$$

$$\hat{\mathbf{S}}_{r_{12}} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{12,1}}s_{t_3} + q_{r_{12,2}}c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{12,1}}c_{t_3} + q_{r_{12,2}}s_{t_3} \end{bmatrix} \quad (\text{C.123})$$

$$\hat{\mathbf{S}}_{r_{21}} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{21,1}}s_{t_3} + q_{r_{21,2}}c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{21,1}}c_{t_3} + q_{r_{21,2}}s_{t_3} \end{bmatrix} \quad (\text{C.124})$$

$$\hat{\mathbf{S}}_{r_{22}} = \begin{bmatrix} 0 & 0 & 1 \\ c_{t_3} & -s_{t_3} & q_{t_2} + q_{r_{22,1}}s_{t_3} + q_{r_{22,2}}c_{t_3} \\ s_{t_3} & c_{t_3} & -q_{t_1} - q_{r_{22,1}}c_{t_3} + q_{r_{22,2}}s_{t_3} \end{bmatrix} \quad (\text{C.125})$$

Um particionamento para resolver a cinemática do sistema para os manipuladores é apresentado na Equação C.126. Considera-se que o espaço de trabalho é livre de obstáculos e que os veículos permanecem estacionários nesse cenário.

$$\mathbf{N}\dot{\mathbf{q}} = \begin{bmatrix} \hat{\mathbf{S}}_{v_1} & \hat{\mathbf{S}}_{m_{11}} & \hat{\mathbf{S}}_{m_{12}} & \hat{\mathbf{S}}_{v_2} & \hat{\mathbf{S}}_{m_{21}} & \hat{\mathbf{S}}_{m_{22}} & -\hat{\mathbf{S}}_t & -\hat{\mathbf{S}}_{r_{11}} & -\hat{\mathbf{S}}_{r_{12}} & -\hat{\mathbf{S}}_{r_{21}} & -\hat{\mathbf{S}}_{r_{22}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_{v_1} \\ \dot{\mathbf{q}}_{m_{11}} \\ \dot{\mathbf{q}}_{m_{12}} \\ \dot{\mathbf{q}}_{v_2} \\ \dot{\mathbf{q}}_{m_{21}} \\ \dot{\mathbf{q}}_{m_{22}} \\ \dot{\mathbf{q}}_t \\ \dot{\mathbf{q}}_{r_{11}} \\ \dot{\mathbf{q}}_{r_{12}} \\ \dot{\mathbf{q}}_{r_{21}} \\ \dot{\mathbf{q}}_{r_{22}} \end{bmatrix} = \mathbf{0} \quad (\text{C.126})$$

Como no cenário anterior, podem haver diferentes particionamentos de acordo com as exigências da tarefa.

APÊNDICE D – Mapa Conceitual da Robótica Subaquática

Ver arquivo tese-mapaConceitual.pdf

Projeto hospedado em <https://sourceforge.net/projects/kastframework>