



Pós-Graduação em Ciência da Computação

“Mapa Auto-Organizável para Controle e  
Gerenciamento de Locomoção Artificial.”

Por

**Orivaldo Vieira de Santana Júnior**

Dissertação de Mestrado



Universidade Federal de Pernambuco  
[posgraduacao@cin.ufpe.br](mailto:posgraduacao@cin.ufpe.br)  
[www.cin.ufpe.br/~posgraduacao](http://www.cin.ufpe.br/~posgraduacao)

Recife, Agosto/2010



Universidade Federal de Pernambuco  
Centro de Informática  
Pós-graduação em Ciência da Computação

Orivaldo Vieira de Santana Júnior

**“Mapa Auto-Organizável para Controle e  
Gerenciamento de Locomoção Artificial.”**

*Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.*

Orientador: Aluizio Fausto Ribeiro Araújo

Recife, Agosto/2010

**Santana Júnior, Orivaldo Vieira de**  
Mapa auto-organizável para controle e gerenciamento de  
locomoção artificial / Orivaldo Vieira de Santana Júnior -  
Recife: O Autor, 2010.  
xii, 101 folhas : il., fig., tab.

Dissertação (mestrado) Universidade Federal de  
Pernambuco. CIn. Ciência da computação, 2010.

Inclui bibliografia e apêndice.

1. Ciência da computação – Redes neurais artificiais. I.  
Título.

006.3

CDD (22. ed.)

MEI2010 – 0170



SERVIÇO PÚBLICO FEDERAL  
UNIVERSIDADE FEDERAL DE PERNAMBUCO  
CENTRO DE INFORMÁTICA  
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata de Defesa de Dissertação de Mestrado do  
Centro de Informática da Universidade Federal  
de Pernambuco, 30 de agosto de 2010.

Ao trigésimo dia do mês de agosto do ano  
dois mil e dez, às oito horas, no Centro de Informática da Universidade Federal de  
Pernambuco, teve início a **nongentésima vigésima quarta** defesa de dissertação de  
mestrado em Ciência da Computação, intitulada "**Mapa Auto-Organizável para Controle e  
Gerenciamento de Locomoção Artificial**", do candidato **Orivaldo Vieira de Santana  
Júnior**, o qual já havia preenchido anteriormente as demais condições exigidas para a  
obtenção do grau de mestre. A Banca Examinadora, composta pelos professores Adriano  
Lorena Inácio de Oliveira, pertencente ao Centro de Informática desta Universidade, Adrião  
Duarte Dória Neto, pertencente Departamento de Engenharia de Computação e Automação  
da Universidade Federal do Rio Grande do Norte e Aluizio Fausto Ribeiro Araújo,  
pertencente ao Centro de Informática desta Universidade, sendo o primeiro presidente da  
Banca Examinadora e o último orientador do trabalho de dissertação decidiu, **em exigência**,  
dar o prazo de **trinta dias** para a entrega da versão final do trabalho, com as devidas  
correções propostas pelos examinadores. E para constar lavrei a presente ata que vai por  
mim assinada e pela Banca Examinadora. Recife, 30 de agosto de 2010.

Maria Lília Pinheiro de Freitas

(secretária)

Adriano D. D. Oliveira

Prof. Adriano Lorena Inácio de Oliveira  
Centro de Informática / UFPE

Adrião Duarte Dória Neto

Prof. Adrião Duarte Dória Neto  
Departamento de Engenharia de Computação e Automação / UFRN

Aluizio Fausto Ribeiro Araújo

Centro de Informática / UFPE

  
Maria do Socorro Chaves de Oliveira  
Assistente em Administração  
Pós-Graduação em  
Ciência da Computação / UFPE

Confere Com o Original  
Recife, 01/09/2010

*Eu dedico esta dissertação aos meus pais como fruto do investimento em minha educação.*

# Agradecimentos

Agradeço ao meu orientador, Aluízio Araújo, pela presença constante, pelo apoio, pelo incentivo, por ter sido de fato um guia em todos os momentos da execução deste trabalho, sem ele este trabalho não teria sido realizado.

A minha mãe, Girlêde Santana, pelo amor de mãe que se fazer presente mesmo a centenas de quilômetros de distância. Ao meu pai, Orivaldo Santana, por ter dado início ao meu processo de formação educacional. A minha irmã simplesmente por fazer parte da minha vida e do meu processo de formação como pessoa.

Aos meus tios, José Maria, Orlando, Osvaldino, Oldericó, aos meus primos Fabrício e Aise Anne, aos membros da família aqui não citados por estarem disponíveis para ajudar em qualquer momento.

Aos amigos que fiz em Pernambuco, André Tiba, Hansenclever, Flávia, Davi, Miguel e todos os outros aqui não citados, pela horas compartilhadas de trabalho e diversão.

Os professores do Centro de Informática que contribuíram para minha formação no mestrado.

A toda minha família e amigos que direta ou indiretamente contribuíram para a realização do mestrado.

Aos membros da banca pela contribuições na versão final desta dissertação.

# Resumo

Este trabalho está situado na área de controle e gerenciamento de locomoção de robôs com membros e apresenta um novo modelo de rede neural, o STRAGIC (Gerador de Trajetórias de Estados com Inter-Conexões), bem como outros modelos de redes neurais aplicadas a este domínio do conhecimento. O STRAGIC foi projetado a partir do STRAGEN, um mapa auto-organizável de topologia variável. O STRAGIC controla a locomoção do robô por meio de uma trajetória de estados que descreve a postura do robô em intervalos regulares de tempo. Alguns ambientes de teste foram elaborados para verificar a capacidade do STRAGIC em: controlar o robô com um determinado modo de locomoção; controlar o robô a partir de dados ruidosos; controlar o robô a partir de uma base de dados com estados de diferentes trajetórias; gerenciar a transição entre modos de locomoção; e por fim extrair trajetórias de estados a partir da locomoção de um animal real. Além disso, faz um estudo de dois parâmetros importantes do STRAGEN.

**Palavras-chave:** Gerador Central de Padrões, CPG, Redes Neurais, Mapas Auto-Organizáveis, Robôs com Membros, Locomoção.

# Abstract

This work is placed within the area of “Control and Management of Legged Robot’s locomotion” and presents a new model of neural networks: the STRAGIC (State’s Trajectories Generator with Inter-Connection) as well as others models and neural networks applied to this knowledge domain. The STRAGIC was designed from the STRAGEN that is a self-organized map of variable topology. The STRAGIC controls the robot’s locomotion by the state’s trajectories that describe robot’s posture in regular intervals of time. Some environments of test were made to verify the STRAGIC’s capacity to: control the robot with a determined locomotion mode; control the robot with noisy data; control the robot from a database with states of different trajectories; manage transitions between gaits; and finally extract state’s trajectories from the locomotion of a real animal. Furthermore, studies two important parameters of the STRAGIC.

**Keywords:** CPG, Central Pattern Generator, Neural Networks, Self-Organized Maps, Legged Robots, Locomotion

# Sumário

<b>Lista de Figuras</b>	<b>ix</b>
<b>Lista de Tabelas</b>	<b>xi</b>
<b>Lista de Acrônimos</b>	<b>xii</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Descrição do Problema</b>	<b>6</b>
2.1 Gerador Central de Padrões - CPG . . . . .	9
2.1.1 Modelos de CPGs . . . . .	11
2.1.2 Projeto de CPGs . . . . .	12
2.2 Trajetória de Estados . . . . .	13
2.3 Considerações . . . . .	13
<b>3 CPGs e Redes Neurais</b>	<b>15</b>
3.1 CPGs Biologicamente Inspirados . . . . .	15
3.2 Modelos Matemáticos de Neurônios de CPG . . . . .	18
3.3 Modelagem Biológica do CPG da Salamandra . . . . .	19
3.4 Rede Celular Não-linear . . . . .	21
3.4.1 A Célula CNN . . . . .	22
3.4.2 Modelagem Matemática . . . . .	23
3.4.3 Visão de Circuito Elétrico . . . . .	25
3.4.4 A CNN Difusão de Reação . . . . .	25
3.5 O Modelo de Arena . . . . .	28
3.5.1 Abordagem Multi- <i>Template</i> . . . . .	29
3.5.2 Mapas Motores . . . . .	31
3.6 Considerações . . . . .	31
<b>4 Mapas Auto-Organizáveis</b>	<b>33</b>
4.1 Algumas Definições . . . . .	33
4.2 SOM . . . . .	34
4.3 GCS . . . . .	35
4.4 GNG . . . . .	38
4.5 GWR . . . . .	40
4.6 Considerações . . . . .	43

---

<b>5 Proposição do Modelo: STRAGIC</b>	<b>44</b>
5.1 A Estrutura do STRAGIC . . . . .	46
5.1.1 Módulo de Controle de Locomoção . . . . .	46
5.1.2 Módulo de Gerenciamento de Locomoção . . . . .	51
5.2 Discussões . . . . .	53
<b>6 Experimentos</b>	<b>56</b>
6.1 Dados Artificiais . . . . .	57
6.1.1 Estudo Paramétrico do CCMT . . . . .	62
6.1.2 Controle de Locomoção . . . . .	67
6.1.3 Três Modos de Locomoção numa Base de Dados . . . . .	67
6.1.4 Dados Ruidosos . . . . .	69
6.1.5 Gerenciamento de Locomoção . . . . .	75
6.2 Dados Reais . . . . .	76
6.2.1 Discussão sobre Taxa de Êxito . . . . .	80
6.3 Considerações . . . . .	82
<b>7 Conclusão e Trabalhos Futuros</b>	<b>84</b>
7.1 Contribuições para o STRAGEN . . . . .	85
7.2 Trabalhos Futuros . . . . .	86
<b>Referências</b>	<b>88</b>
<b>Appendices</b>	<b>93</b>
<b>A Comportamento Dinâmico de um Sistema Autônomo</b>	<b>94</b>
<b>B O Simulador</b>	<b>98</b>

# Lista de Figuras

1.1	Robô manipulador industrial (©KUKA). . . . .	1
1.2	Robô para detectar minas terrestres. . . . .	2
1.3	Robô biologicamente inspirado na lagosta para localizar minas subaquáticas. . . . .	3
2.1	Robô móvel <i>Sojourner</i> . . . . .	7
2.2	Robô caminhante projetado pela © <i>Plustech</i> . . . . .	8
3.1	CPG da salamandra. . . . .	16
3.2	Uma CNN de dimensão MxN . . . . .	22
3.3	Célula autônoma da CNN . . . . .	24
3.4	Um circuito elétrico de uma célula CNN. . . . .	25
3.5	Gráfico do estado interno versus a saída de uma célula CNN . . . . .	26
3.6	Gráficos dos sinais de uma rede CNN com dois neurônios. . . . .	27
5.1	Treinamento do STRAGIC. . . . .	47
5.2	Diagrama de utilização do STRAGIC para um modo de locomoção. . . . .	53
5.3	Diagrama de utilização do STRAGIC para mais de um modo de locomoção. . . . .	54
5.4	Diagrama desvio padrão, fonte Wikipédia. . . . .	55
6.1	Os graus de liberdade dos membros do robô hexápode. . . . .	56
6.2	Captura de tela com a janela do simulador. . . . .	58
6.3	Quadros do modo de locomoção médio. . . . .	59
6.4	Sinais aplicados às articulações $\beta$ no modo de locomoção lento . . . . .	60
6.5	Sinais aplicados às articulações $\beta$ no modo de locomoção médio . . . . .	60
6.6	Sinais aplicados às articulações $\beta$ durante o modo de locomoção rápido . . . . .	61
6.7	Três redes de tamanhos diferentes construídas pelo STRAGIC . . . . .	63
6.8	Situação em que o STRAGIC falhou ao tentar construir a trajetória original. . . . .	65
6.10	Trajetória de estados . . . . .	68
6.11	Falha ao criar uma rede dos três modos de locomoção . . . . .	70
6.12	Experimento com ruído para o modo de locomoção lento . . . . .	71
6.13	Experimento com ruído para o modo de locomoção médio . . . . .	72
6.14	Experimento com ruído para o modo de locomoção rápido . . . . .	73
6.15	Ruídos mais elevados . . . . .	74
6.16	Transições imediatas entre os modos de locomoção. . . . .	77
6.17	Melhor transição entre modos de locomoção. . . . .	78

---

6.18 Locomoção de um cachorro. . . . .	79
6.19 Anatomia lateral do esqueleto de um cachorro . . . . .	81
6.20 Marcação das articulações do cachorro. . . . .	81
6.21 Situação em que o CCMT falha ao criar uma conexão. . . . .	82
6.22 Situação em que o CCMT consegue criar uma conexão que possui uma distância maior. . . . .	83
A.1 Plano de fase e oscilações no tempo, para $\mu$ assumindo valores negativos. . . . .	95
A.2 Plano de fase e oscilações no tempo para $\mu$ assumindo valores positivos. . . . .	96
A.3 Plano de fase e oscilações no tempo para $\mu = 0$ . . . . .	97
B.1 Estrutura Geral dos componentes do Gazebo. . . . .	99

# Lista de Tabelas

3.1	<i>Template A</i>	24
3.2	<i>Template A</i>	26
6.1	Limiar de Atividade e Tamanho da Rede.	64
6.2	Fator de Poda e Quantidade de Conexões.	64
6.3	Distância entre trajetórias.	67
6.4	<i>Setup experimental para os três modos de locomoção</i>	74
6.5	Taxa de êxito ao criar a rede.	80

# Lista de Acrônimos

**CCMT** Componente Construtor de Mapa Topológico.

**CNN** Rede Neural Não-linear Celular.

**CPG** Gerador Central de Padrões.

**DTW** Comparador Dinâmico no Tempo (*Dynamic Time Warping*).

**GCS** Estruturas Celulares Crescentes (*Growing Cells Structures*).

**GNG** Gás Neural Crescente (*Growing Neural Gas*)

**GWR** Cresce Quando Necessário (*Grow When Required*).

**MCL** Módulo Controle de Locomoção.

**MGL** Módulo Gerenciamento de Locomoção.

**MLR** Região Locomotora Mesencefálica (*Mesencephalic Locomotor Region*).

**NASA** Administração Nacional do Espaço e da Aeronáutica (*National Aeronautics and Space Administration*).

**SOM** Mapa Auto-Organizável (*Self-Organizing Map*).

**STRAGEN** Gerador de Trajetórias de Estados (*State Trajectories Generator*).

**STRAGIC** Gerador de Trajetórias de Estados com Interconexões (*State Trajectories Generator with Interconections*).

# 1

## Introdução

Os robôs são utilizados com bastante sucesso na produção industrial. Em uma linha de montagem de uma fábrica, um braço robótico pode mover se com grande velocidade e precisão para executar tarefas repetitivas como pintar e soldar, Figura 1.1. De acordo com [Siegwart e Nourbakhsh \(2004\)](#), apesar do sucesso de aplicação na indústria o espaço físico de ação de um braço robótico é limitado, pois este tipo de robô fica preso em uma determinada posição na fábrica.



Figura 1.1: Robô manipulador industrial (©KUKA).

A característica marcante dos robôs móveis é justamente a sua capacidade de locomoção por um ambiente. No entanto, robôs com membros possuem maior capacidade de adaptar-se a terrenos diversos. Embora o controle de robôs caminhantes seja mais complexo do que o de máquinas com rodas. Na natureza, de acordo com [Arena et al. \(2004\)](#), a locomoção com patas é a forma mais comum dos animais deslocarem-se por diversos tipos de terrenos. Por esta razão, soluções biologicamente inspiradas frequentemente são adotadas para construir robôs com membros.

Os robôs móveis podem ser aplicados em diversas tarefas tais como guia turístico, defesa nacional, exploração de recursos, exploração subaquática, desarmamento de bomba, busca de sobreviventes em escombros ([Jing, 2005](#)). [Siegwart e Nourbakhsh \(2004\)](#) apresentam um robô com rodas utilizado para a inspeção de dutos de ar e tam-

---

bém um robô guia de turismo capaz de interagir com pessoas e apresentar exposições de uma forma educativa. Um exemplo de robô de seis membros é apresentado por Santos *et al.* (2007) para a detecção de minas terrestres, Figura 1.2. Ayers e Witting (2007) construíram um robô biologicamente inspirado na lagosta para reconhecer mudanças na água do mar, localizar e desarmar minas subaquáticas, Figura 1.3.



Figura 1.2: Robô para detectar minas terrestres (imagem obtida em [http://www.iai.csic.es/users/silo6/SILO6\\_WalkingRobot.htm](http://www.iai.csic.es/users/silo6/SILO6_WalkingRobot.htm)).

Este trabalho faz um estudo de técnicas que utilizam redes neurais no controle de locomoção de robôs com membros inferiores e propõe uma abordagem baseada em um mapa auto-organizável para o controle de locomoção robôs com membros. As abordagens mais citadas na literatura para o controle de locomoção de robôs com membros são biologicamente inspiradas em um circuito neural chamado de Gerador Central de Padrões (CPG) (Ijspeert, 2008). Um CPG é composto de osciladores neurais que produz padrões (sinais) repetitivos envidados para os músculos dos membros do animal. Como os osciladores do CPG atuam sincronizadamente, os músculos são ativados de maneira sincronizada e rítmica provocando o deslocamento do animal. Normalmente um CPG é modelado matemática através de equações diferenciais, mais especificamente com equações de osciladores não-lineares (Arena *et al.*, 2002; Ijspeert *et al.*, 2007). Este tipo de modelagem implica em ajustar um conjunto de parâmetros ou até mesmo modificar equações para produzir o controle de locomoção desejado.

A abordagem proposta, chamada de STRAGIC (Gerador de Trajetórias de Estados com Interconexões), possui algumas características extraídas dos CPGs. São elas: executar a transição entre velocidades de deslocamento do robô através de um simples comando. No CPG este comando é um sinal vindo do tronco cerebral; os sinais de saída gerados são sincronizados e rítmicos, assim como nos CPGs. A abordagem proposta neste trabalho é inovadora, pois não foi encontrado nenhum registro na literatura de abordagens semelhantes para robôs caminhantes.



Figura 1.3: Robô biologicamente inspirado na lagosta para localizar minas subaquáticas.

O mapa auto-organizável escolhido foi o STRAGEN ([Benante e Araujo, 2007](#)), pois entre alguns mapas auto-organizáveis de topologia variável, o STRAGEN foi aplicado com sucesso a um problema semelhante, o controle de manipuladores robóticos. Os dados fornecidos ao modelo proposto são obtidos de um robô simulado ou um animal real, mas sem considerar informações cronológicas. Estes dados são organizados de maneira autônoma pela rede gerando uma estrutura semelhante a um grafo. Cada neurônio da rede é considerado um nó e possui um estado contendo informações relacionadas ao posicionamento dos membros.

O controle de locomoção consiste basicamente em construir uma trajetória de estados e utilizar as informações contidas nos estados para determinar o posicionamento dos membros do robô durante a locomoção. De maneira simplificada, a abordagem proposta captura informações da locomoção de um agente externo, gera uma representação interna e utiliza este representação interna para controlar o robô. O agente externo pode ser um robô ou animal real e o robô controlado pela abordagem proposta precisa ser o mais similar possível ao agente externo.

Este trabalho tem como principal objetivo propor e implementar um modelo para controle de locomoção de um robô com membros inferiores. O modelo deve tomar como base um mapa auto-organizável. Os objetivos específicos podem ser definidos a seguir:

- Investigar a aplicabilidade de modelos de redes neurais baseados em Mapas Auto-Organizáveis (SOM – *Self-Organizing Maps*), derivando adaptações ou novos modelos segundo as necessidades inerentes ao problema;
- Montar um ambiente de simulação;
- Desenvolver mecanismos de simulação e testes para a avaliação do modelo

---

proposto;

- Verificar a capacidade do modelo em gerar os mesmos sinais de saída de um CPG e gerenciar a mudança de velocidade;
- Investigar a capacidade do modelo proposto em lidar com dados extraídos da locomoção de um animal real.

Uma contribuição relevante deste trabalho é a proposição de um modelo para o controle de locomoção de robôs articulados baseado em uma abordagem inovadora. A principal característica desta nova abordagem é construir de maneira autônoma um mecanismo de controle a partir de dados extraídos da locomoção de um agente externo, seja este agente um robô ou um animal real. O modelo também é capaz de lidar com dados ruidosos. A autonomia do modelo ocorre por meio da utilização de uma mapa auto-organizável de topologia variável. Além disso, a proposta do modelo inclui um mecanismo para a transição entre diferentes modos de locomoção.

Os experimentos estão divididos em duas partes, uma para dados artificiais extraídos de uma locomoção artificial e outra com dados reais. Os dados artificiais foram gerados para um robô hexápode (com seis pernas) controlado com o algoritmo de [Arena et al. \(2004\)](#). As seguintes questões são consideradas nos experimentos com dados artificiais:

- O controle de locomoção e gerenciamento da transição entre modos de locomoção de um determinando robô hexápode;
- Um estudo paramétrico do STRAGIC (Gerador de Trajetórias de Estados com Interconexões) analisando dois dos seus parâmetros, denominados de limiar de atividade e o fator de poda. O limiar de atividade influencia na quantidade de neurônios presentes na rede produzida pelo STRAGIC e o fator de poda influencia na quantidade de conexões;
- A capacidade do modelo proposto em gerar uma trajetória de estados para controlar um determinado modo de locomoção;
- Testar a capacidade do modelo em diferenciar alguns modos de locomoção. Para tanto, dados relativos aos três modos de locomoção foram armazenados em uma só base de dados de forma aleatória, sem informação das sequências dos estados em cada trajetória;
- Comportamento do modelo diante de dados ruidosos;

- 
- O gerenciamento da transição entre modos de locomoção. A situação proposta para avaliar esta questão foi colocar em uma única base de dados de estados dos três modos de locomoção (lento, médio e rápido). Assim, o modelo teria que automaticamente identificar a qual trajetória um dado estado pertence, montar esta trajetória e as transições entre diferentes trajetórias.
  - A capacidade do modelo em lidar com dados obtidos a partir da locomoção de um animal real. Os dados sobre a locomoção foram extraídos de um vídeo de um cachorro de médio porte, obtido no *youtube*<sup>1</sup>.

Esta dissertação está organizada da seguinte maneira, o Capítulo 2 caracteriza a locomoção em animais e robôs com membros, faz um contextualização entre CPGs e controle de locomoção além de relacionar o controle de locomoção com trajetórias de estados. O Capítulo 3, em essência, faz um estudo da abordagem biologicamente inspirada e da abordagem baseada na CNN (Rede Neural Não-linear Celular) para o controle de locomoção. Antes de propor a solução do problema no Capítulo 5 o Capítulo 4 descreve os mapas auto-organizáveis mais relevantes para a proposição do STRAGEN, base para o STRAGIC. Os experimentos estão descritos no Capítulo 6. A conclusão e os trabalhos futuros estão presentes no Capítulo 7.

---

<sup>1</sup><http://www.youtube.com/watch?v=-StFMBw3W-U>

# 2

## Descrição do Problema

A robótica, um campo relativamente recente da tecnologia moderna, ultrapassa as fronteiras da engenharia tradicional. Entender a complexidade dos robôs e suas aplicações requer conhecimento em engenharia elétrica, engenharia mecânica, ciência da computação, matemática etc. O termo robô é aplicado a uma grande variedade de dispositivos mecânicos possuidores de algum grau de autonomia, podendo ser teleoperados ([Spong et al., 2006](#)).

Os robôs geralmente são projetados para realizar algum tipo de trabalho, por exemplo, manipulação industrial. O uso da robótica oferece diversas vantagens, como a diminuição do custo do trabalho, aumento da precisão e produtividade. Os robôs geralmente são utilizados em trabalhos nos quais o ser humano é submetido a condições monótonas, repetitivas ou perigosas. A robótica não é aplicada somente na indústria, mas em áreas onde o uso de humanos é impraticável ou indesejado, como: a exploração do fundo do mar ou de outro planeta; reparo ou resgate de satélite; desarmamento de dispositivos explosivos e trabalho em ambiente radioativo ([Spong et al., 2006](#)). Eles são aplicados também em áreas onde a comercialização de robôs é viável, como: polimento no chão; vigilância de uma fábrica; corte de grama; passeios em um museu; orientações em um supermercado ([Siegwart e Nourbakhsh, 2004](#)).

Em ambientes hostis, perigosos ou inhabitáveis, a aplicação de sistemas teleoperados torna-se cada vez mais comum. Para explorar a superfície de Marte, a NASA (Administração Nacional do Espaço e da Aeronáutica – *National Aeronautics and Space Administration*) utilizou um robô em modo teleoperado (Controlado a partir da Terra), Figura 2.1. A Plustech desenvolveu um robô caminhante para carregar madeira para fora da floresta, Figura 2.2, onde a navegação é feita por um operador dentro do robô e a coordenação entre pernas é automática ([Siegwart e Nourbakhsh, 2004](#)). Um exemplo de robô semi-autônomo capaz de navegar de maneira independente ou teleoperada no



Figura 2.1: Robô móvel *Sojourner* usado pela ©NASA durante a missão de exploração de Marte em 1997.

ambiente é o robô de seis membros proposto por [Santos et al. \(2007\)](#) para a detecção de minas terrestres.

Em robôs teleoperados, a complexidade por trás do mecanismo de controle de locomoção geralmente torna impossível para o operador humano controlar o deslocamento do robô. O homem executa as atividades cognitivas e de localização, mas depende do esquema de controle interno do robô para conduzir sua locomoção ([Siegwart e Nourbakhsh, 2004](#)).

Segundo [Sproewitz et al. \(2008\)](#), o controle de locomoção, bem como a reprodução de um determinado modo de locomoção, em um robô cujos membros inferiores possuam múltiplos graus de liberdade é um problema complexo e desafiador . Algumas das abordagens mais comuns ([Nakamura et al., 2007; Ijspeert et al., 2007; Ayers e Witting, 2007; Arena et al., 2004; Righetti e Ijspeert, 2006; Ijspeert, 2008](#)) para resolver o problema de locomoção de robôs com membros inferiores estão relacionadas ao CPG (*Central Pattern Generator*). O CPG controla o movimento periódico executado por cada membro, bem como o sincronismo entre membros. Sob o ponto de vista da biologia, um CPG é um circuito neural capaz de produzir sinais neurais rítmicos sem receber estímulos rítmicos. Este circuito neural, constituído de osciladores neurais, encontrado principalmente na medula espinhal de animais vertebrados durante a locomoção produz descargas periódicas de impulsos nervosos. Estes impulsos ativam os motoneurônios produzindo sequências alternadas entre flexão e extensão em vários músculos de um membro.

As características dos sinais gerados pelo CPG influenciam o movimento de cada membro. Considerando que o CPG é composto de osciladores e que o movimento de uma articulação é controlado por um conjunto de osciladores, as oscilações geradas influenciam diretamente o movimento da articulação. Logo, características como, frequência, amplitude e formas dos sinais gerados modulam o movimento de cada



Figura 2.2: Robô caminhante projetado pela ©Plustech.

articulação, influenciam na eficiência do controle motor, consequentemente, no modo de locomoção resultante ([Ijspeert, 2001](#)).

Em animais, um determinado modo de locomoção é caracterizado pelo ciclo de trabalho e pela fase relativa de cada membro. Um passo, isto é, um ciclo de locomoção do membro, é dividido em duas fases, uma chamada de apoio e outra de balanço. [Righetti e Ijspeert \(2006\)](#) caracterizam a fase de apoio pelo contato do membro com o chão e a fase de balanço pelo membro livre no ar e sem contato com o chão. Em cada instante de tempo, um determinado membro está em um posicionamento dentro da sequência de posicionamentos que compõe o passo. Cada modo de locomoção tem sua própria maneira de posicionar os membros em cada instante. Logo, um modo de locomoção de um ser articulado pode ser descrito como a movimentação coordenada dos membros, isto é, o momento e a localização de apoiar e levantar cada pé coordenados com o movimento do corpo a fim de conduzir o corpo de um lugar a outro do espaço. Um modo de locomoção determina a velocidade e a direção da movimentação de um animal ou robô caminhante ([Song e Waldron, 1989; Bekey, 2005](#)).

[McMahon \(1984\)](#) descrevem alguns modos de locomoção de quadrúpedes da seguinte forma: (i) no modo de locomoção caminhada, cada membro atinge o chão

um após o outro, e o intervalo entre cada descida é de 1/4 do ciclo da duração de um passo; (ii) no modo trote, os membros nos cantos diagonais do corpo trabalham sincronizadamente; (iii) no galope leve, um pé frontal e um pé traseiro diagonal tocam o chão juntos; (iv) um galope é um modo de locomoção rápido no qual a sequência de passadas acontece em torno de um ciclo .

A locomoção de robôs com pernas é caracterizada por uma sequência de pontos de contato entre os membros do robô e chão. Durante a locomoção, uma parte dos membros está em contato com o chão e a outra parte esta livre no ar. A principal vantagem de um robô com membros é a adaptabilidade e a capacidade de manobra em terrenos irregulares. Pois apenas um conjunto de pontos de contato é necessário para manter o robô equilibrado e deslocando-se, não importando as características do solo. O robô apenas precisa manter os membros livres e distantes do solo de maneira que não atrapalhe seu deslocamento. Além disso, um robô caminhante é capaz de atravessar um buraco ou uma fenda enquanto seu corpo passa sobre o buraco ([Siegwart e Nourbakhsh, 2004](#)).

Outra vantagem da locomoção com pernas é o potencial de manipular habilidosa-mente objetos do ambiente. Um exemplo é a locomoção do escaravelho, inseto capaz de rolar uma bola com as patas traseiras enquanto locomove-se habilidosamente com as patas frontais ([Siegwart e Nourbakhsh, 2004](#)).

A principal desvantagem da locomoção com pernas inclui a complexidade mecânica e energética. A perna, que pode possuir vários graus de liberdade, deve ser capaz de sustentar uma parte do peso total do robô. Além disso, alta capacidade de manobra será viável apenas se as pernas possuírem um número suficiente de graus de liberdade.

## 2.1 Gerador Central de Padrões - CPG

Geradores centrais de padrões (*Central Pattern Generators – CPGs*) são circuitos neurais encontrados tanto em animais vertebrados quanto em invertebrados que podem produzir padrões rítmicos de atividades neurais de forma autônoma. Estes padrões rítmicos surgem através de interações entre as unidades de processamento deste circuito neural e produzem os padrões motores responsáveis pelo deslocamento animal. Assim, os padrões rítmicos tornam movimentos periódicos viáveis, tais como andar em velocidade constante, mastigar, respirar e digerir. Neste circuito neural, a realimentação sensorial (vinda do sistema nervoso periférico) não é necessária para gerar oscilações rítmicas. Nos animais vertebrados a medula espinhal é quem faz o papel dos CPGs. Embora a realimentação sensorial não seja necessária para gerar os sinais rítmicos, ela

exerce um papel muito importante na formação destes sinais rítmicos e na manutenção da coordenação entre CPGs e os movimentos do corpo. A existência dos CPGs já foi demonstrada em muitos vertebrados tais como peixes, anfíbios, gatos e humanos (Ijspeert, 2008; Holmes *et al.*, 2006; Billard e Ijspeert, 2000; Bekey, 2005).

Um sinal elétrico simples é o bastante para induzir uma mudança de comportamento no CPG. Em muitos animais vertebrados, estímulos elétricos vindos de uma região específica no tronco cerebral chamada MLR (Região Locomotora Mesencefálica – *Mesencephalic Locomotor Region*) provocam alterações no comportamento locomotor. A MLR, uma importante região locomotora, possui vias de interligação com a Medula Espinal. A variação do estímulo aplicado ao CPG permite a modulação da velocidade e da direção de locomoção. Níveis mais baixos de estímulos na MLR levam a movimentos lentos, e níveis mais altos de estímulos levam a movimentos rápidos. Como a MLR é dividida em duas partes, direita e esquerda, ao aplicar níveis diferentes de estímulos na MLR direita e esquerda a direção de locomoção é alterada (Sproewitz *et al.*, 2008; Ijspeert, 2008).

Outra questão tratada nos estudos sobre CPGs é a integração da realimentação sensorial com controle de locomoção. Os sinais sensoriais proprioceptivos<sup>1</sup> identificam a postura e são muito importantes na locomoção de animais com patas. Fatores como velocidade, carga e posição da perna fornecem impulsos sensoriais que afetam os CPGs. Por exemplo, uma perna pode deslizar em uma superfície molhada, representando uma situação instável para o sistema que dificilmente pode ser controlada por um comando padrão. Outra situação comportamental crítica para animais caminhantes é a falta repentina de apoio para os pés quando andando. Outro exemplo é que ao ultrapassar obstáculos de vários tipos e localizações pode exigir um ajuste em tempo real do padrão motor. Assim, a realimentação sensorial modula a atividade do CPG induzindo o robô a realizar uma locomoção mais estável em um terreno complexo (Ijspeert, 2008; Durr *et al.*, 2003; Holmes *et al.*, 2006; Bekey, 2005).

Existem diferentes maneiras de aplicar algoritmos de aprendizagem e otimização aos CPGs. As abordagens basicamente são divididas em duas categorias: aprendizagem supervisionada e não-supervisionada. Técnicas de aprendizagem supervisionada são aplicadas quando um desejado padrão rítmico produzido pelo CPG é conhecido. Técnicas de aprendizagem não-supervisionada são usadas quando o comportamento desejado do CPG não é definido por um padrão específico, mas por um critério de desempenho de alto-nível, por exemplo, mover o mais rápido possível. Entre as

---

<sup>1</sup>Propriocepção é a capacidade de reconhecer a posição e o movimento dos membros do corpo. Informações estas, obtidas através da atividade de receptores localizados nos músculos.

---

técnicas de aprendizagem não-supervisionada, os algoritmos evolucionários são extensivamente aplicados para projetar modelos como CPG, apesar de possuírem a desvantagem de serem lentos e necessitarem de um uso prolongado de simulação ([Ijspeert, 2008](#)).

### 2.1.1 Modelos de CPGs

Modelos de CPGs foram construídos principalmente para insetos e vertebrados inferiores. Muitos modelos de CPG são inspirados no circuito de natação da lampreia, construídos a partir de redes não-lineares celulares ou sistemas de osciladores acoplados. Um CPG muito investigado é o da lampreia sendo modelado de várias maneiras: biofísica, conexionista, sistemas de osciladores acoplados e simulação neuromecânica ([Ijspeert, 2008](#)).

O interessante trabalho de [Ijspeert et al. \(2007\)](#) na linha de robôs biologicamente inspirados testa um modelo de CPG de uma salamandra<sup>2</sup> real em um robô anfíbio inspirado na salamandra. Este modelo, baseado no trabalho de [Örjan Ekeberg \(1993\)](#), é composto de 20 osciladores de fase de amplitude-controlada. Os osciladores recebem um sinal representando o estímulo descendente da Região Locomotora Mesencefálica. As saídas do CPG são valores para o posicionamento angular das articulações enviados para um controlador proporcional-derivativo para controlar os membros do robô.

O modelo de CPG desenvolvido por [Arena et al. \(2004\)](#) utiliza CNNs (Redes Não-lineares Celulares – *Cellular Nonlinear Networks*). Neste caso, as CNNs são transformadas em um conjunto de osciladores acoplados para compor o CPG. Este modelo também propõe um novo método para selecionar um modo de locomoção, onde a transição entre um modo de locomoção e outro é suave. Isto é viabilizado através de um controle de alto-nível baseado em um mapa motor, baseado em um mapa auto-organizado. Este mapa, inspirado no córtex motor, recebe como entrada a velocidade do robô e produz como saída os parâmetros dos osciladores CNNs, controlando assim o modo de locomoção resultante.

[Reeve e Hallam \(2005\)](#) investigaram alguns modelos de neurônios para CPGs, como por exemplo, o modelo sigmoidal. A tratabilidade analítica do modelo sigmoidal facilita o treinamento de redes *feedforward* utilizando algoritmos de gradiente descendente como *backpropagation*, aplicados ao controle de robôs insetos caminhantes. Para [Reeve e Hallam \(2005\)](#) ficou claro que modelos mais completos são mais fáceis de treinar para uma tarefa de controle, produzindo passos repetidos e aparentemente mais estáveis.

---

<sup>2</sup>As salamandras são anfíbios possuidores de cauda, assemelham-se aos lagartos e nadam como serpentes.

Isto pode estar relacionado com a facilidade de neurônios mais complexos produzirem movimentos oscilatórios e também a simplicidade de evoluírem os poucos pesos das redes menores.

[Nakamura et al. \(2007\)](#) apresentaram um método de aprendizagem por reforço aplicado ao problema de controle automático em um simulador de robô bípede. O robô é controlado por seis torques, cada um aplicado a uma junta. O CPG básico é uma rede composta por seis osciladores neurais. Cada oscilador neural gera torques de controle para uma junta correspondente. A recompensa varia de acordo com a altura do pé e da cintura, com a velocidade horizontal da cintura, e com as penalidades aplicadas quando o robô cai.

### 2.1.2 Projeto de CPGs

Ainda não existe uma metodologia bem estabelecida para projetar CPGs ([Ijspeert, 2008](#); [Wu et al., 2009](#)) e diferentes abordagens têm sido investigadas tais como osciladores não-lineares, modelagem biológica de neurônios, redes celulares não-lineares, etc. Para que o CPG construído com algumas destas abordagens comporte-se da maneira desejada é necessário que os parâmetros destas abordagens sejam configurados corretamente. Para chegar a estes parâmetros, algumas técnicas são utilizadas, como os algoritmos evolucionários ([Reeve e Hallam, 2005](#); [Ijspeert e Kodjabachian, 1999](#); [Ijspeert et al., 1999](#)), aprendizagem por reforço ([Nakamura et al., 2007](#)) ou uma metodologia específica para ajustar esses parâmetros ([Arena et al., 2002](#)).

[Ijspeert \(2008\)](#) define alguns itens presentes no projeto de CPGs:

1. A arquitetura geral do CPG, incluindo o tipo e número de neurônios;
2. As conexões entre neurônios, que determinam a sincronização entre os osciladores e influencia no modo de locomoção resultante;
3. As oscilações determinando as trajetórias executadas por cada junta durante um ciclo;
4. O efeito dos sinais de entrada, isto é, como o controle de parâmetros modula a frequência, a amplitude, a fase de movimento das pernas ou as oscilações;

Para construir CPGs baseados em osciladores não-lineares ou até mesmo em modelos matemáticos de neurônios reais é necessário conhecer diversos modelos para encontrar um que se comporte de maneira desejada. Caso tal modelo não seja encontrado, o modelo possuidor do comportamento mais próximo do desejado é escolhido

---

e adaptado. Para conseguir esta adaptação, parâmetros das equações que descrevem o modelo precisam ser modificados, adicionados ou retirados (Wu *et al.*, 2009). O Capítulo 3 descreve em detalhes duas abordagens utilizadas para projetar CPGs.

## 2.2 Trajetória de Estados

Um estado no contexto deste trabalho armazena informações que definem a postura do robô. Uma postura é determinada pela posição angular de cada articulação ou espaço euclidiano tridimensional de cada articulação. Assim, os estados de um sistema contêm a configuração, situação ou descrição daquele sistema em um dado momento (Benante, 2008).

Um espaço de estados de um sistema, de acordo com Benante (2008), é o conjunto  $E$  de estados que caracterizam este sistema, onde cada ponto  $\mathbf{e} \in E$  é uma configuração única e não-redundante deste espaço de estados. Uma descrição sobre modelo de estado pode ser encontrada em Bonet e Geffner (2001b,a).

Uma trajetória é o conjunto de transições entre estados de sistema partindo de um estado possuidor de uma configuração inicial ( $\mathbf{e}_{ini}$ ) até outro estado com uma configuração pré-definida ou de destino ( $\mathbf{e}_{dest}$ ). Durante a execução da trajetória, diferentes estados intermediários são assumidos dentro do espaço de estados possíveis do sistema. Assim, uma trajetória  $T$  é definida como uma sequência finita de estados:

$$T = \{\mathbf{e}_{ini}, \dots, \mathbf{e}_i, \dots, \mathbf{e}_{dest}\}, \quad (2.1)$$

onde  $\mathbf{e}_{ini} \in \mathbb{R}^D$  é um estado do sistema configurado como sendo um vetor de dimensão  $D$ .

De acordo com a Equação 2.1, a locomoção do robô pode ser descrita por uma trajetória de estados, onde cada estado contém informações sobre a postura do robô durante um passo. O intervalo de tempo de coleta dos estados não precisa ser maior do que o tempo de um passo completo, já que o movimento executado pelo robô nos passos seguintes são repetidos. Assim,  $\mathbf{e}_{ini}$  é a postura inicial do passo,  $\mathbf{e}_i$  uma postura intermediária e  $\mathbf{e}_{dest}$  a postura final do passo, que neste caso é a mesma postura inicial.

## 2.3 Considerações

O controle de locomoção através de CPGs requer o conhecimento de alguma metodologia para construir um modelo capaz de gerar padrões que se repetem constantemente

---

### **2.3. CONSIDERAÇÕES**

---

durante o tempo, i.e., oscilações. Os artigos, investigados e publicados antes de 2010, indicam a não existência de uma metodologia bem definida para projetar CPGs, ver Seção 2.1.2. A maneira mais comum de construir estes CPGs é através de equações diferenciais não sendo um processo trivial, ver Seção 2.1.2.

Por outro lado o STRAGIC evita o esforço de modelagem de equações matemáticas. Pois no STRAGIC as informações sobre as oscilações são passadas como entrada para o seu treinamento ao invés de criar equações matemáticas para reproduzirem as oscilações. Assim, o STRAGIC depois de treinado possui o mesmo comportamento de um CPG, mesmo não sendo um CPG.

# 3

## CPGs e Redes Neurais

Os assuntos tratados nesta seção estão relacionados a duas abordagens utilizadas para o controle de locomoção de robôs com pernas, uma baseada em um modelo de CPG biológico e a outra baseada em uma rede não-linear celular. Estas abordagens de maneira geral são biologicamente inspiradas, cada uma com um nível de abstração. A Seção 3.1 faz uma breve introdução de alguns estudos sobre CPGs biologicamente inspirados. Já a Seção 3.2 mostra como alguns CPGs são modelados matematicamente. A Seção 3.3 apresenta a abordagem com maior grau de inspiração biológica. Esta abordagem abstrai o circuito neural de controle de locomoção biológico em nível de neurônio, modelando o neurônio levando em consideração a organização dos neurônios dentro do CPG. A segunda abordagem, descrita na Seção 3.5 abstrai o CPG, em nível de comportamento, projetando uma rede não-linear celular para comportar-se de maneira semelhante a um circuito neural de controle de locomoção. Para facilitar o entendimento da segunda abordagem, a rede não-linear celular é descrita na Seção 3.4.

### 3.1 CPGs Biologicamente Inspirados

Esta seção apresenta alguns modelos de CPGs modelados utilizando uma abordagem biologicamente inspirados, principalmente os trabalhos relacionados a Ijspeert. Os primeiros estudos de Ijspeert reportam a aplicação de algoritmos evolucionários para ajustar os parâmetros de um modelo neural de controlar de locomoção da lampreia em um ambiente simulação. Este modelo neural foi desenvolvido a partir do modelo biológico da lampreia proposto por Ekeberg. Ijspeert *et al.* (1999) também apresentaram uma abordagem que utilizava um algoritmo genético para evoluir a arquitetura de um modelo conexionista capaz de determinar a atividade muscular durante a execução do nado da lampreia simulada. Ijspeert e Kodjabachian (1999) propuseram outra

### 3.1. CPGS BIOLOGICAMENTE INSPIRADOS

---

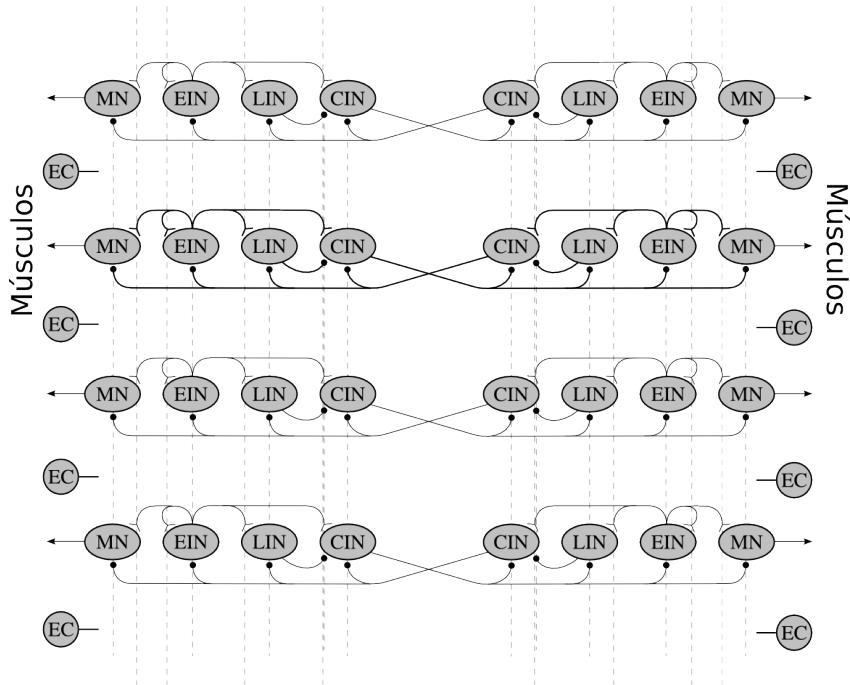


Figura 3.1: 4 segmentos de rede do controlador biológico, onde cada segmento de rede é composto de 8 neurônios. Quatro tipos de neurônios estão presentes nos osciladores: três tipos de interneurônio (EIN, CIN e LIN) e os motoneurônios MN. Os controladores podem receber realimentação das células excitatórias (EC). As linhas tracejadas mostram as interconexões entre segmentos vizinhos. Figura extraída de [Ijspeert et al. \(1999\)](#).

abordagem baseada em programação genética para evoluir programas que codificavam o crescimento de uma rede neural dinâmica.

[Örjan Ekeberg \(1993\)](#), inspirado na rede neural biológica responsável pelo movimento do corpo da lampreia, desenvolveu um modelo de controle neural com neurônios individualmente simplificados, porém com conectividade semelhante ao modelo biológico. Neste modelo, cada unidade representa uma população de neurônios reais que possuem funcionalidades semelhantes. Além disso, Ekeberg também descreveu como os sinais gerados pelo modelo de controle neural são transformados em movimentos.

O controlador neural proposto por [Örjan Ekeberg \(1993\)](#) biologicamente inspirado no CPG da lampreia é composto de 100 segmentos de rede interconectadas, ver Figura 3.1. Cada segmento de rede é um oscilador neural feito por dois motoneurônios (MN), dois interneurônios excitatórios (EIN), dois interneurônios inibitórios contralaterais (CIN) e dois interneurônios inibitórios laterais (LIN). A nomenclatura de cada neurônio descreve suas conexões eferentes. Cada neurônio, individualmente, representa uma

### 3.1. CPGS BIOLOGICAMENTE INSPIRADOS

---

população de neurônios funcionalmente similares na lampreia real, que recebem sinais excitatórios do tronco cerebral. Uma interconexão é uma conexão entre dois neurônios pertencentes a dois segmentos vizinhos na rede.

Um neurônio é modelado como um *leaky-integrator*, ver equações 3.8. Sua saída  $u$  equivale a frequência de disparo ( $\in [0, 1]$ ) calculada como a seguir:

$$\dot{\xi}_+ = \frac{1}{\tau_D} \left( \sum_{i \in \Psi_+} u_i w_i - \xi_+ \right) \quad (3.1)$$

$$\dot{\xi}_- = \frac{1}{\tau_D} \left( \sum_{i \in \Psi_-} u_i w_i - \xi_- \right) \quad (3.2)$$

$$\dot{\vartheta} = \frac{1}{\tau_A} (u - \vartheta) \quad (3.3)$$

$$u = \begin{cases} 1 - \exp\{(\Theta - \xi_+) \Gamma\} - \xi_- - \mu \vartheta & (u > 0) \\ 0 & (u \leq 0) \end{cases} \quad (3.4)$$

onde  $w_i$  é o peso sináptico,  $\Psi_+$  e  $\Psi_-$  representam os grupos de neurônios excitatórios e inibitórios pré-sinápticos respectivamente,  $\xi_+$  e  $\xi_-$  são as 'reações' atrasadas para entradas excitatórias e inibitórias, e  $\vartheta$  representa a adaptação da frequência observada em alguns neurônios reais (Ijspeert *et al.*, 1999).  $\tau_D$  é um limiar para a ativação,  $\Gamma$  é uma constante de ganho, e  $\mu$  controla o nível de adaptação (Örjan Ekeberg, 1993).

Prosseguindo os estudos sobre CPG, Ijspeert (2001) desenvolveu um modelo de CPG biologicamente plausível da salamandra. O circuito neural controlador da locomoção da salamandra é semelhante ao CPG da lampreia, mas incrementado com CPGs que controlam os membros. Este modelo era composto de 14 articulações, 10 distribuídas pelo tronco e cauda, mais 1 para cada membro contabilizando 4 articulações nos membros. Os parâmetros deste circuito neural eram determinados por algoritmo genético. O modelo resultante simulava tanto o corpo quanto o circuito locomotor da salamandra sendo capaz de fazer a transição do modo de locomoção aquático para o terrestre.

Ijspeert não se restringiu ao estudo da lampreia e da salamandra, desenvolvendo seus trabalhos com outros tipos de robôs. Righetti e Ijspeert (2006) introduziram uma metodologia para projetar controladores de robôs humanoides rastejantes, baseada no paradigma CPG. Assim como nos outros trabalhos, esta metodologia segue uma abordagem biologicamente inspirada e apresenta um modelo matemático de CPG baseado em osciladores não-lineares acoplados.

Alguns trabalhos elaboram CPGs levando em conta as interações dos neurônios

com os músculos das pernas como é o caso de [Maufroy et al. \(2008, 2010\)](#) que propõe um modelo biologicamente inspirado na neurofisiologia da locomoção de gatos. Capaz de controlar habilidosamente um robô quadrúpede em terrenos irregulares, fazer a transição entre modos de locomoção de maneira autônoma e compatível com a velocidade de deslocamento do robô.

### 3.2 Modelos Matemáticos de Neurônios de CPG

Alguns modelos matemáticos baseados no comportamento dos neurônios reais são utilizados para compor o CPG. Dentre eles está o famoso modelo H-H de [Hodgkin e Huxley \(1952\)](#), um modelo complexo e com muitos parâmetros ([Wu et al., 2009](#)). Uma simplificação do modelo H-H é o modelo FitzHugh-Nagumo ([FitzHugh, 1961; Nagumo et al., 1962](#)) definido por:

$$\begin{aligned}\dot{x}_i &= c \left( y_i + x_i + \frac{x_i^3}{3} + f_{ci} \right), \\ \dot{y}_i &= -(x_i - a + by_i)/c,\end{aligned}\tag{3.5}$$

onde  $x_i$ , é o potencial da membrana do  $i$ -ésimo neurônio;  $f_{ci}$  é um sinal de controle no neurônio  $i$ ;  $a, b$  e  $c$  são constantes e não correspondem a nenhum parâmetro fisiológico. Notação  $\dot{x}_i$  é utilizada para descrever a primeira derivada de  $x$  em relação ao tempo.

Um modelo baseado no neurônio real e voltado para a produção de sinal oscilatório na saída é o modelo de [Stein et al. \(1973\)](#), descrito matematicamente por:

$$\begin{aligned}\dot{x}_i &= a \left( -x_i + \frac{1}{1 + \exp(-f_{ci} - by_i + bz_i)} \right), \\ \dot{y}_i &= x_i - py_i,\end{aligned}\tag{3.6}$$

$$\dot{z}_i = x_i - qz_i,\tag{3.7}$$

onde  $x_i$  representa o potencial da membrana do  $i$ -ésimo oscilador;  $a$  é uma constante que afeta a frequência de oscilação;  $f_{ci}$  é um sinal de controle para o oscilador  $i$ ;  $b$  permite ao modelo adaptar-se as mudanças de estímulos;  $q$  e  $p$  controlam a taxa de adaptação.

O modelo de [Matsuoka \(1987\)](#) do tipo *Leaky-Integrator* definido matematicamente

pela Equações 3.8, descreve o comportamento básico dos neurônios reais:

$$T_r \dot{u} + u_i = - \sum_{j=1}^n w_{ij} y_j - \beta v_i + s_i, \quad T_a \dot{v}_i + v_i = y_i, \quad (3.8)$$

$$y_i = g(u_i) = \max(u_i, 0), \quad (3.9)$$

onde  $u_i$  é o potencial da membrana do  $i$ -ésimo neurônio;  $v_i$  é uma variável que representa o grau de adaptação do neurônio  $i$ ;  $T_r$  e  $T_a$  são constantes do tempo de crescimento e do tempo de adaptação;  $w_{ij}$  é o peso da sinapse inibitória da conexão que sai do neurônio  $j$  para o  $i$ ;  $\beta$  é o parâmetro que determina a taxa de disparos;  $s_i$  é uma entrada externa, e  $y_i$  é a saída do neurônio.

O neurônio de um CPG pode ser entendido como um oscilador não-linear, pois o papel do neurônio no CPG é justamente produzir periodicamente sinais oscilatórios. Entre os modelos de osciladores não-lineares estão o modelo de Kuramoto e o de Hopf (Wu *et al.*, 2009). O modelo de Kuramoto (Acebrón *et al.*, 2005) é um oscilador simples que consiste em uma população de  $N$  osciladores de fase acoplados, descrito matematicamente por:

$$\dot{\theta}_i = w_i + \sum_{j=1}^N K_{ij} \sin(\theta_j - \theta_i), \quad i = 1, 2, \dots, N, \quad (3.10)$$

onde  $\theta_i$  é a fase do  $i$ -ésimo oscilador;  $w_i$  é a frequência natural do  $i$ -ésimo oscilador;  $K_{ij} > 0$  é a força do acoplamento do oscilador  $j$  para o oscilador  $i$ .

O oscilador de Hopf pode ser descrito por:

$$\begin{aligned} \dot{x} &= (\mu - r^2)x + wy, \\ \dot{y} &= (\mu - r^2)y + wx, \end{aligned} \quad (3.11)$$

onde  $r = \sqrt{x^2 + y^2}$ ;  $\mu > 0$  determina a amplitude do sinal de saída; o parâmetro  $w$  controla a frequência do oscilador. O oscilador tem um ciclo limite estável com raio  $\sqrt{\mu}$  e velocidade angular  $w$  rad/s.

### 3.3 Modelagem Biológica do CPG da Salamandra

Ijspeert *et al.* (2007) descrevem um modelo de CPG para controlar a locomoção de um robô salamandra. Este modelo é inspirado no CPG biológico da salamandra que por

### 3.3. MODELAGEM BIOLÓGICA DO CPG DA SALAMANDRA

---

sua vez é baseado no modelo biológico do CPG da lampreia. Além de controlar o modo de locomoção da salamandra, este modelo é capaz de fazer a transição entre os modos de locomoção nadar e andar. Para modelar matematicamente este CPG, algumas questões relacionadas ao modo de locomoção de vertebrados foram levantadas. Estas questões tratam: da estrutura do circuito neural capaz de controlar tanto o modo de locomoção nadar quanto andar; da coordenação dos membros; e da mudança entre modos de locomoção apenas alterando à intensidade do estímulo elétrico aplicado ao tronco cerebral.

No modelo de CPG da salamandra de [Ijspeert et al. \(2007\)](#) a rede é distribuída ao longo da medula espinhal formando uma cadeia dupla de centros oscilatórios localizados em ambos os lados da medula espinhal. Este conjunto de centros oscilatórios são responsáveis por controlar o nado da salamandra. Os centros oscilatórios encarregados de movimentar os membros frontais estão localizados no segmento cervical e os membros traseiros estão localizados no segmento torácico-lombar.

[Ijspeert et al. \(2007\)](#) explicam o funcionamento da rede CPG da lampreia da seguinte maneira:

1. O CPG do corpo inteiro é dividido em dois CPGs, um controla os movimentos ao longo do corpo e o outro controla o movimento dos membros. O CPG do corpo é semelhante ao da lampreia e espontaneamente produz oscilações movimentando o corpo de maneira ondulatória. Quando a rede CPG dos membros entra em atividade, o CPG do corpo inteiro gera os padrões de locomoção para o modo andar.
2. A rede CPG dos membros é capaz de induzir a rede CPG do corpo a mudar do modo nadar para o modo andar. Já que os acoplamentos entre osciladores dos membros e do corpo são mais fortes que os acoplamentos entre os osciladores corpo.
3. Ao aumentar a intensidade do estímulo elétrico aplicado ao tronco cerebral acontece a mudança entre os modos de locomoção andar e nadar. Os níveis mais baixos de estímulos, menores que um determinado limiar, levam a um modo andar lentamente enquanto estímulos mais intensos induzem a uma mudança para o modo nadar rapidamente. Em ambos os modos, a frequência de movimentação é proporcional a intensidade do estímulo. Para que ocorra esta transição os osciladores dos membros não oscilam em altas frequências. Eles saturam e param de oscilar para altos níveis de estímulos. Logo, as frequências do modo nadar são sistematicamente mais elevadas que as do modo caminhar. Durante

o aumento da intensidade do estímulo, os osciladores passam por três fases: (i) uma fase de oscilação abaixo do limiar sem atividade, (ii) uma fase de oscilação onde a amplitude e a frequência aumentam de acordo com o estímulo e (iii) uma fase de saturação onde os centros param de oscilar.

4. As frequências dos osciladores aumentam rapidamente quando acontece a mudança do modo andar para nadar. Já que os osciladores dos membros possuem menores frequências que os osciladores do corpo, assim quando os osciladores dos membros saturam, a frequência dos osciladores do corpo já está elevada.

O CPG de [Ijspeert et al. \(2007\)](#) é implementado como um sistema de osciladores não-lineares acoplados baseado no modelo de Kuramoto ([Wu et al., 2009](#)). Semelhante ao modelo da lampreia de [Örjan Ekeberg \(1993\)](#), os neurônios são modelados como osciladores de fase com amplitude controlada:

$$\begin{aligned}\dot{\theta}_i &= 2\pi v_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \\ \ddot{r}_i &= a_i \left( \frac{a_i}{4} (R_i - r_i) - \dot{r}_i \right), \\ x_i &= r_i (1 + \cos(\theta_i)),\end{aligned}$$

onde  $\theta_i$  e  $r_i$  são as variáveis de estado representando a fase e a amplitude do oscilador  $i$ ,  $v_i$  e  $R_i$  determinam a sua frequência e amplitude intrínseca e  $a_i$  é uma constante positiva. O acoplamento entre osciladores são definidos pelos pesos  $w_{ij}$  e a fase enviesada  $\phi_{ij}$ . O sinal oscilatório  $x_i$  representa a fase de atividade do centro.

Os parâmetros de acoplamento  $w_{ij}$  e  $\phi_{ij}$  são configurados de tal maneira que o CPG do corpo produza movimentos oscilatórios em forma de ondas e o CPG dos membros produza os passos da salamandra. Existem acoplamentos unidirecionais vindos dos osciladores dos membros para os osciladores do corpo cuja força é maior que a dos acoplamentos entre os osciladores do corpo, desta maneira o comportamento do CPG dos membros influencia no comportamento do CPG do corpo ([Ijspeert et al., 2007](#)).

## 3.4 Rede Celular Não-linear

As CNNs ( *Cellular Nonlinear Networks* – Redes Celulares Não-lineares) tornam possível implementar dinâmicas não-lineares através de sistemas de osciladores acoplados, fornecendo características importantes na implementação de CPG para o controle de locomoção ([Arena et al., 2004](#)). A célula ou neurônio artificial de uma CNN funciona

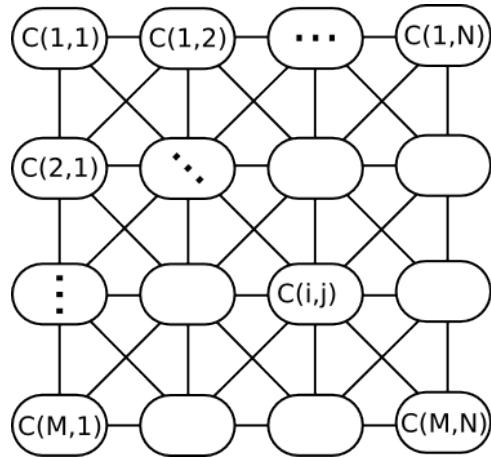


Figura 3.2: Uma CNN de dimensão  $M \times N$

como um processador analógico dinâmico. Duas características marcantes na CNN são: capacidade de processamento paralelo e interconexões essencialmente locais entre células. No entanto, devido à sua dinâmica de propagação, toda a rede interage direta ou indiretamente. Esta é uma característica que distingui a CNN das demais redes ([Chua et al., 1995](#); [Chua e Roska, 1993](#); [Chua e Yang, 1988b](#)). Uma CNN de duas dimensões e de tamanho  $M \times N$  é mostrada na Figura 3.2. Embora a CNN possa assumir qualquer dimensão, o foco deste texto é em duas dimensões, pois a CNN estudada neste trabalho para o controle robótico possui tal dimensão.

### 3.4.1 A Célula CNN

Em vários trabalhos sobre CPGs baseados em osciladores acoplados, o neurônio representa o oscilador e na CNN ele é definido como uma célula ([Arena et al., 2004](#)). Na estrutura de uma CNN cada célula é um sistema dinâmico, conectada apenas a sua vizinhança satisfazendo algumas propriedades: interações limitadas a uma vizinhança de raio finito; e todas as variáveis de estado são de valores contínuos ([Chua e Yang, 1988b](#); [Chua e Roska, 1993](#)). A vizinhança de uma célula em uma CNN de tamanho  $M \times N$  é expressa pela Equação 3.12:

$$N_r(i,j) = \{C(k,l) | \max\{|k - i|, |l - j|\} \leq r, 1 \leq k \leq M, 1 \leq l \leq N\}, \quad (3.12)$$

onde  $C(i,j)$  denota a célula da  $i$ -ésima linha e  $j$ -ésima coluna. O raio  $r$  da CNN utilizada neste trabalho tem tamanho 1.

### 3.4.2 Modelagem Matemática

Segundo Chua e Yang (1988a), uma CNN é caracterizada pelo conjunto de equações diferenciais a seguir:

$$C \frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x} x_{ij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l) y_{kl}(t) + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l) u_{kl}(t) + z_{ij}, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (3.13)$$

$$y_{ij} = f(x_{ij}) = \frac{1}{2}(|x_{ij} + 1| - |x_{ij} - 1|), \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (3.14)$$

$$u_{ij} = E_{ij}, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (3.15)$$

Algumas restrições:

$$|x_{ij}(0)| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (3.16)$$

$$|u_{ij}| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (3.17)$$

A célula básica CNN  $n_{ij}$  possui um estado  $x_{ij}$ , uma entrada  $u_{ij}$ , um limiar  $z_{ij}$ , uma saída  $y_{ij}$  e uma corrente de entrada sináptica  $I_{ij}^N$ . Esta corrente sináptica depende da entrada  $u_{i+k,j+l}(t)$  e do estado  $x_{i+k,j+l}$  de todas as células localizadas na vizinhança de tamanho  $r$  de  $n_{ij}$ , caso  $r = 1$ ,  $k$  e  $l \in \{-1, 0, 1\}$ . A contribuição vinda da entrada  $u_{i+k,j+l}(t)$  de cada célula vizinha é modelada por uma fonte controlada do tipo linear  $b_{kl}u_{i+k,j+l}(t)$ . A contribuição de cada estado  $x_{i+k,j+l}(t)$  de cada célula vizinha  $n_{i+k,j+l}$  é modelada por um fonte controlada do tipo não-linear  $a_{kl}f(x_{i+k,j+l})$ , onde  $f(\cdot)$  descreve uma função escalar não-linear, ver Equação 3.14. Os coeficientes  $a_{kl}$  pertencem ao template de retroalimentação  $A$  e os coeficientes  $b_{kl}$  pertencem ao template de entrada ou de controle  $B$ . O template aparece na forma  $A(i,j,k,l)$ , onde:  $A$  é o nome do template;  $i$  e  $j$  identificam a célula;  $k$  e  $l$  identificam um elemento dentro do template. Um template de realimentação  $A$  de tamanho 3x3 e com uma vizinhança de raio 1 é mostrado na Tabela 3.1. O coeficiente central  $a_{0,0}$  do template  $A$  está relacionado à realimentação da própria célula  $n_{ij}$ . Uma célula CNN é dita autônoma quando não possui entradas externas ou seja  $u_{ij} = u_{i+k,j+l} = 0$ .

A modelagem da corrente sináptica em uma célula CNN é semelhante ao comportamento biológico onde cargas elétricas chegam aos dendritos de uma célula vindas

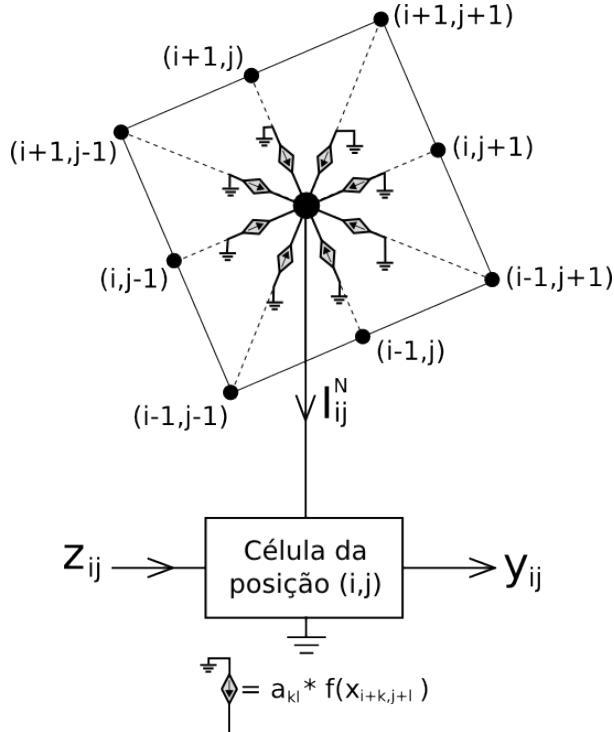


Figura 3.3: Célula autônoma CNN ( $u_{ij} = u_{i+k, j+l} = 0$ ). Cada sinapse (fonte de corrente controlada) é mostrada como uma função não-linear de seu estado atual  $x_{ij}$  e seus estados vizinhos  $x_{i+k, j+l}$ .

$a_{1,-1}$	$a_{1,0}$	$a_{1,1}$
$a_{0,-1}$	$a_{0,0}$	$a_{0,1}$
$a_{-1,-1}$	$a_{-1,0}$	$a_{-1,1}$

Tabela 3.1: *Template A*

dos axônios de outras células, através da sinapse. Um ilustração de como uma célula  $n_{ij}$  relaciona-se com suas vizinhas através da corrente sináptica  $I_{ij}^N(t)$ , é apresentada na Figura 3.3 e a definição matemática na Equação 3.18.

$$I_{ij}^N = \sum_{kl \neq 0,0} a_{kl} f(x_{i+k, j+l}) + \sum_{kl \neq 0,0} b_{kl} u_{i+k, j+l} \quad (3.18)$$

De acordo com Chua *et al.* (1995), a célula básica pode ser descrita utilizando uma equação de estado de primeira ordem como a Equação 3.19:

$$\dot{x}_{ij} = -\frac{1}{C} \left[ \frac{x_{ij}}{R_x} - a_{00}f(x_{ij}) - b_{00}u_{ij} - z_{ij} - I_{ij}^N \right] \quad (3.19)$$

### 3.4.3 Visão de Circuito Elétrico

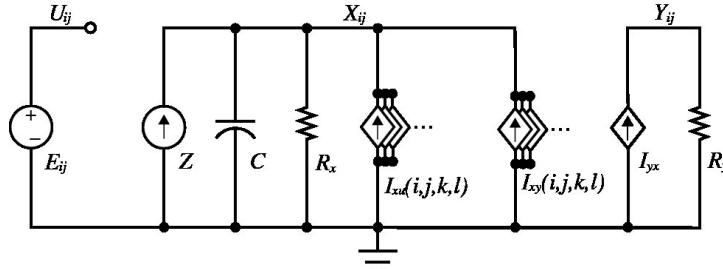


Figura 3.4: Um circuito elétrico de uma célula CNN.

Sob o ponto de vista elétrico, a célula CNN é composta de elementos lineares como capacitores, resistores e fontes. Possui também fontes de corrente não-lineares como ilustrado na Figura 3.4. Segundo Chua e Yang (1988b), cada célula  $C(i,j)$  da CNN contém: uma fonte de tensão independente  $E_{ij}$ ; uma fonte de corrente independente  $Z = z_{ij}$ ; um capacitor linear  $C$ ; dois resistores lineares  $R_x$  e  $R_y$ ; fontes de correntes lineares  $I_{xu}$  e  $I_{xy}$ . As fontes de corrente lineares são descritas por  $I_{xy}(i,j,k,l) = A(i,j,k,l)y_{kl}$  e  $I_{xu}(i,j,k,l) = B(i,j,k,l)u_{kl}$ , onde  $k$  e  $l$  são os índices para as células vizinhas. Para todo  $C(k,l) \in N_r(i,j)$ ,  $u_{kl}$  é a voltagem de entrada e  $y_{kl}$  é a voltagem de saída de cada célula vizinha. Em cada célula o elemento não linear é a fonte de corrente controlada por tensão  $I_{yx} = (1/R_y)f(x_{ij})$ .

### 3.4.4 A CNN Difusão de Reação

Na modelagem de CPG proposta por Arena *et al.* (2004), a CNN utilizada é do tipo Difusão de Reação. Esta CNN é uma rede simples de duas camadas capaz de gerar ondas autônomas. Algumas características marcantes nas ondas autônomas são: possuir forma constante durante a propagação; e não ser afetadas por interferências (Arena *et al.*, 1999). As conexões entre as células são definidas por um *template* de difusão de reação e cada célula é um circuito não linear de segunda-ordem que independente de sua vizinhança comporta-se como um oscilador não-linear (Arena *et al.*, 1997). A autonomia da rede é devido ao fato de não existir sinais de entrada. Ela é chamada de difusão de reação porque é descrita matematicamente por uma versão discretizada de um sistema de equações diferenciais parciais não-lineares, geralmente referenciadas na literatura como equações de difusão de reação (Chua *et al.*, 1995).

As duas camadas da RD-CNN (CNN de Difusão de Reação) interagem dentro de cada célula gerando oscilações, sendo que a interação com a vizinhança é obtida

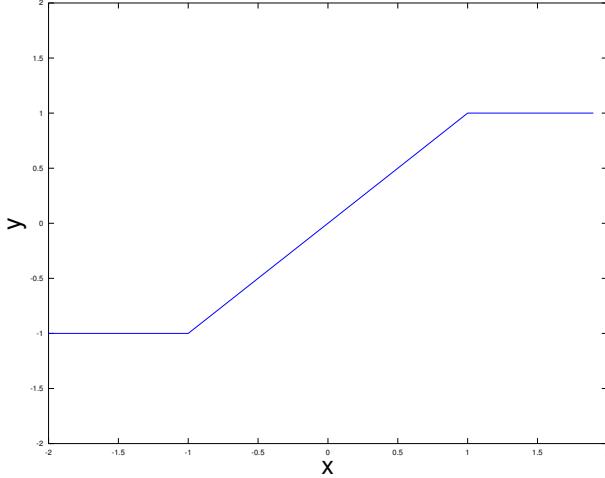


Figura 3.5: Gráfico do estado interno  $x_{ij}$  versos a saída  $y_{ij}$  de uma célula CNN.

0	1	0
1	-4	1
0	1	0

Tabela 3.2: *Template A*

separadamente por meio de dois *templates* de difusão, um para a primeira camada e outro para a segunda camada. Não existe interação direta entre a camada 1 de uma célula  $C(i,j)$  e a camada 2 das células de sua vizinhança e vice versa ([Arena et al., 1997](#)). Os *templates* de difusão de reação são baseados no Laplaciano, cujo intuito é ponderar o efeito das variáveis de estado das células vizinhas ([Arena et al., 1997](#); [Chua et al., 1995](#)). Para exemplificar, um *template* Laplaciano discretizado em duas dimensões é apresentado na Tabela 3.2. As células da RD-CNN para gerar as ondas autônomas são descritas pelo seguinte sistema de segunda ordem ([Arena et al., 1999](#)):

$$\dot{x}_{1,i,j} = -x_{1,i,j} + (1 + \mu + \varepsilon)y_{1,i,j} - s_1y_{2,i,j} + i_1 \quad (3.20)$$

$$\dot{x}_{2,i,j} = -x_{2,i,j} + s_2y_{1,i,j} + (1 + \mu - \varepsilon)y_{2,i,j} + i_2 \quad (3.21)$$

Com  $i = 0, 1, \dots, M - 1$  e  $j = 0, 1, \dots, N - 1$ . A influência das constantes  $\mu$  e  $s$  no comportamento de um sistema dinâmico autônomo simples, semelhante ao de uma célula RD-CNN, é descrito no apêndice A. Os sinais dos estados internos e das saídas de uma rede com dois neurônios são expressos nos gráficos da Figura 3.6.

A representação vetorial da RD-CNN ([Arena et al., 1998, 1999](#)) é descrita pela

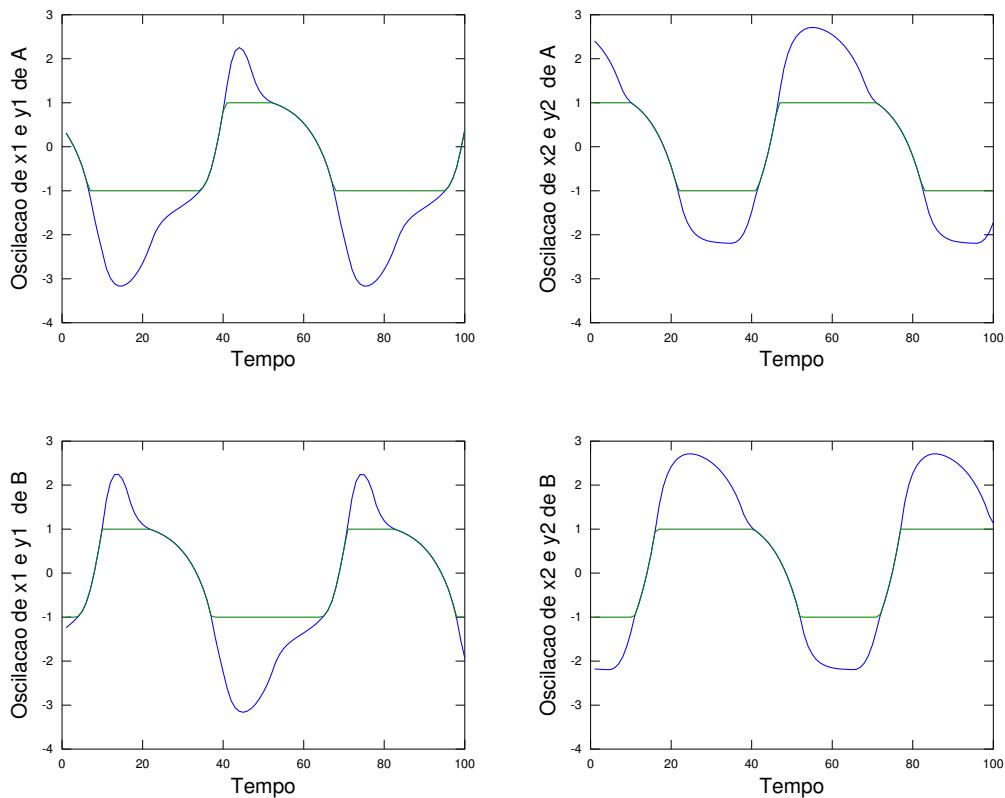


Figura 3.6: Gráfico dos valores dos estados internos ( $x_1$  e  $x_2$ ) e das saídas ( $y_1$  e  $y_2$ ) para uma rede CNN com dois neurônios A e B cada um com duas camadas. A linha verde indica os valores de  $y$  e a linha azul indica os valores de  $x$ .

Equação 3.22:

$$\dot{\mathbf{x}}_{ij} = -\mathbf{x}_{ij} + A * \mathbf{y}_{ij} + B * \mathbf{u}_{ij} + I \quad (3.22)$$

onde  $\dot{\mathbf{x}}_{ij} = [\dot{x}_{1,ij} \quad \dot{x}_{2,ij}]^T$ ,  $\mathbf{x}_{ij} = [x_{1,ij} \quad x_{2,ij}]^T$ ,  $\mathbf{y}_{ij} = [y_{1,ij} \quad y_{2,ij}]^T$  e  $\mathbf{u}_{ij} = [u_{1,ij} \quad u_{2,ij}]^T$  representam a variação no tempo do estado interno, o estado interno propriamente dito, a saída e a entrada da CNN, respectivamente. A, B e I representam os *templates* de realimentação, de controle e de bias, respectivamente. O *template* A, de difusão discreta, define a relação das células com seus vizinhos sendo expresso da seguinte maneira:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}; \quad B = 0; \quad I = \begin{pmatrix} i_1 \\ i_2 \end{pmatrix}.$$

$$A_{12} = -A_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad (3.23)$$

$A_{11}$  e  $A_{22}$  são matrizes 3x3 e definem a relação com as células vizinhas para as camadas 1 e 2 respectivamente. O operador de convolução bidimensional \* para um *template* T é definido por Chua e Yang (1988b) como:

$$T * x_{ij} = \sum_{C(k,l) \in N_r(i,j)} \mathbf{T}(k-i, l-j) x_{kl}; \quad (3.24)$$

onde  $\mathbf{T}(m,n)$  é um elemento do *template* cujos índices m e n pertencem ao conjunto  $\{-1, 0, 1\}$ .  $T(0,0)$ , por exemplo, é o elemento central da matriz. Esta forma de indexação é melhor ilustrada na Tabela 3.1.

## 3.5 O Modelo de Arena

Arena *et al.* (1999) investigaram o problema de locomoção artificial em um robô caminhante de seis patas para reproduzir o caminhar de insetos. Eles utilizaram uma CNN para construir um gerador central de padrões. Arena *et al.* (2004) também desenvolveram um mecanismo de controle adaptativo de alto-nível capaz de encontrar os parâmetros que definem o comportamento do CPG durante a locomoção do robô. Estes parâmetros são organizados na forma de *templates*, um estudo mais detalhado sobre estes *templates* é apresentado na Seção 3.5.1. O controle adaptativo é elaborado com base em um mapa motor, descrito na Seção 3.5.2.

Os movimentos locomotores de um membro são controlados por uma célula da

CNN, já a coordenação entre membros esta relacionada às conexões entre as células. Os neurônios oscilam na mesma frequência e possuem fase constante entre eles. O que diferencia um modo de locomoção de outro é justamente a defasagem entre perna, ou seja, a defasagem entre as oscilações dos neurônios. Modos distintos de locomoção são executados através do uso de conexões distintas, correspondendo a conjuntos distintos de *templates* (Arena *et al.*, 2004).

Os modos de locomoção mais básicos como andar, mover em velocidade moderada e correr podem ser combinados para formar um modo de locomoção intermediário e consequentemente uma transição mais suave entre estes modos de locomoção básicos. Esta transição é alcançada ativando mais de um *template* ao mesmo tempo, já que cada *template* define um modo de locomoção (Arena *et al.*, 2004). O modo de locomoção contínuo é definido através do *template* de realimentação  $A_r$ , como a seguir:

$$A_r = \alpha A_f + \beta A_m + \gamma A_w, \quad (3.25)$$

onde o primeiro termo da equação representa o modo de locomoção rápido  $A_f$ , o segundo representa o modo de locomoção moderado  $A_m$  e o terceiro, o modo de locomoção lento  $A_w$ . Os parâmetros  $\alpha$ ,  $\beta$  e  $\gamma$  variam entre 0 e 1 (Arena *et al.*, 2004), sendo o controle destes parâmetros ( $\alpha$ ,  $\beta$  e  $\gamma$ ) feito por meio de um mapa motor, que a partir de uma velocidade de referência é capaz de determinar cada um deles.

### 3.5.1 Abordagem Multi-*Template*

Na RD-CNN, a criação de um determinado modo de locomoção depende de como a rede esteja estruturada. O sincronismo entre células esta relacionado à maneira como a células são conectadas. Como cada célula esta conectada a uma perna, o sincronismo entre células dita a defasagem do movimento das pernas. *Template* determina a estrutura da RD-CNN, logo o modo de locomoção resultante, no entanto escolher o *template* apropriado para obter um determinado modo de locomoção não é uma tarefa trivial (Arena *et al.*, 2002)

Arena *et al.* (2002) desenvolveram um heurística chamada de Abordagem Multi-*Template* para encontrar a estrutura da RD-CNN que gera o padrão de locomoção desejado. Então, a partir de uma RD-CNN estruturada em um anel de células, vários padrões de locomoção podem ser obtidos. Bastando apenas reconfigurar as conexões entre as células. Para mudar o padrão de locomoção basta usar um número diferente de células que constituem o anel e rearrumar as conexões entre neurônios (células RD-CNN). Isto corresponde à reorganização da topologia biológica da rede neural. As

sinapses são reorganizadas envolvendo novos neurônios ou diminuindo o numero de neurônios na rede. Para obter o respaldo biológico o modelo de propagação do sinal (pulso elétrico) da RD-CNN manteve-se compatível com uma modelagem matemática da sinapse química.

Baseado no comportamento da sinapse química ([Arena et al., 2002](#)) propuseram uma heurística para auxiliar na construção de uma RD-CNN que se comportasse de maneira desejada. Para facilitar a manipulação da RD-CNN e projetar o CPG, o problema foi decomposto em uma estrutura simples semelhante a um anel formada por um núcleo composto apenas de dois neurônios. A partir desta rede simples novos neurônios são acoplados para alcançar o CPG desejado. As conexões sinápticas entre os neurônios desta rede podem ser do tipo excitatória ou inibitória. O tipo de sinapse determina o sentido em que a sequência de disparos dos neurônios ocorrem, podendo ser horária ou anti-horária. Na construção desta rede em forma de anel o número de fases  $N$  contidas em um modo de locomoção indica a quantidade inicial de neurônios, por exemplo, no modo de locomoção trípode alternado (três pernas sincronizadas) que possui duas fases ( $N = 2$ ) a quantidade de neurônios no anel será 2. Mas, ao projetar um CPG a quantidade final de neurônios  $n$  é escolhida a mesma quantidades de membros controlados pela rede.

As recomendações heurísticas para montar um CPG, chamadas de abordagem *multi-template* ([Arena et al., 2001, 2002](#)), são descritas a seguir:

1. A quantidade de diferentes fases  $N$  que o CPG gera está relacionada a mesma quantidade inicial de neurônios  $N$  presentes no anel;
2.  $n - N$  neurônios são adicionados a rede, as conexões são estabelecidas e também o sincronismo dos novos neurônios com os neurônios já existentes;
3. Os pesos sinápticos influenciam no período de oscilação. O mesmo valor é escolhido para todas as sinapses da rede. Este peso sináptico  $\varepsilon$  é escolhido com o objetivo de garantir a estabilidade do padrão de sincronização no qual cada neurônio dispara em uma fase diferente. Para que o total de pesos sinápticos na entrada de um neurônio permaneça a mesma em todos os neurônios,  $\varepsilon/k$  onde  $k$  é a quantidade de pesos na entrada de um neurônio.

Depois de estabelecida a estrutura da rede, o próximo passo é escrever o conjunto de *templates* correspondente a esta estrutura. Para elaborar estes *templates*, as conexões entre neurônios são levadas em conta. Neste caso, o *template* é dependente do espaço, ou seja, da posição que a célula ocupa na rede. Cada conjunto de *templates* gera um

---

modo de locomoção, logo, mudar entre modos de locomoção significa alterar o conjunto de *templates* atuantes neste sistema ([Arena et al., 2002](#)).

### 3.5.2 Mapas Motores

A formação da topologia que constitui os mapas motores no cérebro é fundamentada na representação de sinais de entrada sensoriais e na habilidade de executar uma ação em resposta a um dado estímulo. Neurônios no cérebro são organizados em agrupamentos locais aptos a executar tarefas como enviar o sinal apropriado para o músculo. Estes agrupamentos neurais, inspiradas no paradigma de Mapas de Kohonen, mapeiam as excitações em movimentos. Desta forma, estas redes são aptas a reagir às excitações disparando um movimento como o córtex motor no cérebro ([Arena et al., 2004](#); [Ritter et al., 1992](#)).

Um Mapa Motor possui uma arquitetura em duas camadas: uma dedicada a armazenar os pesos da entrada e outra dedicada ao pesos da saída. Ao apresentar uma velocidade de referência para a rede, um neurônio da camada de entrada é ativado que por sua vez ativa um conjunto de neurônios da camada de saída, selecionando um conjunto de parâmetros definidores dos pesos da Equação 3.25. Como visto anteriormente, a Equação 3.25 determina o modo de locomoção corrente, por tanto a velocidade resultante de deslocamento do robô. A fase de aprendizagem lida com a atualização tanto dos pesos de entrada quanto dos da saída, permitindo ao mapa aprender ações relacionadas ao controle motor. O algoritmo de aprendizagem é uma extensão do algoritmo vencedor-leva-tudo (*winner-take-all*). O Mapa Motor aprende uma nova velocidade de referência apenas se a variação da função de recompensa, Equação 3.26, for maior que um determinado limiar.

$$Reward = -(V_{ref} - v)^2 \quad (3.26)$$

## 3.6 Considerações

Este capítulo apresentou duas abordagens usadas na implementação de CPGs, uma biologicamente inspirada no CPG da lampreia e a outra construída a partir de uma rede celular não-linear. Em ambas as abordagens, a modelagem matemática do CPG foi realizada através de equações diferenciais, mais especificamente equações de osciladores não-lineares ([Arena et al., 2002](#); [Ijspeert et al., 2007](#)). Estas abordagens implicam no ajuste de um conjunto de parâmetros ou até mesmo na modificação destas equações

para produzir o modo de locomoção desejado.

As técnicas mais utilizadas para resolver o problema de locomoção são biologicamente inspiradas em CPGs. O objetivo inicial deste trabalho era transformar um mapa auto-organizável em um CPG, mas esta estratégia tornou-se inviável, pois as modelagens dos CPGs encontradas na literatura não são compatíveis com as modelagens de mapas auto-organizáveis. Os CPGs são descritos por equações diferenciais enquanto que os mapas auto-organizáveis são descritos algorítmicamente e precisam de uma base de dados para produzirem alguma resposta. Então, o caminho seguido para projetar um modelo de controle de locomoção de robôs com membros levando em conta um mapa auto-organizável de topologia variável foi escolher dentre os modelos de CPGs aquele que tivesse maior relação com os mapas auto-organizáveis. O modelo escolhido foi o de [Arena et al. \(2004\)](#) que utiliza em sua modelagem um mapa motor, uma derivação de mapa auto-organizável. Embora Arena modele matematicamente os osciladores do CPG por meio da CNN.

A abordagem proposta nesta dissertação possui duas características interessantes. Primeiro, não necessita de modelagem com base em equações diferenciais. Segundo é inovadora, pois não foi encontrado nenhum registro na literatura de alguma abordagem semelhante para o controle de locomoção de robôs com membros. O STRAGEN proposto por [Benante e Araujo \(2007\)](#), utilizado como base do modelo apresentado no Capítulo 5, destaca-se dos outros mapas por sua flexibilidade. Pois o critério de vizinhança pode ser selecionando de qualquer parte da amostra e o mesmo pode ser feito para o critério de atividade.

O Capítulo 5 apresenta o modelo proposto nesta dissertação. Este modelo é capaz de construir o movimento oscilatório a partir de posturas do robô, dadas como amostras para o modelo, fazendo um mapeamento direto entre o comportamento observado e o comportamento resultante. Diferentemente do que acontece nas abordagens baseadas em osciladores não-lineares onde o comportamento observado precisa ser modelado matematicamente através de equações diferenciais e a partir destas equações produzir o comportamento desejado.

# 4

## Mapas Auto-Organizáveis

Os mapas auto-organizáveis em essência constroem um mapeamento de um espaço de entrada de alta dimensionalidade em um espaço de estruturas topológicas de baixa dimensão. Neste mapeamento, elementos vizinhos no espaço de entrada são mapeados em regiões vizinhas do espaço de estruturas topológicas.

Para melhor entender o STRAGEN (*State Trajectories Generator*) utilizado na proposição do modelo tratado nesta dissertação, este capítulo apresenta alguns mapas auto-organizáveis relacionados ao STRAGEN. O primeiro modelo apresentado é o clássico modelo de Kohonen, o ponto de partida dos mapas auto-organizáveis. Em seguida apresenta o modelo GCS (Estruturas Celulares Crescentes) projetado para superar algumas limitações do modelo de Kohonen, originadas devido a sua estrutura rígida. Os outros modelos são: o GNG, muito semelhante ao GCS; e o GWR que traz algumas novidades em relação aos modelos citados anteriormente.

### 4.1 Algumas Definições

Antes de mostrar os modelos de mapas auto-organizáveis algumas definições precisam ser levadas em conta: ([Kohonen, 1982, 1998](#); [Kohonen e Hari, 1999](#); [Fritzke, 1994, 1995](#)):

- Estímulo de entrada, sinal de entrada, ou apenas entrada da rede é um vetor de dados  $n$ -dimensional,  $\xi = [\xi_1 \xi_2 \dots, \xi_n]$ , isto é, uma lista de números que representam os valores do estímulo em cada dimensão;
- Neurônio, unidade, nó ou célula  $n_i$  possui um conjunto de valores numéricos ou pesos sinápticos,  $w_i$ . O vetor  $w_i = [w_{i1}, w_{i2}, \dots, w_{in}]$  possui a mesma dimensão de  $\xi$  e pode ser considerado uma posição no espaço de entrada;

- O neurônio vencedor  $s$  possui vetor sináptico  $\mathbf{w}_s$ , também conhecido como neurônio mais adaptado, é aquele que possui o maior grau de semelhança com o estímulo de entrada;
- Vértice ou conexão, um conceito comum nos mapas auto-organizáveis, unem os neurônios para formar a sua vizinhança;

## 4.2 SOM

Segundo [Kohonen \(1998\)](#), um SOM (Mapa Auto-Organizado – *Self-Organizing Map*) é uma ferramenta matemática para visualização de dados de alta-dimensionalidade. Ele cria um mapeamento de uma distribuição de alta-dimensão em uma grade regular de baixa-dimensão. Sendo capaz de compactar informações preservando os relacionamentos topológicos e as métricas mais importantes dos dados originais. Com base nestas características dois aspectos são evidenciados, o de abstração e exibição simplificada da informação. Estes dois aspectos podem ser utilizados de diversas maneiras em uma variedade de aplicações práticas como em reconhecimento de voz, análise de imagem, processos industriais de controle, organização automática de documentos numa biblioteca, visualização de registros financeiros etc

Os estímulos chegam para todas as unidades na rede, no entanto a unidade mais ativa é aquela cujo vetor de pesos  $\mathbf{w}_i$  é mais próximo do estímulo de entrada. Esta unidade chamada de vencedora  $\mathbf{w}_s$ , mantém-se ativa induzindo à ativação dos neurônios vizinhos. Um requisito para a auto-organização é que os pesos sinápticos de uma unidade sejam modificados apenas na vizinhança local da unidade vencedora e todos os pesos modificados assemelhem-se ao estímulo atual com mais precisão que no passado. Diferentes sinais de entrada em diferentes tempos afetam regiões diferentes na grade de unidades. Deste modo, depois de muitos passos de aprendizagem, os pesos sinápticos ( $\mathbf{w}_i$ ) começam a adquirir valores que se relacionam suavemente dentro desta grade de maneira equivalente aos estímulos do espaço de entrada ( $\xi$ ) ([Kohonen e Hari, 1999](#)).

A unidade vencedora  $\mathbf{w}_s$  é aquela possuidora do maior grau de semelhança com o estímulo de entrada. A equação 4.1 descreve o processo de comparação:

$$\forall i, \|\xi(t) - \mathbf{w}_s(t)\| \leq \|\xi(t) - \mathbf{w}_i(t)\|. \quad (4.1)$$

A métrica de comparação normalmente escolhida é a distância Euclidiana ([Kohonen, 1982, 1998; Kohonen e Hari, 1999](#)).

---

O algoritmo SOM modifica os pesos sinápticos das unidades vizinhas da unidade vencedora de modo a aumentar o grau de semelhança entre estas unidades e estímulo de entrada. A vizinhança do neurônio vencedor pode ser descrita pela função de vizinhança  $h_{ci}$  na equação 4.2, que atinge seu máximo para o vencedor, isto é  $i = s$ . Esta função  $h_{si}$  é normalmente descrita pela curva Gaussiana e retorna um valor escalar:

$$h_{si} = \alpha(t) \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_s\|}{2\sigma^2(t)}\right), \quad (4.2)$$

onde  $0 < \alpha(t) < 1$  é a taxa de aprendizagem,  $\mathbf{r}_i \in \mathbb{R}^2$  e  $\mathbf{r}_s \in \mathbb{R}^2$  são as posições vetoriais dos elementos na grade, e  $\sigma(t)$  corresponde a largura ou raio da função de vizinhança. Os parâmetros  $\sigma(t)$  e  $\alpha$  decrescem monotonicamente com o decorrer tempo  $t$  (Kohonen, 1998; Kohonen e Hari, 1999).

Na etapa de aprendizagem um estímulo de entrada  $\xi(t)$  modifica os valores dos pesos sinápticos  $\mathbf{w}_i(t)$  para novos valores  $\mathbf{w}_i(t+1)$ ,  $t$  indica a iteração atual. A Equação 4.3 mostra como a atualização dos pesos sinápticos da unidade vencedora e de suas vizinhas dependem dos estímulos de entrada: quanto maior a diferença entre o estímulo e os pesos sinápticos de uma unidade, maior será o salto em direção ao vetor que representa o estímulo de estrada (Kohonen e Hari, 1999).

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{si}(t)(\xi(t) - \mathbf{w}_i(t)). \quad (4.3)$$

### 4.3 GCS

Algumas limitações do SOM motivaram a criação de um modelo de rede neural de topologia variável. As limitações do SOM estão relacionadas à estrutura e dimensão fixa decididas previamente. Como a distribuição de probabilidade dos dados de entrada não é conhecida, a configuração da estrutura da rede decidida previamente pode não capturar bem este dados de entrada, consequentemente comprometendo a precisão da rede. A rede GCS (*Growing Cell Structures*) proposta por Fritzke (1994), semelhante à rede SOM, é capaz de mapear um estímulo de entrada  $n$ -dimensional, denotado por  $V = \mathbb{R}^n$ , em um estrutura com topologia variável  $A$  de  $k$  dimensões. Este mapeamento contém as seguintes propriedades:

- Estímulos similares de entrada são mapeados em unidades de  $A$  topologicamente próximas;
- Elementos topologicamente próximos em  $A$  possuem sinais similares de entrada;

- Regiões de  $V$  onde a densidade de probabilidade da distribuição do vetor de entrada é alta deve ser representada por muitas unidades correspondentes em  $A$ .

A topologia inicial da rede  $A$  é um *simplex* de dimensão  $k$ . Para  $k = 1$  o *simplex* é um segmento de reta, para  $k = 2$  um triângulo e para  $k = 3$  ou maior, um tetraedro ou hiper-tetraedros. Os vértices do *simplex* são os neurônios, as arestas ou conexões representam a relação de vizinhança topológica. Durante o processo de auto-organização, novos neurônios são adicionadas à rede e neurônios não relevantes são removidas. Ao término de cada iteração a rede inteira mantém sua estrutura *simplex* consistente com dimensão  $k$  (Fritzke, 1994).

Cada neurônio  $n_i$  está associada a um vetor sináptico  $\mathbf{w}_i$  de dimensão  $n$ . Este vetor pode ser entendido como a posição de  $n_i$  no espaço vetorial de entrada. Um mapeamento  $\phi_w$  do espaço vetorial de entrada  $V$  para a rede  $A$  é definido como o mapeamento entre o estímulo de entrada e o neurônio vencedor. Formalmente escrito por Fritzke (1994):

$$\phi_w : V \rightarrow A, (\xi \in V) \mapsto (\phi_w(\xi) \in A) \quad (4.4)$$

onde  $\phi_w(\xi) = \mathbf{w}_s$  é o vetor sináptico do neurônio vencedor encontrado matematicamente com a equação 4.1. Com o mapeamento  $\phi_w$  que associa cada estímulo de entrada  $\xi$  a um vetor sináptico  $\mathbf{w}_i$ ,  $V$  é particionado em regiões  $F_i$  ( $i \in A$ ), cada uma formada de localizações que possuem em comum o vetor sináptico  $\mathbf{w}_i$  mais próximo. Este particionamento é conhecido como decomposição de Voronoi, e as regiões são denotadas como regiões de Voronoi (Fritzke, 1994).

A princípio, a adaptação dos vetores sinápticos no GCS é feita como Kohonen propôs:

1. Determine a unidade mais semelhante para o estímulo de entrada atual.
2. Torne a unidade mais semelhante e a sua vizinhança topológica ainda mais semelhante a este estímulo.

No modelo de Kohonen, a taxa de aprendizagem, chamada de força de adaptação por Fritzke (1994), diminui com o decorrer das iterações. Além disso a vizinhança topológica é escolhida grande no início e também diminui com o tempo. Existem, no entanto, duas diferenças importantes entre o GCS e o SOM, são elas:

- A força de adaptação é constante no tempo, mas especificamente os parâmetros de adaptação  $\varepsilon_s$  e  $\varepsilon_n$  relacionados a unidade mais semelhante e aos neurônios vizinhos respectivamente;

- Apenas a unidade mais semelhante e seus vizinhos topologicamente diretos são adaptados.

O passo de adaptação do algoritmo GCS pode ser descrita como a seguir [Fritzke \(1994\)](#):

1. Escolha um estímulo de entrada  $\xi$  levando em conta a distribuição de probabilidade  $P(\xi)$ ;
2. Encontre o neurônio vencedor  $s$ ;
3. Mova  $s$  e sua vizinhança topológica direta em direção a  $\xi$ :

$$\Delta \mathbf{w}_s = \varepsilon_s (\xi - \mathbf{w}_s) \quad (4.5)$$

$$\Delta \mathbf{w}_{n_s} = \varepsilon_n (\xi - \mathbf{w}_{n_s}) \quad (\forall n_s \in N_s), \quad (4.6)$$

onde  $N_s$  denota o conjunto dos vizinhos topológicos do neurônio  $s$ ;

4. Incremente o contador de vitórias de  $s$ ,

$$\Delta \tau_s = 1, \quad (4.7)$$

5. Decremente todos os contadores de vitórias:

$$\Delta \tau_i = -\alpha \tau_i \quad (\forall i \in A), \quad (4.8)$$

assim, as vitórias mais recentes recebem uma ponderação mais forte do que os anteriores.

A distribuição de probabilidade  $P(\xi)$  não é conhecida explicitamente, mas pode ser estimada com contador local de vitórias. Este contador indica a frequência relativa dos estímulos de entrada recebidos por um neurônio. A frequência relativa de vitórias de um neurônio  $i$  é dada por

$$h_i = \tau_i / \sum_{j \in A} \tau_j. \quad (4.9)$$

Espera-se que a frequência relativa de vitórias de cada neurônio seja similar. Um valor elevado de  $h_i$ , portanto, indica uma boa posição para inserir um novo neurônio desde que o novo neurônio seja capaz de reduzir este valor elevado [Fritzke \(1994\)](#).

Sempre depois de um número fixo  $\lambda$  de passos de adaptação o neurônio  $q$  com a seguinte propriedade é determinado

$$h_q \geq h_i \forall i \in A. \quad (4.10)$$

Depois, o vizinho  $f$  de  $q$  que possui a maior distância para o espaço de entrada é determinado, de acordo com

$$\|\mathbf{w}_f - \mathbf{w}_q\| \geq \|\mathbf{w}_i - \mathbf{w}_q\| \quad (\forall i \in N_q). \quad (4.11)$$

O novo neurônio  $r$  é inserido entre  $q$  e  $f$ . Este novo neurônio é conectado com outros neurônios de maneira que a estrutura da rede continue consistente com *simplices* de dimensão  $k$ . O vetor sináptico  $r$  é inicializado como

$$\mathbf{w}_r = 0.5(\mathbf{w}_q + \mathbf{w}_f). \quad (4.12)$$

A inserção de  $r$  gera a uma nova região de Voronoi  $F_r$  dentro do espaço de entrada. Ao mesmo tempo as regiões de Voronoi na vizinhança topológica de  $r$  diminuem. Esta mudança reflete nos contadores,  $\tau_i$ , da seguinte maneira

$$\Delta\tau_i = \frac{|F_i^{(novo)}| - |F_i^{(velho)}|}{|F_i^{(velho)}|} \tau_i \quad (\text{para todo } i \in N_r). \quad (4.13)$$

Onde  $|F_i|$  é o volume da região  $F_i$  de dimensão  $n$ . O valor inicial do contador de vitórias do novo neurônio é definido como

$$\tau_r = - \sum_{i \in N_r} \Delta\tau_i. \quad (4.14)$$

## 4.4 GNG

A rede GNG (*Growing Neural Gas*) proposta por [Fritzke \(1995\)](#) é outro modelo de topologia variável muito semelhante ao GCS e caracterizado da seguinte maneira:

- Um conjunto  $A$  de neurônios, em que cada neurônio  $i \in A$  tem um vetor sináptico associado  $w_i \in \mathbb{R}^n$ .
- Um conjunto  $N$  de conexões entre pares de neurônios. Estas conexões não ponderadas têm o propósito de definir a estrutura topológica da rede.

- O estímulo de entrada  $\xi$  obedece a alguma função de densidade de probabilidade  $P(\xi)$  desconhecida

O crescimento do GNG ocorre levando em conta uma avaliação local com medidas estatísticas obtidas nas iterações ou passos de adaptação do passado, semelhante ao modelo GCS de [Fritzke \(1995\)](#). O algoritmo GNG é apresentado a seguir:

1. Inicie com duas unidades  $a$  e  $b$  em posições aleatórias  $\mathbf{w}_a$  e  $\mathbf{w}_b$  em  $\mathbb{R}^n$ .
2. Obtenha um estímulo de entrada  $\xi$  a partir da distribuição de probabilidade  $P(\xi)$ ;
3. Encontre a unidade vencedora  $s_1$  e a segunda unidade vencedora  $s_2$ .
4. Incremente a idade de todas as conexões de  $s_1$ .
5. Acumule o erro local de  $s_1$ :

$$\Delta\text{error}(s_1) = \|\mathbf{w}_{s_1} - \xi\|^2 \quad (4.15)$$

6. Mova  $s_1$  e seu vizinhos topologicamente diretos na direção de  $\xi$  proporcionalmente a  $\varepsilon_s$  e  $\varepsilon_n$ , respectivamente, segundo:

$$\Delta\mathbf{w}_{s_1} = \varepsilon_s(\xi - \mathbf{w}_{s_1}) \quad (4.16)$$

$$\Delta\mathbf{w}_n = \varepsilon_n(\xi - \mathbf{w}_n), \forall n \in N_{s_1} \quad (4.17)$$

7. Se  $s_1$  e  $s_2$  possuir uma conexão, atribua o valor zero a esta conexão. Se tal conexão não existe, crie.
8. Remova conexões com idade maior que  $a_{max}$ . Se resultar em neurônios desconexos, remova-os.
9. Se o número de estímulos de entrada apresentados a rede até o momento for um múltiplo de um parâmetro  $\lambda$ , insira um novo neurônio como a seguir:
  - Determine o neurônio  $q$  com o máximo erro acumulado;
  - Insira um novo neurônio  $r$  na metade da distância entre  $q$  e seu vizinho  $f$  com o maior erro:

$$\mathbf{w}_r = 0.5(\mathbf{w}_q + \mathbf{w}_f). \quad (4.18)$$

- Insira conexões entre novo neurônio  $r$  e os neurônios  $q$  e  $f$ , e remova a conexão original entre  $q$  e  $f$ ;

- Decrementa a variável de erro de  $q$  e  $f$  multiplicando por uma constante  $\alpha$ . Inicialize a variável de erro de  $r$  com o mesmo valor da variável de erro de  $q$ .
10. Decrementa todas as variáveis de erro, multiplicando-as por uma constante  $d$ .
  11. Se nenhum critério de parada (exemplo, tamanho da rede ou alguma medida de desempenho) for atingido retorne ao passo 1.

## 4.5 GWR

A rede GWR (*Grow When Required*) proposta por [Marsland et al. \(2002\)](#) tem dois componentes importantes, os neurônios e as conexões. Tanto os neurônios quanto as conexões podem ser criados e destruídos durante o processo de aprendizagem. Diferente do GNG, que adiciona um neurônio a cada  $\lambda$  iterações, o GWR adiciona um neurônio a qualquer momento. O posicionamento deste novo neurônio depende da entrada e do neurônio vencedor, ao invés de adicionar onde o erro acumulado é maior, como no GNG.

Um novo neurônio é adicionado quando a atividade do neurônio vencedor não é alta o suficiente. A atividade de um neurônio é calculada através da função de distância Euclidiana entre seu vetor de pesos e o vetor de entrada. Cada neurônio é equipado com uma variável para registrar a sua frequência de vitórias ([Marsland et al., 2002](#)).

Uma maneira de registrar a frequência de vitórias é armazenar o valor 1 em uma variável e fazer o valor desta variável decrescer exponencialmente de 1 para 0 a medida que o neurônio vá vencendo. Deste modo, um neurônio novo terá o valor 1 e os neurônios que disparam frequentemente terão o valor próximo a 0. Os vizinhos do neurônio vencedor também são atualizados, mas o valor armazenado decresce mais lentamente ([Marsland et al., 2002](#)).

Ao apresentar uma entrada à rede, a atividade de cada neurônio no mapa é calculada e o vencedor é escolhido. Se este neurônio vencedor representar bem a entrada então a atividade deste neurônio será próxima a 1. Neste caso, o nível de adaptação aplicado ao neurônio vencedor bem como sua vizinhança é pequeno. Entretanto, se a atividade da rede é menor que o limiar de inserção  $a_T$ , então o neurônio vencedor foi adicionado recentemente à rede ou ele e o padrão de entrada são diferentes então o neurônio vencedor precisa ser adaptado. Se o neurônio foi adicionado recentemente então a variável que registra os disparos terá um valor elevado, perto de 1. Caso contrário, um novo neurônio é adicionado entre o neurônio vencedor e a entrada ([Marsland et al., 2002](#)).

Um limiar de ativação identifica quando o estímulo de entrada já foi aprendido pela rede, caso o estímulo de entrada provoque uma atividade na rede menor que o limiar estabelecido, então este estímulo deve ser aprendido pela rede. Assim, baixa atividade significa pouca semelhança entre a rede e a entrada. O valor do limiar de inserção  $a_T$  exerce grande influência na quantidade de neurônios inseridos na rede. Se o valor é configurado bem próximo a 1 então mais neurônios serão produzidos e a entrada será bem representada. Para valores pequenos de  $a_T$  poucos neurônios são adicionados (Marsland *et al.*, 2002).

Para o algoritmo descrito a seguir, considere  $A$  o conjunto de todos os nós do mapa e  $C \in A \times A$  o conjunto de conexões entre nós contidos no mapa. A distribuição dos dados de entrada representada por  $p(\xi)$ , e a entrada por  $\xi$ . O vetor de pesos do neurônio  $n_i$  como  $\mathbf{w}_i$  (Marsland *et al.*, 2002).

Inicialização: Coloque dois nós no conjunto  $A$

$$A = \{n_1, n_2\}, \quad (4.19)$$

com  $n_1$  e  $n_2$  inicializados randomicamente a partir de  $p(\xi)$ . O conjunto de conexões  $C$  é inicializado vazio

$$C = \emptyset. \quad (4.20)$$

O algoritmo é apresentado a seguir:

1. Apresente para a rede uma amostra  $\xi$  dos dados de entrada;
2. Para cada nó  $i$  da rede, calcule a distância para amostra de entrada  $\|\xi - \mathbf{w}_i\|$ .
3. Selecione o nó mais semelhante, e o segundo mais semelhante ao padrão de entrada, isto é, os nós  $s_1, s_2 \in A$  tal que

$$s_1 = \arg \min_{n \in A} \|\xi - \mathbf{w}_n\| \quad (4.21)$$

e

$$s_2 = \arg \min_{n \in A / \{s_1\}} \|\xi - \mathbf{w}_n\| \quad (4.22)$$

onde  $\mathbf{w}_n$  é o vetor de pesos do nó  $n$ .

4. Caso não exista uma conexão entre  $s_1$  e  $s_2$ , crie

$$C = C \cup \{(s_1, s_2)\}, \quad (4.23)$$

caso exista, atribua o valor 0 para a idade da conexão.

5. Calcule a atividade da unidade mais semelhante

$$a = \exp(-\|\xi - \mathbf{w}_{s_1}\|). \quad (4.24)$$

6. Se a atividade  $a$  for menor que o limiar de atividade  $a_T$  e a quantidade de ativações do neurônio for alta (o valor presente na variável que registra os disparos é menor que o limiar  $h_T$ ), então um novo neurônio deve ser adicionado entre os dois neurônios mais semelhantes ( $s_1$  e  $s_2$ ) ao padrão de entrada

- Adicione um novo neurônio,  $r$

$$A = \cup\{r\}. \quad (4.25)$$

- Crie o novo vetor de pesos, atribuindo aos pesos a média entre pesos do neurônio vencedor e o vetor de entrada

$$\mathbf{w}_r = (\mathbf{w}_{s_1} + \xi)/2. \quad (4.26)$$

- Insira conexões entre  $r$  e  $s_1$ , e entre  $r$  e  $s_2$

$$C = C \cup \{(r, s_1), (r, s_2)\}. \quad (4.27)$$

- Remova a conexão entre  $s_1$  e  $s_2$

$$C = C / \{(s_1, s_2)\}. \quad (4.28)$$

7. Se um novo neurônio não for adicionado, ajuste a posição do neurônio vencedor e dos neurônios conectados a ele, os vizinhos  $i$ ,

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b \times h_{s_1} \times (\xi - \mathbf{w}_{s_1}) \quad (4.29)$$

$$\Delta \mathbf{w}_i = \varepsilon_n \times h_i \times (\xi - \mathbf{w}_i), \forall i \in N_{s_1} \quad (4.30)$$

onde  $0 < \varepsilon_n < \varepsilon_b < 1$  e  $h_{s_1}$  é o valor do registrador de disparos do nó  $s_1$ .

8. Incremente a idade das conexões que chegam ao neurônio  $s_1$

$$age_{(s_1,i)} = age_{(s_1,i)} + 1 \quad (4.31)$$

9. Reduza o registrador de vitórias do neurônio  $s_1$  de acordo com

$$h_{s_1}(t) = h_0 - \frac{S(t)}{\alpha_b} (1 - e^{(-\alpha_b t / \tau_b)}) \quad (4.32)$$

e de seus vizinhos ( $i$ )

$$h_i(t) = h_0 - \frac{S(t)}{\alpha_n} (1 - e^{(-\alpha_n t / \tau_n)}) \quad (4.33)$$

onde  $h_i(t)$  é o valor do registrador de disparos do neurônio  $i$ ,  $h_0$  é o valor inicial para o registrador de vitórias, e  $S(t)$  é a força do estímulo, normalmente 1. As constantes  $\alpha_n$ ,  $\alpha_b$ ,  $\tau_n$ , e  $\tau_b$  controlam o comportamento da curva. O registrador de disparos do vencedor reduz mais rapidamente que dos seus vizinhos. A equação 4.32 é a solução da seguinte equação diferencial

$$\tau_b \frac{dh_{s_1}(t)}{dt} = \alpha_b [h_0 - h_{s_1}(t)] - S(t), \quad (4.34)$$

que é o modelo de redução da eficácia da sinapse com o passar do tempo.

10. Verifique se existe qualquer neurônio ou conexão a ser deletada, isto é, se existe qualquer neurônio que não possui mais nenhum vizinho, ou conexão que é mais velha que o maior valor permitido, então delete.
11. Se existirem mais entradas disponíveis, então retorne ao passo 1, caso nenhum critério de parada tenha sido alcançado.

## 4.6 Considerações

Este capítulo apresentou os mapas auto-organizáveis mais importantes para entender o STRAGEN. As características destes mapas mais marcantes e presentes no STRAGEN são: a estrutura topológica variável, como no GCS, GNG e GWR; o crescimento de acordo com a resposta da rede a um dado estímulo, caso a rede não responda bem a este estímulo então a rede deverá aprender este estímulo, semelhante ao comportamento do GWR.

Os assuntos mais importantes para a compreensão da abordagem proposta no próximo Capítulo, o 5, estão presentes nos Capítulos 3 e 4.

# 5

## Proposição do Modelo: STRAGIC

A proposta de solução para o problema tratado no Capítulo 2 consiste em construir trajetórias de estados contendo informações úteis para o controle de locomoção de um robô com membros. Estas trajetórias são formadas a partir de dados capturados durante a locomoção de um robô simulado semelhante ao robô que se deseja controlar. Estes dados podem ser ângulos das articulações, sinais de saída dos osciladores aplicados às articulações, ou as posições das articulações no espaço Euclidiano 3D. Embora o texto esteja focando no sinal de saída do CPG por ser uma informação comumente utilizada para controlar a locomoção de um robô com membros.

A proposta de solução é denominada STRAGIC (Gerador de Trajetórias de Estados com Interconexões). O STRAGIC é um modelo projetado a partir do STRAGEN (*State Trajectories Generator* – Gerador de Trajetórias de Estados) para desempenhar um comportamento semelhante a um CPG artificial. A estrutura interna da rede gerada pelo STRAGIC é diferente dos modelos de CPGs artificiais vistos no Capítulo 3. No entanto, o STRAGIC é capaz de gerar trajetórias de estados que descrevem o mesmo sinal produzido por osciladores que compõem um CPG. O STRAGEN proposto por Benante e Araujo (2007) é um modelo de Rede Neural Artificial de topologia variável capaz de gerar trajetórias de estados a partir do mapeamento do espaço de estados de um sistema.

Um trajetória de estados pode ser vista como uma sequência de estados que parte de um determinado estado inicial para um estado final desejado. A trajetória de estados que descreve a locomoção de um robô contém informações que determinam a postura do robô durante um passo. Um estado pode conter uma descrição da postura do robô ou um conjunto de níveis dos sinais gerados pelos CPG que definem a postura. A postura é o conjunto das posições angulares de todas as articulações dos membros do robô em um instante de tempo. As informações contidas nos estados são coletadas

---

em um intervalo de tempo regular durante um passo do robô. Quanto menor este intervalo de tempo, mais estados estarão contidos na trajetória de estados que descreve o passo do robô.

A locomoção do robô é realizada com a execução constante dos passos do robô. Uma trajetória que descreve um passo do robô é uma trajetória fechada, já que o início do passo coincide com o final. Assim, o robô se locomoverá ao executar uma trajetória fechada sucessivas vezes.

Algumas características ou propriedades do funcionamento dos modelos de CPG biologicamente inspirados são incorporadas no modelo proposto:

- O comportamento da saída dos CPGs são sinais oscilatórios que podem ser visualizados em cada instante de tempo como um estado. Deste ponto de vista, o STRAGIC também gera sinais oscilatórios contidos em estados. Embora a estrutura interna dos modelos seja completamente diferente.
- Um estímulo elétrico simples vindo do tronco cerebral é capaz de controlar os padrões de saída do CPG e consequentemente a velocidade e o modo de locomoção ([Ijspeert, 2008](#)). O STRAGIC pode receber comandos para aumentar ou diminuir a velocidade sendo que cada velocidade resultante está associada a um modo de locomoção.
- Um CPG é composto de osciladores neurais conectados aos neurônios motores. Estes neurônios motores estimulam os músculos para movimentar os membros ([Ijspeert, 2001](#)). O STRAGIC não reproduz a estrutura interna de um CPG, mas mantém um comportamento semelhante em relação a sua saída. Ele gera os movimentos rítmicos nos membros através de trajetórias cíclicas de estados.
- O sincronismo entre osciladores de um CPG determina o modo de locomoção. O STRAGIC captura um conjunto de amostras de posturas e constrói uma trajetória de estados com a mesma sequência dos estados originais gerados durante a locomoção, logo o STRAGIC mantém o mesmo sincronismo presente nas amostras de entrada.

Além dessas características, o STRAGIC é capaz de construir trajetórias até mesmo quando as amostras são originadas de uma fonte ruidosa. A construção dos movimentos pode ser realizada através do balbuciamento<sup>1</sup> motor ([Benante e Araujo, 2007](#)), um procedimento que possui plausibilidade no modo que os bebês aprendem seus

---

<sup>1</sup>Balbuciar: falar imperfeitamente, como as crianças, ou hesitando.

primeiros movimentos. Os nodos do STRAGIC podem tratar diferentes tipos de informação, pois a dinâmica da rede pode ser modificada através da escolha do critério de vizinhança. O STRAGIC tem capacidade de lidar com dados que caracterizam a locomoção (ex. postura) não importando a abordagem utilizada para gerar estes dados.

Uma vantagem do STRAGIC em relação aos modelos tratados no Capítulo 3 é que o STRAGIC não precisa preocupar-se com a defasagem entre membros, já que esta informação está embutida indiretamente nas informações dos estados.

## 5.1 A Estrutura do STRAGIC

O STRAGIC está dividido em dois módulos, o Módulo de Gerenciamento de Locomoção (MGL) e o Módulo de Controle de Locomoção (MCL). O MCL constrói trajetórias de estados que determinam os modos de locomoção do robô, já o MGL constrói as interligações entre trajetórias. O MCL possui um Componente Construtor de Mapa Topológico (CCMT), um Montador de Trajetórias e um Identificador de sub-redes. O MGL possui um Classificador de Sub-Redes e um Gerador de Inter-Conexões, ver Figura 5.1.

Para treinar o STRAGIC o primeiro passo é fornecer uma base de dados contendo estados originados de modos de locomoção de um robô. Esta base de dados é passada ao (MCL) que constrói sub-redes associadas a um modo de locomoção. O CCMT constrói uma rede contendo sub-redes, o Montador de Trajetórias cria trajetórias cíclicas a partir de uma sub-rede e o Identificador de Sub-Redes a partir dos estados de controle determina quais neurônios pertencem a uma sub-rede.

Depois de identificadas as sub-redes são passadas para o MGL, o Classificador de Sub-Redes do MGL identifica a velocidade associada a cada sub-rede ordena. O Gerador de Inter-Conexões constrói interligações entre redes que representam velocidades próximas. Com a realização das interligações a rede resultante é capaz de fazer a transição entre sub-redes ou permanecer em uma sub-rede de acordo com o comando recebido.

### 5.1.1 Módulo de Controle de Locomoção

No Módulo de Controle de Locomoção (MCL), cada neurônio representa um estado do robô que na etapa de aprendizagem as amostras (estados) são apresentadas a rede neural aleatoriamente de acordo com a distribuição uniforme.

O MCL possui um Componente Construtor de Mapa Topológico (CCMT) imple-

## 5.1. A ESTRUTURA DO STRAGIC

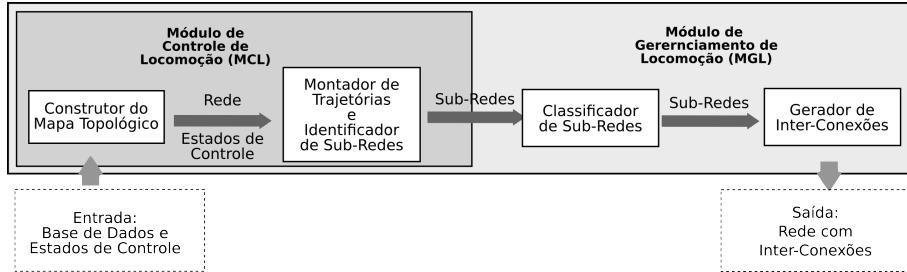


Figura 5.1: Treinamento do STRAGIC.

mentado com o STRAGEN. No decorrer da fase de aprendizagem o Componente Construtor de Mapa Topológico recebe um conjunto de estados aleatórios relacionados de maneira a representarem uma trajetória de estados que descreve a locomoção do robô. Considerando que estados de diferentes trajetórias sejam apresentados, o Construtor de Mapa Topológico é capaz de construir cada trajetória presente nestes estados. Considerando que uma trajetória de estado descreve um modo de locomoção, o CCMT a partir destes estados aprende diferentes modos de locomoção. Depois da trajetória de estados construída pelo CCMT, as informações sobre as posturas contidas nestes estados são utilizadas para determinar as posturas do robô durante a locomoção.

O STRAGEN original foi projetado para gerar trajetórias de estados para representar sistemas que tenham suas configurações determinadas por estados possíveis, e um conjunto válido de transições entre estados. Um dos problemas resolvidos pelo STRAGEN é o controle de manipuladores robóticos interpretado como um problema de gerar trajetórias de estados. Redes neurais com topologia variável como a GCS, GNG e GWR podem ser utilizadas para aprender um conjunto de dados de exemplos para representar o espaço de estados do problema. No entanto, estas redes modificam as informações contidas nos estados, em relação aos dados originais, devido a movimentação dos neurônios (Benante e Araujo, 2007). Neurônios com informações diferentes das amostras originais resultam em trajetórias também diferentes das originais.

Para a construção do mapa topológico, considere  $C$  o conjunto de conexões entre estes nodos,  $\xi$  uma amostra de entrada,  $w_i$  o vetor de pesos associado ao nodo  $i$ . Inicialize o conjunto  $A$  com dois nodos  $n_1$  e  $n_2$  posicionados em  $w_{n_1}$  e  $w_{n_2}$ , em  $\mathbb{R}^D$ , representando dois padrões aleatórios do conjunto de dados, sendo  $D$  a dimensão dos dados de entrada, assim  $A = \{n_1, n_2\}$ . Inicialize o conjunto de conexões vazio com uma conexão entre  $n_1$  e  $n_2$ ,  $C = \{c_{n_1, n_1}\}$ .

Cada vetor de pesos  $w_i$  pode conter informações heterogêneas, divididas em  $m$  grupos com elementos pertencentes a um mesmo domínio. Dois destes  $m$  grupos têm maior importância no algoritmo, são eles o grupo  $w_a$  usado para calcular a atividade

do neurônio e o grupo  $\mathbf{w}_n$  usado para criar a vizinhança da rede. O grupo  $\mathbf{w}_a$  pode ser dividido em outro  $l$  subgrupos, desta forma o limiar de atividade pode conter  $l$  elementos,  $\mathbf{a} = [a_1 a_2 \dots a_l]^T$ . O vetor de pesos associados ao nodo  $i$  é definido a seguir:

$$\mathbf{w}_i = [v_{1_i} v_{2_i} \dots v_{D_i}]^T = [\mathbf{w}_{1_i} \dots \mathbf{w}_{a_i} \dots \mathbf{w}_{n_i} \dots \mathbf{w}_{m_i}]^T.$$

O algoritmo de treinamento do CCMT (STRAGEN) é apresentado abaixo ([Benante, 2008](#)):

1. Apresente a rede uma amostra  $\xi_j = [\xi_{1_j} \dots \xi_{a_j} \dots \xi_{n_j} \dots \xi_{m_j}]^T$ . As amostras são obtidas da base de dados de maneira aleatórias e sem informar a sua posição cronológica;
2. Para cada vetor de peso  $\mathbf{w}_i$  na rede, calcule a distância Euclidiana para o estímulo de entrada  $\xi_j$ , empregando o grupo que define a vizinhança. Determine o nodo mais próximo,  $s_1$ , e o segundo mais próximo,  $s_2$ , da amostra de entrada:

$$\|\xi_n - \mathbf{w}_{n_{s_1}}\| \leq \|\xi_n - \mathbf{w}_{n_i}\|, \forall i \in A \quad (5.1)$$

$$\|\xi_n - \mathbf{w}_{n_{s_2}}\| \leq \|\xi_n - \mathbf{w}_{n_i}\|, \forall i \in A - \{s_1\} \quad (5.2)$$

3. Atualize o número de vitórias do vencedor,  $\gamma_{s_1} = \gamma_{s_1} + 1$ ;
4. Insira uma nova conexão entre  $s_1$  e  $s_2$ , se ainda não existir,  $C = C \cup (s_1, s_2)$ ;
5. Calcule a atividade do estímulo de entrada  $\xi_j$  em relação ao nodo vencedor ( $s_1$ ) para cada subgrupo  $k$  do grupo de atividade considerando:

$$a_{s_1, k} = \exp(-\|\xi_{a_j, k} - \mathbf{w}_{s_1, k}\|), 1 \leq k \leq l \quad (5.3)$$

onde  $l$  é o total de subgrupos de atividade,  $\mathbf{w}_{a_i} = [\mathbf{w}_{a_i, 1} \mathbf{w}_{a_i, 2} \dots \mathbf{w}_{a_i, l}]^T$  e  $\xi_j = [\xi_{j, 1} \xi_{j, 2} \dots \xi_{j, l}]^T$ .

6. Se a atividade de qualquer subgrupo de  $\xi_n$  for menor que um limiar estabelecido para este subgrupo, então adicione um novo nodo na localização descrita pelo exemplo de entrada:
  - (a) Adicione o novo nodo  $r$  ao conjunto  $A$ :  $A = A \cup \{r\}$ ;
  - (b) Crie um novo vetor de pesos para o nodo  $r$ , ou seja,  $\mathbf{w}_r = \xi_j$ ;
  - (c) Remova a conexão  $(s_1, s_2)$ ,  $C = C - \{(s_1, s_2)\}$ ;

- (d) Calcule as distâncias,  $D = \{\text{Dst}(r, s_1), \text{Dst}(r, s_2), \text{Dst}(s_1, s_2)\}$ , onde a distância entre dois nodos  $n_a$  e  $n_b$  é dado por:

$$\text{Dst}(n_a, n_b) = \|\mathbf{w}_{n_a} - \mathbf{w}_{n_b}\|,$$

- (e) Selecione as duas menores distâncias  $\text{Dst}_1$  e  $\text{Dst}_2$ :

$$\text{Dst}_1 = \arg \min(D), \quad \text{Dst}_2 = \arg \min(D - \{\text{Dst}_1\}).$$

- (f) Insira duas novas conexões, uma entre os nodos de  $\text{Dst}_1$  e outra entre os nodos de  $\text{Dst}_2$ :

$$C = C \cup \{\arg(\text{Dst}_1)\}$$

$$C = C \cup \{\arg(\text{Dst}_2)\}$$

7. Se um novo nodo não for inserido no passo anterior (6), atualize o vetor de pesos do nodo vencedor  $s_1$ :

$$\Delta \mathbf{w}_{s_1} = \rho \times (\boldsymbol{\xi}_j - \mathbf{w}_{s_1}), \text{ onde : } \rho = \begin{cases} \epsilon_b \times \alpha_f^{(\gamma_{s_1}/\gamma_{max})} & \gamma_{s_1} \leq \gamma_{max} \\ \epsilon_b \times \alpha_f & \gamma_{s_1} > \gamma_{max} \end{cases}$$

$0 < \epsilon_b < 1$  é a taxa de aprendizagem;  $\alpha_f \approx 0$  é a taxa de aprendizagem final;  $\gamma_{s_1}$  é o contador de disparos do nodo vencedor;  $\gamma_{max}$  é o número máximo de disparos, sugerido como  $\gamma_{max} = 2 \cdot I_{max}/L$ ,  $I_{max}$  é o número máximo de iterações e  $L$  a quantidade de amostras na base de dados.

8. Calcule o tamanho médio ( $\mu_{s_1}$ ) e o desvio padrão ( $\sigma_{s_1}$ ) de todas as conexões emanando do nodo vencedor  $s_1$ :

$$\mu_{s_1} = \frac{\sum_{i=1}^{|N_{s_1}|} \text{Dst}(s_1, n_i)}{|N_{s_1}|};$$

$$\sigma_{s_1} = \sqrt{\frac{\sum_{i=1}^{|N_{s_1}|} (\text{Dst}(s_1, n_i) - \mu_{s_1})^2}{|N_{s_1}| - 1}},$$

onde  $|N_{s_1}|$  é o número de vizinhos,  $N_{s_1}$  é o conjunto de vizinhos de  $s_1$  e  $n_i$  é um nodo vizinho de  $s_1$ , i.e.,  $\forall n_i \in N_{s_1}$ .

9. Se  $|N_{s_1}| > 2$ , remova todas as conexões ( $c_{s_1, n_i}$ ) do conjunto C para as quais:

$$\text{Dst}(s_1, n_i) > \mu_{s_1} + fp * \sigma_{s_1}, \forall n_i \in N_{s_1},$$

onde  $fp$  é o fator de poda da rede, cujo valor recomendado é 0,8 de acordo com os experimentos do Capítulo 6.

10. Remova todos os nodos sem vizinhos,  $|N_{n_i}| = 0, \forall n_i \in A$ .
11. Repita todos os passos a partir do passo (1) até que um número máximo de iterações  $t_{\max}$ , ou algum outro critério de parada tenha sido atingido.

A trajetória entre dois estados é gerada de acordo com uma função para propagação de energia (Equação 5.4). Para utilizá-la, assinala-se um nodo alvo e um nodo inicial que representa o estado de partida no CCMT. O nodo alvo tem energia de valor 1,0 e todos os demais são inicializados com energia zero. A energia iterativamente flui através da rede, de vizinho em vizinho até atingir o ponto inicial. O **Algoritmo de Geração de Trajetórias** entre dois estados pré-determinados é o seguinte:

1. Inicialize a função de difusão  $f_{n_i}(0) = 0, \forall n_i \in A / n_i \neq n_{\text{targ}}$  e  $f_{n_{\text{targ}}}(0) = 1,0$ .
2. Repita para todo  $n_i \in A$  até  $f_{n_{\text{init}}}(k) \neq 0$

$$f_{n_i}(k+1) = \begin{cases} 1 & \forall k, \text{if } n_i = n_{\text{targ}} \\ \lambda \sum_{n_j \in N_{n_i}} f_{n_j}(k) & \text{if } n_i \neq n_{\text{targ}} \end{cases} \quad (5.4)$$

onde  $N_{n_i}$  é o conjunto de todos os nodos que são vizinhos de  $n_i$ ,  $|N_{n_i}|$  é sua cardinalidade, e  $\lambda = 1(|N_{n_i}| + 1)$ . O valor de  $f_{n_{\text{targ}}}$  permanece em 1,0 em todas as iterações  $k$ .

O **Componente Montador de Trajetória** constrói uma trajetória cíclica a partir de um conjunto  $C = \{c_1, c_2, \dots, c_n\}$  de estados de controle. Estes estados, aos pares são enviados para o Algoritmo de Geração de Trajetórias do CCMT que devolve uma sub-trajetória para cada par. O Montador de Trajetória agrupa as sub-trajetórias fornecidas pelo ao algoritmo de geração de trajetórias formando uma **Trajetória Cíclica**. Os estados de controle  $C$ , são processados pelo Montador de Trajetória como mostra o algoritmo a seguir:

1. Repita para  $i$  de 1 até  $n - 1$ :
  - (a) Execute o algoritmo de geração de trajetória passando como o alvo  $c_i$  e como estado inicial  $c_{i+1}$

- (b) Salve a trajetória entre  $c_i$  e  $c_{i+1}$ ;

### 5.1.2 Módulo de Gerenciamento de Locomoção

O **Módulo de Gerenciamento de Locomoção** atua depois que o Módulo de Controle de Locomoção aprende diferentes trajetórias. O MGL tem o papel de escolher qual destas trajetórias deve ser executadas pelo robô. Para tanto, é preciso um mecanismo que permita a transição de um modo de locomoção para outro. Esta transição torna-se viável devido a existência de conexões entre neurônios pertencentes a duas sub-redes distintas. Com estas conexões inter-trajetórias, o MGL torna-se apto a gerenciar a transição de um modo de locomoção para outro.

Antes de gerar as conexões inter-trajetórias é preciso identificar qual o modo de locomoção contido em cada trajetória. Esta identificação pode ser feita através alguma métrica aplicada às informações contidas nos estados, como por exemplo a velocidade média de deslocamento angular dos membros ou a quantidade de estados da trajetória. Depois de identificadas, as trajetórias são organizadas de acordo com a métrica estabelecida em ordem crescente de velocidade. As interconexões são construídas tomando as trajetórias duas a duas, primeiro em ordem crescente de velocidade e em seguida na ordem decrescente.

É importante criar as interconexões tanto no sentido crescente de velocidade quanto no sentido decrescente, para garantir que cada nodo de cada trajetória possua um interconexão. Pois, se utilizar apenas uma direção, algum estado da trajetória de destino poderá ficar sem interconexão, já que mais de um estado da trajetória de origem pode escolher o mesmo estado da trajetória de destino. Além disso, a trajetória de destino pode conter menos estados que a trajetória alvo. Para que a transição ocorra de qualquer estado é necessário que cada neurônio da trajetória de partida possua uma conexão com o neurônio mais próximo da trajetória de destino.

Para evitar a reconstrução da mesma trajetória repetidas vezes foi elaborado uma etapa de memorização. Nesta etapa a trajetória é gravada juntamente com o seu sentido. É importante gravar o sentido da trajetória, pois se a trajetória for processada no sentido invertido o modo de locomoção resultante será diferente do original. Esta etapa de memorização consiste em utilizar três estados obtidos dos dados originais um do início do passo, outro do meio e o terceiro do final. Estes estados de controle são passados como entrada para o CCMT, e como saída o CCMT retorna a trajetória completa considerando a ordem contida nos estados de controle. Para cada novo conjunto de estados de controle o MGL armazena a trajetória gerada para evitar o

cálculo desnecessário de trajetórias que já foram geradas.

Algoritmo para construir as interconexões entre trajetórias:

1. Identifique trajetórias por velocidade, utilizando uma métrica, como:

(a) A velocidade angular dos membros:

$$\mathbf{v} = \sum_{t=1}^{n_p} \Delta\theta(t) / n_p, \quad (5.5)$$

onde  $\mathbf{v}$  é o vetor velocidade angular média dos membros,  $n_p$  é a quantidade de estados da trajetória,  $\theta$  é o vetor posição angular dos membros. O valor de  $n_p$  só pode ser determinado depois que o Montador de Trajetória construir a trajetória cíclica;

(b) Ou, a quantidade de estados na trajetórias. Pois levando em conta que estes estados são coletados em intervalos constantes de tempo, uma trajetória com velocidade mais elevada descreve um passo com menos estados.

2. Ordene cada trajetória  $T_i, i = 1, \dots, n_{traj}$  de acordo como a métrica escolhida em ordem crescente, resultando em  $T = \{T_1, T_2, \dots, T_{n_{traj}}\}$ , onde  $n_{traj}$  é a quantidade de trajetórias;

3. Construa o mapa inter-trajetórias:

(a) Para  $i = 1$  até  $n_{traj} - 1$ :

- Crie as interconexões entre  $T_i$  e  $T_{i+1}$ , para o aumento da velocidade ;

(b) Para  $i = n_{traj}$  até 1:

- Crie as interconexões entre  $T_i$  e  $T_{i-1}$ , para a redução da velocidade;

Algoritmo para criar as conexões entre duas trajetórias identificando a melhor conexão:

1. Informe as trajetórias  $T_a$  origem e  $T_b$  destino;

2. Para cada estado  $a_i \in T_a$ , onde  $T_a = \{a_1, \dots, a_l\}$  e  $l$  é número de estados em  $T_a$ , faça:

(a) Encontre o estado  $b_j$  mais próximo de  $a_i$ , onde  $b_j \in T_b = \{b_1, \dots, b_m\}$  e  $m$  é número de estados em  $T_b$ ;

(b) Crie a conexão entre  $a_i$  e  $b_j$ ;

- (c) Guarde os dois estados  $(a_i, b_j)$  que possuírem a menor distância em  $(a_{i_{min}}, b_{j_{min}})$ ;
3. Crie a conexão entre  $a_{i_{min}}$  e  $b_{j_{min}}$ , definindo esta conexão como a melhor transição entre as trajetórias  $T_a$  e  $T_b$ .

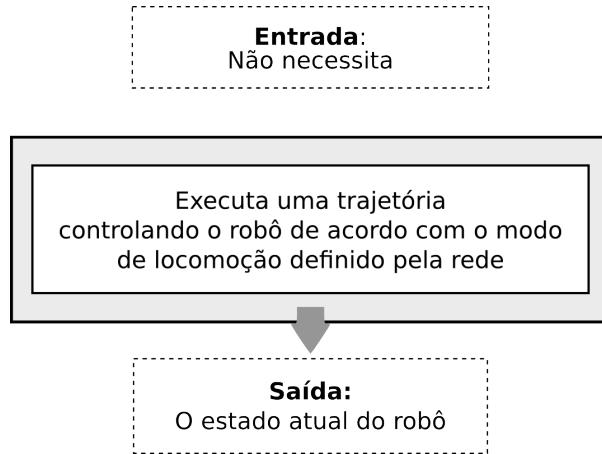


Figura 5.2: Diagrama de utilização do STRAGIC para um modo de locomoção.

Depois do treinamento, o STRAGIC pode ser utilizado de duas maneiras dependendo da quantidade de modos de locomoção aprendida. A forma mais simples de utilização é quando apenas um modo de locomoção for ensinado para o sistema, ver Figura 5.2. Quando mais de um modo de locomoção for aprendido o STRAGIC receberá comandos para alterar a velocidade de locomoção do robô 5.3. Os comandos de entrada do STRAGIC são: aumentar a velocidade, diminuir a velocidade, aumentar ou diminuir a velocidade da maneira mais suave possível. Como saída o STRAGIC fornece em cada intervalo de tempo o estado atual do robô.

## 5.2 Discussões

Esta seção faz uma breve discussão sobre o algoritmo do Construtor do Mapa Topológico focando em dois importantes parâmetros, o limiar de atividade e o fator de poda.

A inicialização do conjunto de conexões pode ser vazia ou com a conexão entre os dois nodos iniciais do conjunto  $A$ . Já que depois da primeira passagem pelo passo (6), durante a criação do primeiro nodo e terceiro inserido no conjunto  $A$ , restarão na rede as duas melhores conexões. Além disso, o passo (4) cria uma conexões entre os dois nodo mais próximos da amostra, caso ainda não exista.

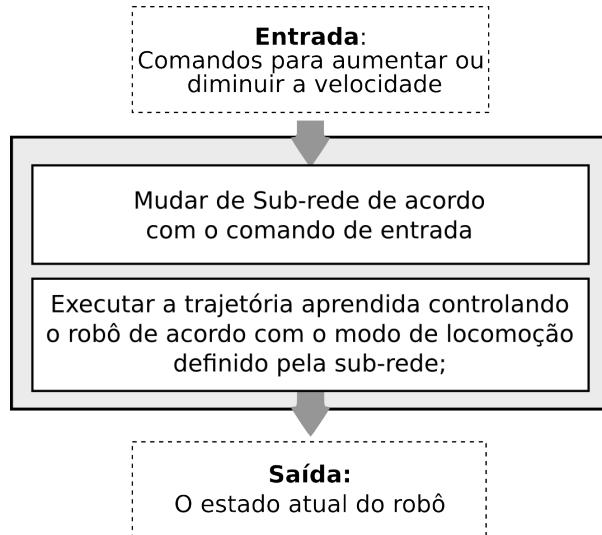


Figura 5.3: Diagrama de utilização do STRAGIC para mais de um modo de locomoção.

O passo (2) determina os dois nodos mais próximos da amostra de entrada. Estes nodos são utilizados em outros passos do CCMT, os passos (4) e (6) utilizam os dois, já os passos (5), (7), (8) e (9) utilizam apenas o nodo mais próximo.

O número de vitórias de um determinado nodo calculado no passo (3) é utilizado no passo (7), para modificar o tamanho do passo de adaptação, também chamada de taxa de aprendizagem. Quanto maior for a quantidade de vitórias menor será o passo de adaptação. O passo (4) garante a existência de uma conexão entre os dois nodos mais próximos do padrão de entrada.

O passo (5) calcula a atividade da amostra, valores menores de atividade significam que a amostra está mais distante do nodo vencedor e valores elevados a amostra está mais próxima do nodo vencedor. O passo (6) verifica se a atividade é menor que um limiar estabelecido, caso verdadeiro, adiciona a amostra à rede em forma de nodo. Então quanto maior o valor pré-estabelecido para o limiar de atividade mais nodos são adicionados a rede.

Quando o limiar de atividade é alto o suficiente para deixar a rede aprender todos as amostras da base de dados o passo (7) de adaptação do Construtor do Mapa Topológico fica inutilizado. Já que novos padrões apresentados a rede cairão na mesma localização de algum nodo da rede e não existirão mais amostra para inserir. Desta maneira a rede convergirá poucos passos depois da iteração de número cujo valor é o mesmo do tamanho da base de dados.

O termo fator de poda foi criado durante a execução dos experimentos desta dissertação, pois verificou-se que o valor do fator de poda exercia uma grande influência na

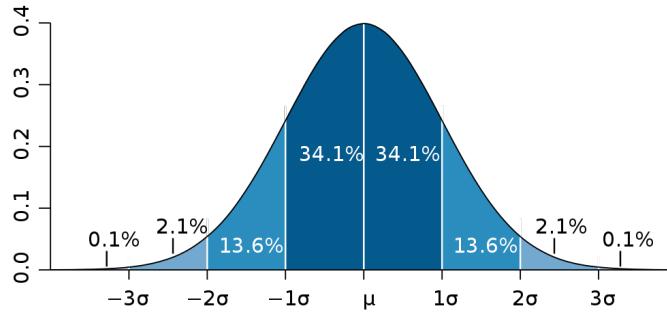


Figura 5.4: Diagrama desvio padrão, fonte Wikipédia.

formação da rede e consequentemente nos resultados gerados. Na versão original do STRAGEN, o fator de poda era um valor fixo em 1,5.

A distribuição de probabilidade do tamanho das conexões é desconhecida, mas para simplificar a análise e facilitar o entendimento do passo (7) a distribuição de probabilidade normal foi escolhida, Figura 5.4. Assim, o fator de poda esta diretamente relacionado ao desvio padrão. O passo (7), para o nodo  $s_1$ , exclui conexões maiores que a média mais o fator de poda vezes o desvio padrão das conexões ligadas a  $s_1$ . Considerando 1,0 como sendo o valor do fator de poda, as conexões têm apenas 15,8% de probabilidade de serem excluídas, por outro lado considerando  $-2,0$  para o fator de poda a probabilidade passa para 97,8%.

Uma boa opção para o critério de parada é o número de conexões não crescer por um determinado número de interações. A escolha por conexões ao invés de nodos é feita devido ao fato de que uma conexão pode ser criada tanto no passo (4) quanto no passo (6), mas um novo nodo só pode ser criado no passo (6).

# 6

## Experimentos

Um dos objetivos desta seção é verificar a capacidade do STRAGIC em aprender a controlar diferentes modos de locomoção independente da fonte de dados. Os dados podem ser originados de uma locomoção real ou artificial. Outro objetivo é mostrar a capacidade do STRAGIC em gerenciar a transição entre modos de locomoção para dados artificiais. Os objetivos específicos são: testar a capacidade do STRAGIC em lidar com dados ruidosos e fazer um estudo paramétrico do Componente Construtor de Mapa Topológico.

Os experimentos estão divididos em duas seções, uma para dados artificiais extraídos de uma locomoção artificial e outra com dados reais. Os dados artificiais foram gerados para um robô hexápode controlado com o algoritmo de [Arena et al. \(2004\)](#). Os dados reais foram extraídos de um vídeo de um cachorro andando numa esteira.

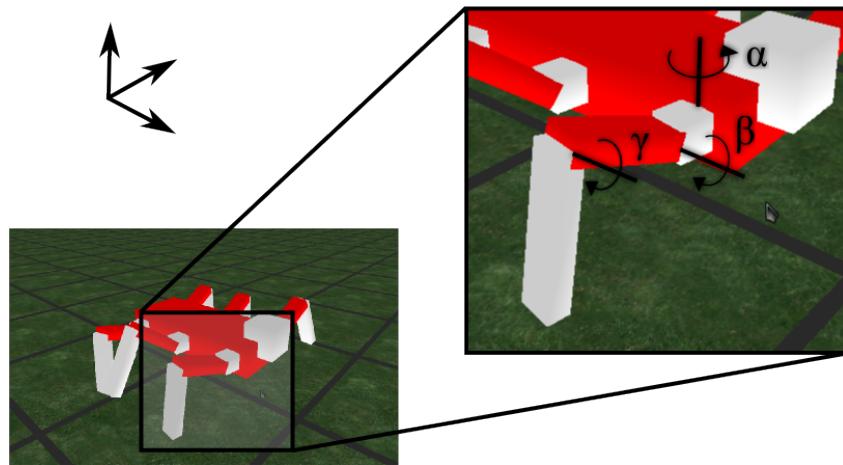


Figura 6.1: Os graus de liberdade dos membros do robô hexápode.

## 6.1 Dados Artificiais

Esta seção contém experimentos que tratam do controle de locomoção e da transição entre modos de locomoção de um robô hexápode. O Módulo de Controle de Locomoção (MCL) do STRAGIC constrói uma trajetória de estados a partir de dados artificiais para controlar a locomoção do robô, já a transição entre modos de locomoção é realizada pelo Módulo de Gerenciamento de Locomoção (MGL) do STRAGIC. Além disso, o experimento de controle de locomoção verifica a capacidade do STRAGIC em lidar com dados ruidosos. Antes de explicar o experimento de controle de locomoção, uma breve seção investiga o comportamento do Construtor de Mapa Topológico diante da variação do valor de dois parâmetros.

O simulador utilizado nos experimentos é o Gazebo, descrito em mais detalhes no Apêndice B. Gazebo é um simulador de alta fidelidade para ambientes dinâmicos ao ar livre. Cada objeto simulado possui massa, velocidade, atrito e outros atributos que tornam o comportamento mais realístico quando empurrados, puxados, em queda ou carregados. Os robôs são estruturas dinâmicas composta de corpos rígidos conectados através de articulações. O ambiente de simulação proporcionado pelo Gazebo pode conter paisagens, construções estruturadas e outros objetos criados pelo usuário.

O ambiente de simulação foi configurado da seguinte maneira: Ubuntu Linux 8.10, processador Pentium Dual Core de 1.6GHz, placa de vídeo Geforce 9600. O simulador Gazebo foi instalado a partir do código fonte obtido do repositório na versão 7551. As principais dependências do Gazebo são as bibliotecas Ogre (Motor Gráfico 3D de Código Aberto) e ODE (Motor de Dinâmica de Código Aberto), também instaladas a partir do código fonte.

A modelagem 3D do robô foi elaborada dentro de um arquivo XML que é carregado no momento que o simulador é executado. Este XML descreve todo o ambiente que envolve o robô inclusive o próprio robô. O robô é formado cubos com dimensões modificadas. Possui seis membros, cada membro com três partes associada a uma articulação, chamadas de  $\alpha$ ,  $\beta$  e  $\gamma$ , como visto na Figura 6.1. O robô possui um total de 18 articulações, mas 12 graus de liberdade controlados pelos sistema, já que as 6 articulações  $\gamma$  foram mantidas numa posição angular constante de -100 graus (-1,4 radiano). A Figura 6.2 mostra uma captura de tela do simulador no momento em que o robô hexápode está sendo controlado pelos osciladores de [Arena et al. \(2004\)](#) e se deslocando no modo de locomoção médio. A Figura 6.3 mostra os quadros gerados durante a simulação do modo de locomoção médio e escolhidos a cada 2 unidades de tempo de simulação. Os sinais gerados pelos osciladores CNN para as articulações  $\beta$

## 6.1. DADOS ARTIFICIAIS

---

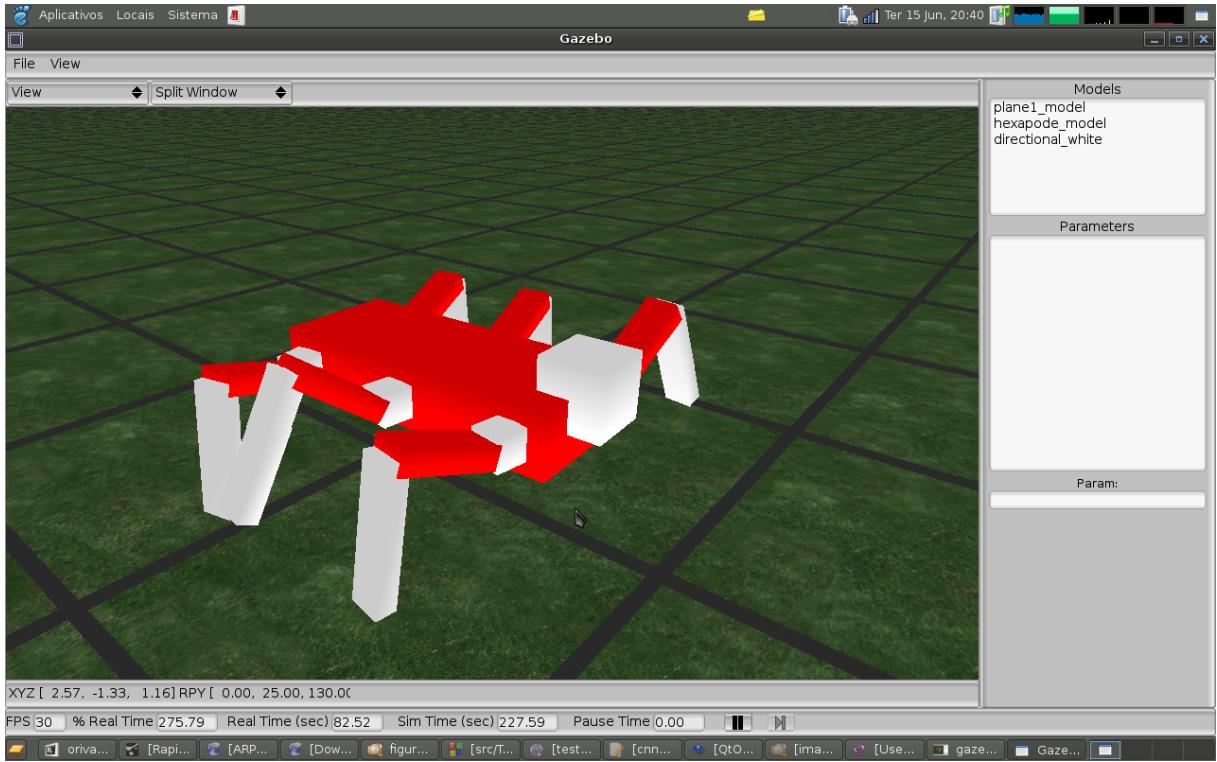


Figura 6.2: Captura de tela com a janela do simulador.

em cada um dos três modos de locomoção estão presentes nos gráficos das Figuras 6.4, 6.5 e 6.6.

Os osciladores de [Arena et al. \(2004\)](#) controlam as articulações livres do robô através da relação contida na Equação 6.1. O sinal de saída dos osciladores CNN controla as articulações  $\alpha$  e o estado interno dos osciladores CNN controla as articulações  $\beta$ :

$$\begin{cases} \alpha = 0,2 * y_2 \\ \beta = 0,8 * h(x_1) - 0,1 \\ \gamma = -1,4 \end{cases} \quad (6.1)$$

$x_1$  estado interno da camada 1 e  $y_2$  saída da camada 2, estas camadas fazem parte da célula da CNN.

$$h(x) = \begin{cases} 1, & \text{if } x \geq 1, \\ x, & \text{if } 0,5 < x < 1, \\ 0,5, & \text{if } x \leq 0,5. \end{cases}$$

As bases de dados utilizadas nos experimentos com o robô hexápode foram geradas a partir de estados capturados dos sinais gerados pelos osciladores que controlam cada articulação do robô. Estes sinais são o estado interno  $x_1$  e a saída  $y_2$  de cada oscilador



Figura 6.3: Quadros do modo de locomoção médio.

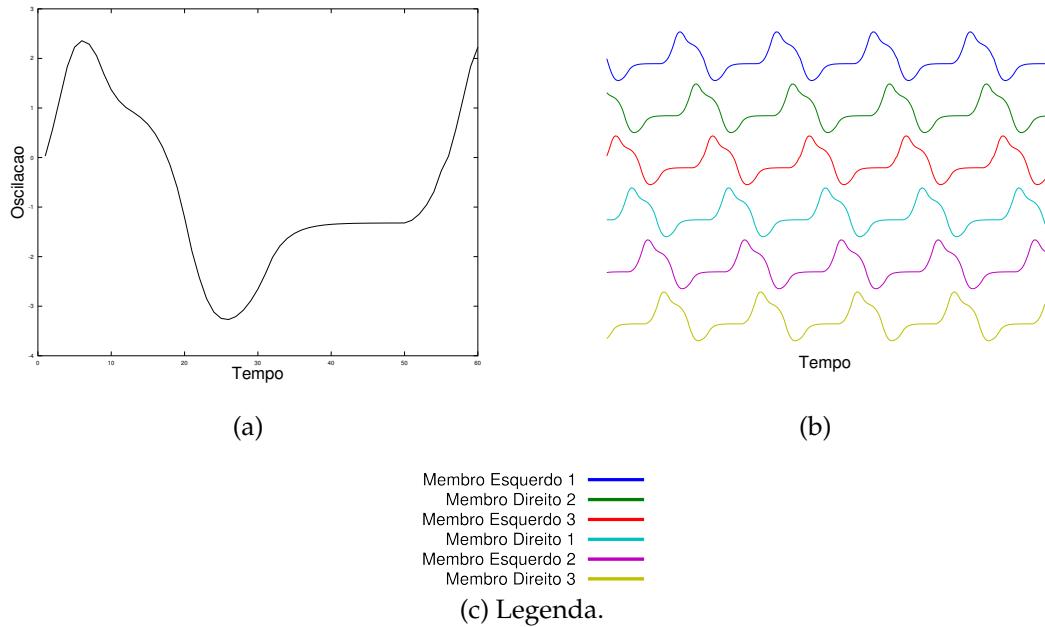


Figura 6.4: Gráficos dos sinais aplicados às articulações  $\beta$  obtidos de osciladores CNN: (a) de uma articulação; (b), de todas as articulações no modo de locomoção lento.

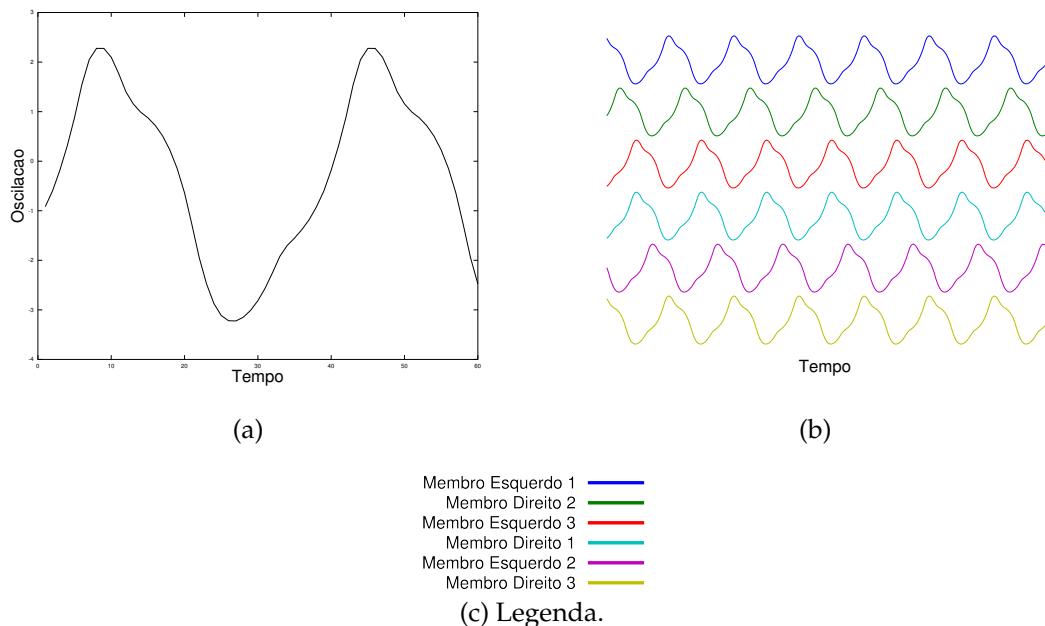


Figura 6.5: Gráficos dos sinais aplicados às articulações  $\beta$  obtidos de osciladores CNN: (a) de uma articulação; (b) de todas as articulações no modo de locomoção médio.

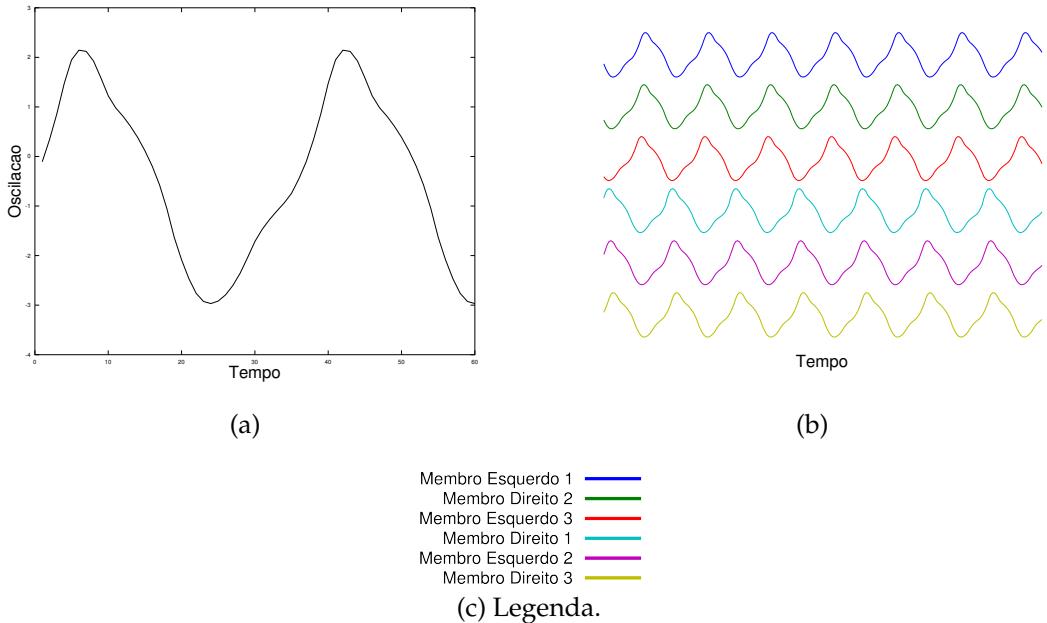


Figura 6.6: Gráficos dos sinais aplicados às articulações  $\beta$  obtidos de osciladores CNN: (a) de uma articulação; (b) de todas as articulações nos modos de locomoção rápido.

CNN, como para cada membro do robô hexápode existe um oscilador CNN, a dimensão dos dados é 12,  $\xi \in \mathbb{R}^{12}$ . Estes sinais foram coletados em um intervalo de tempo regular (330 milissegundos) durante execução de um passo do robô controlado pelo algoritmo de [Arena et al. \(2004\)](#). Os três modos de locomoção gerados foram o lento, médio e rápido, seus estados armazenados em suas respectivas bases de dados sem considerar a posição cronológica do estado durante a locomoção. Assim, cada amostra era passada para o STRAGIC aleatoriamente sem informar a sua posição temporal. A Figura 6.7 mostra três redes de tamanhos diferentes formadas pelo STRAGIC. Os nomes dos nodos foram atribuídos no momento de sua criação, logo o primeiro nó recebeu o nome de “1” o segundo de “2” e assim sucessivamente. Como a ordem temporal dos estados não foi apresentada durante a etapa de criação da rede, os nodos apresentam uma relação de vizinhança aleatória, considerando seus nomes. O algoritmo de [Arena et al. \(2004\)](#) foi implementado em C++ validado com os resultados presentes em [Arena et al. \(2002, 2004\)](#) e com o auxílio do simulador Gazebo desenvolvido por [Koenig e Howard \(2004\)](#). Os valores dos parâmetros de configuração dos osciladores CNN também estão presentes em [Arena et al. \(2004\)](#).

As configurações do STRAGIC comum a todos os experimentos desta Seção 6.1:

- O procedimento de balbuciamiento motor (*motor babbling*) não precisou ser uti-

lizado, já que as bases de dados são pequenas e a rede normalmente converge depois que todas as amostras são apresentadas;

- O critério de parada do treinamento foi a quantidade de conexões da rede não aumentar por 300 iterações.
- Taxa de aprendizagem inicial de 0,1 e final de 0,001;
- Número máximo de disparos para o nodo vencedor de 40;
- Grupo de vizinhança e de atividade como sendo todo o vetor de características;
- Nenhum subgrupo de atividade;
- Fator de poda em 0,8.
- Quantidade de estados na trajetória como métrica para ordenar as trajetórias por velocidade.

Para cada experimento cada configuração foi executada no mínimo 30 vezes.

O algoritmo DTW (*Dynamic Time Warping*) encontrado em [Senin \(2008\)](#), [Keogh e Pazzani \(2001\)](#) e [Myers et al. \(1980\)](#), utilizado para comparar as trajetórias geradas pelos STRAGIC com as trajetórias dos dados originais, fornece uma medida de distância entre duas sequências que variam no tempo. O DTW é uma algoritmo bastante popular e eficiente para calcular similaridade entre séries temporais minimizando o efeito do deslocamento e a distorção no tempo. O cálculo da similaridade é realizado através de operações sobre as séries temporais com o intuito de detectar padrões semelhantes em diferentes fases. Quanto menor o valor da comparação calculada pelo DTW mais semelhantes são as trajetórias. Este algoritmo modifica o eixo temporal de uma ou das duas trajetórias de modo a encontrar eficientemente um alinhamento entre as trajetórias.

### 6.1.1 Estudo Paramétrico do CCMT

Esta seção investiga dois parâmetros do Componente Construtor de Mapa Topológico (CCMT), o limiar de atividade e o fator de poda. O limiar de atividade influencia na quantidade de nodos presentes na rede produzida pelo CCMT e o fator de poda influencia na quantidade de conexões. Quanto maior o limiar de atividade maior será a quantidade de nodos presentes na rede. No entanto, a partir de um determinado valor do limiar de atividade, todas amostras da base de dados de treinamento são

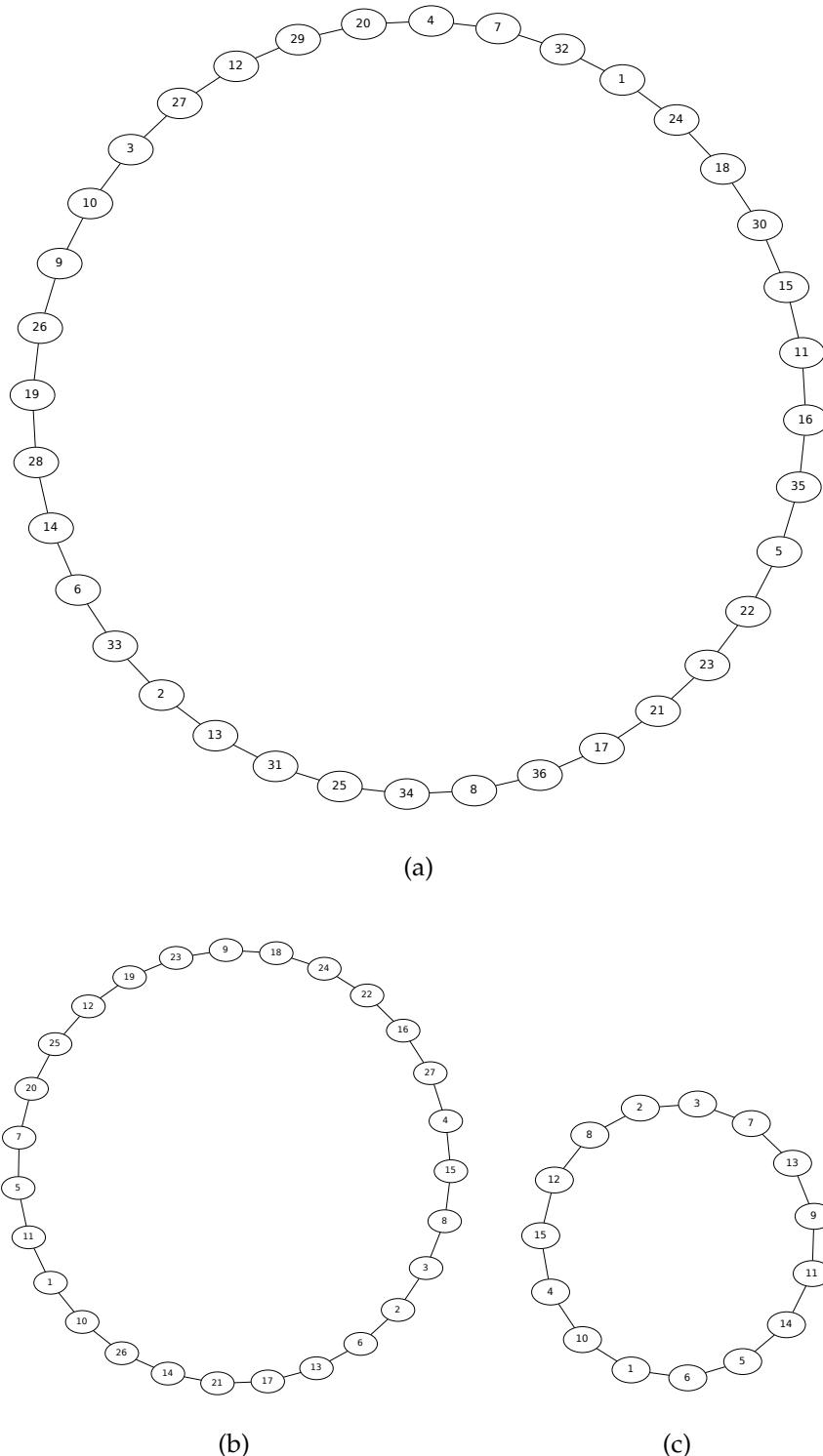


Figura 6.7: Três redes de tamanhos diferentes construídas pelo STRAGIC: (a) com 36 nodos, (b) com 27 e (c) com 15.

Tabela 6.1: Limiar de Atividade e Tamanho da Rede.

Config.	Fator de Poda	Limiar de Atividade	Média de Nodos	desvio
1	0,80	0,66	36,0	0,0
2	0,80	0,40	27,133	0,77608
3	0,80	0,30	15,100	0,88474

Tabela 6.2: Fator de Poda e Quantidade de Conexões.

Config.	Limiar de Atividade	Fator de Poda	Média de Conexões	desvio
1	0,66	2,5	41,100	1,3734
2	0,66	0,80	36,000	0,0

transformadas em nodos, neste caso a quantidade de nodos na rede não aumenta. A Figura 6.7 mostra três redes de tamanhos diferentes uma para cada configuração da Tabela 6.1.

O fator de poda está relacionado ao desvio padrão do conjunto de conexões do nodo a ser podado, sendo que o valor da conexão é a distância euclidiana entre os dois nodos unidos pela conexão. Duas situações para o fator de poda foram testadas: uma para o valor de 2,5, com este valor o CCMT falha ao criar redes capazes de gerar uma trajetória que descreva o modo de locomoção original; outra situação com o fator de poda em 0,8, permitiu ao CCMT criar redes com sucesso. A Figura 6.8 mostra um resultado obtido com a configuração 1 da Tabela 6.2 onde o CCMT falha ao construir a trajetória que descreve o modo de locomoção original. Com a configuração 2, o CCMT em todas as execuções construiu uma trajetória idêntica a original.

As configurações elaboradas neste experimento estão presentes nas Tabelas 6.1 e 6.2. Durante a análise do limiar de atividade, o fator de poda foi fixado em 0,8 e durante a análise do fator de poda o limiar de atividade permaneceu configurado em 0,66. A base de dados escolhida foi extraída do modo de locomoção rápido. Cada configuração foi executada 30 vezes obtendo assim a média e o desvio padrão para a quantidade de nodo e também para a quantidade de conexões.

A falha ilustrada na Figura 6.8 acontece por que o STRAGIC foi configurado com um fator de poda que não retira conexões que representam mal a relação de vizinhança dos estados. A rede contida na figura 6.8a gera trajetórias incompletas, Figura 6.8b. O gráfico da Figura 6.8c faz uma comparação entre o sinal oscilatório gerado por esta rede, linha contínua, e o sinal original, linha tracejada.

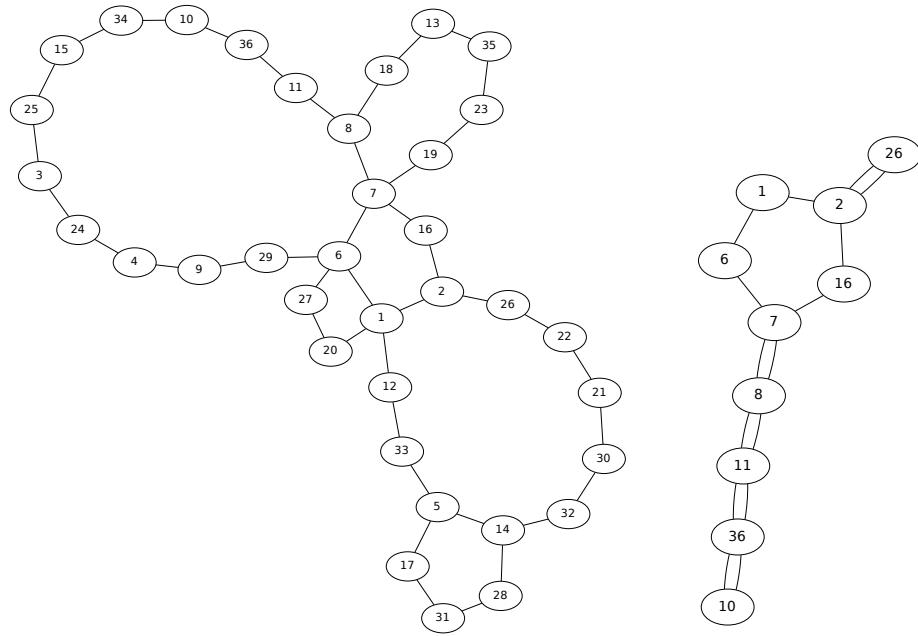


Figura 6.8: Situação em que o STRAGIC falhou ao tentar construir a trajetória original.

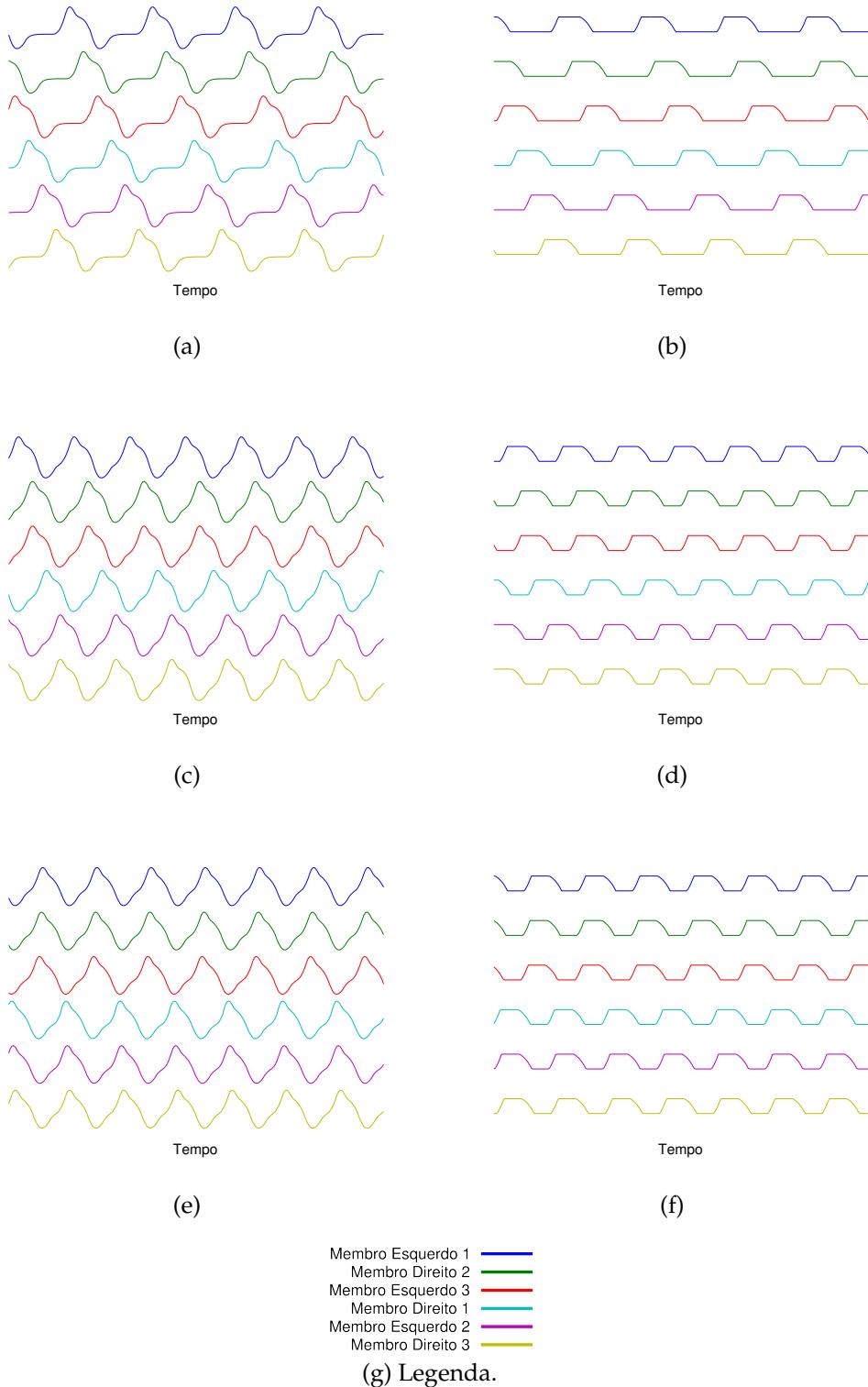


Figura 6.9: Trajetórias dos osciladores CNN construídas pelo STRAGIC com fator de poda 0,8 e limiar de atividade de 0,65.

Tabela 6.3: Distância entre as trajetórias geradas pelo CCMT e as trajetórias geradas pelos osciladores CNN.

Trajetória	DTW
Modo de locomoção lento	0,0
Modo de locomoção médio	0,0
Modo de locomoção rápido	0,0

### 6.1.2 Controle de Locomoção

Nesta seção os experimentos investigam a capacidade do STRAGIC em gerar uma trajetória de estados para controlar um determinado modo de locomoção. As bases de dados, como descritas anteriormente, são para os modos de locomoção lento, médio e rápido. A Tabela 6.3 apresenta as distâncias calculadas com o DTW entre cada base de dados e sua respectiva trajetória de estado gerada pelo STRAGIC, provando que estas trajetórias são idênticas.

A Figura 6.9 mostra os gráficos das trajetórias de estados construídas pelo CCMT para os modos de locomoção lento, médio e rápido. Os Gráficos 6.9a, 6.9c e 6.9e são idênticos aos os sinais dos osciladores CNN aplicados às articulações  $\alpha$  e os gráficos 6.9b, 6.9d e 6.9f aos sinais aplicados às articulações  $\beta$  para os modos de locomoção lento, médio e rápido respectivamente.

### 6.1.3 Três Modos de Locomoção numa Base de Dados

Nesta etapa do experimento os dados relativos aos três modos de locomoção foram armazenados em uma só base de dados de forma aleatória, sem informação das sequências dos estados em cada trajetória. O CCMT gerou cada uma das três trajetórias a partir dos estados armazenados. A Figura 6.10 ilustra a relação de vizinhança entre os nodos resultante ao final do treinamento do CCMT com sucesso. Já a Figura 6.11 mostra uma situação em que o CCMT falha ao construir a rede para os três modos de locomoção.

O limiar de atividade do CCMT foi configurado para 0,65, valor com o qual as redes geradas passaram a ter uma quantidade de nodos equivalente a quantidade de estados das trajetórias originais. O fator de poda foi ajustado para 0,8. Antes o valor padrão do fator de poda no CCMT era de 1,2. Com fator de poda em 1,2 o CCMT sempre falhava ao construir as redes, já que com este valor do fator de poda, conexões entre estados não vizinhos eram mantidas na etapa de poda do CCMT. Em todas as execuções, o CCMT com fator de poda em 0,8 construiu as três trajetórias completamente.

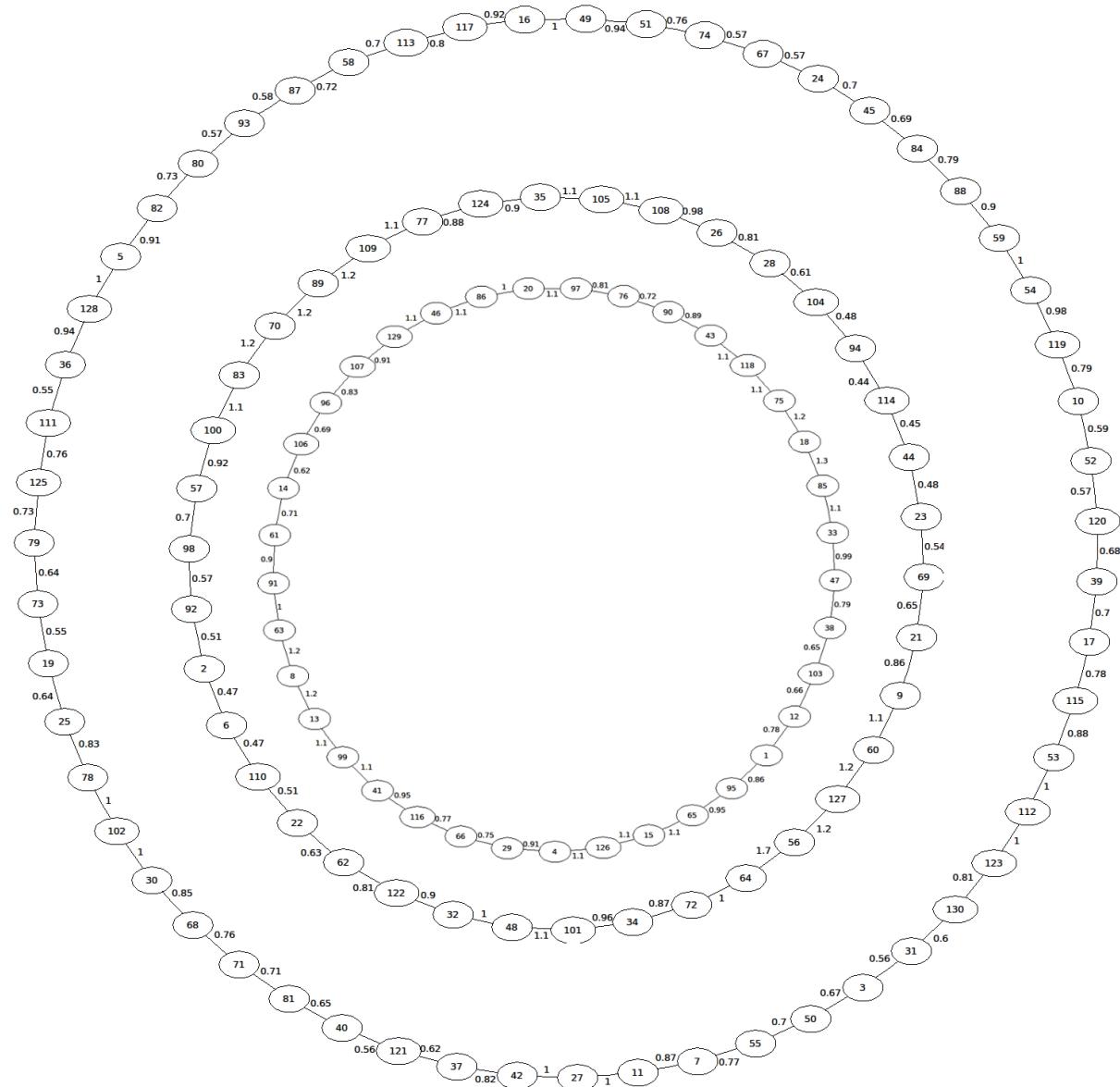


Figura 6.10: Cada nó no grafo representa um nodo do CCMT, logo um estado. A figura mostra os três modos de locomoção. O fator de poda foi configurado para 0,8 e o limiar de atividade para 0,65.

A Figura 6.11 deveria mostrar três redes cíclicas separadas, mas como o CCMT não removeu conexões que representam de maneira inadequada a relação entre os estados. Neste caso a rede contém estados conectados dois a dois que não representam a sequência de estados da trajetória original.

### 6.1.4 Dados Ruidosos

Esta etapa dos experimentos testa a capacidade do CCMT em lidar com dados ruidosos. A partir das três bases de dados originais, nove novas bases de dados foram criadas, assim para cada modo de locomoção três bases de dados foram elaboradas uma para cada nível de ruído. Os dados ruidosos foram criados a partir dos dados originais com auxílio da função gaussiana. Em cada dimensão dos dados originais era somado um número real calculado com a função gaussiana configurada com média igual a 0 ( $\mu = 0$ ) e desvio padrão ( $\sigma$ ) de 0,001, 0,01 e 0,1, Tabela 6.4. Cada configuração foi executada no mínimo 30 vezes e com os resultados foram gerados os gráficos saída (Figuras 6.12, 6.13, 6.14 e 6.15). O CCMT foi configurado com: parâmetro de poda para 0,8; três estados de controle, cada um escolhido a cada 1/3 do total de amostras de treinamento; para cada configuração, um limiar de atividade diferente conforme a Tabela 6.4.

Os resultados obtidos neste experimento estão expressos nos gráficos das Figuras 6.12, 6.13 e 6.14 para diferentes níveis de ruído e modos de locomoção. O gráfico 6.12a foi obtido com a configuração 1, o gráfico 6.12b com a configuração 2 e assim sucessivamente. O  $\sigma$  em 0,1 gera um ruído intenso o bastante para misturar duas amostras que antes eram representadas por nodos diferentes. Logo, o limiar de atividade precisou ser reduzido para que cada nodo representasse uma região maior do espaço. Consequentemente uma quantidade menor de nodos passou a representar uma trajetória. Valores mais elevados de ruído resultam em trajetórias mais distantes das originais, já que os nodos são criados a partir de amostras ruidosas e distantes das originais, Tabela 6.4. A Figura 6.15 contém gráficos do sinal do estado interno do oscilador CNN, o gráfico 6.15a compara a trajetória original do modo de locomoção médio com a trajetória gerada pelo CCMT a partir de dados ruidosos gerado com  $\sigma$  em 0,01, já o gráfico 6.15b compara os dados originais do modo de locomoção lento com a trajetória obtida pelo CCMT a partir de dados com ruidosos gerados com  $\sigma$  em 0,1.

Em cada configuração contida na Tabela 6.4 o limiar de atividade precisou ser ajustado. Já que valores mais elevados do limiar de atividade a rede aprendia uma quantidade maior de amostras ruidosas e baixos valores do limiar de atividade a rede aprendia uma quantidade menor dos estados originais.

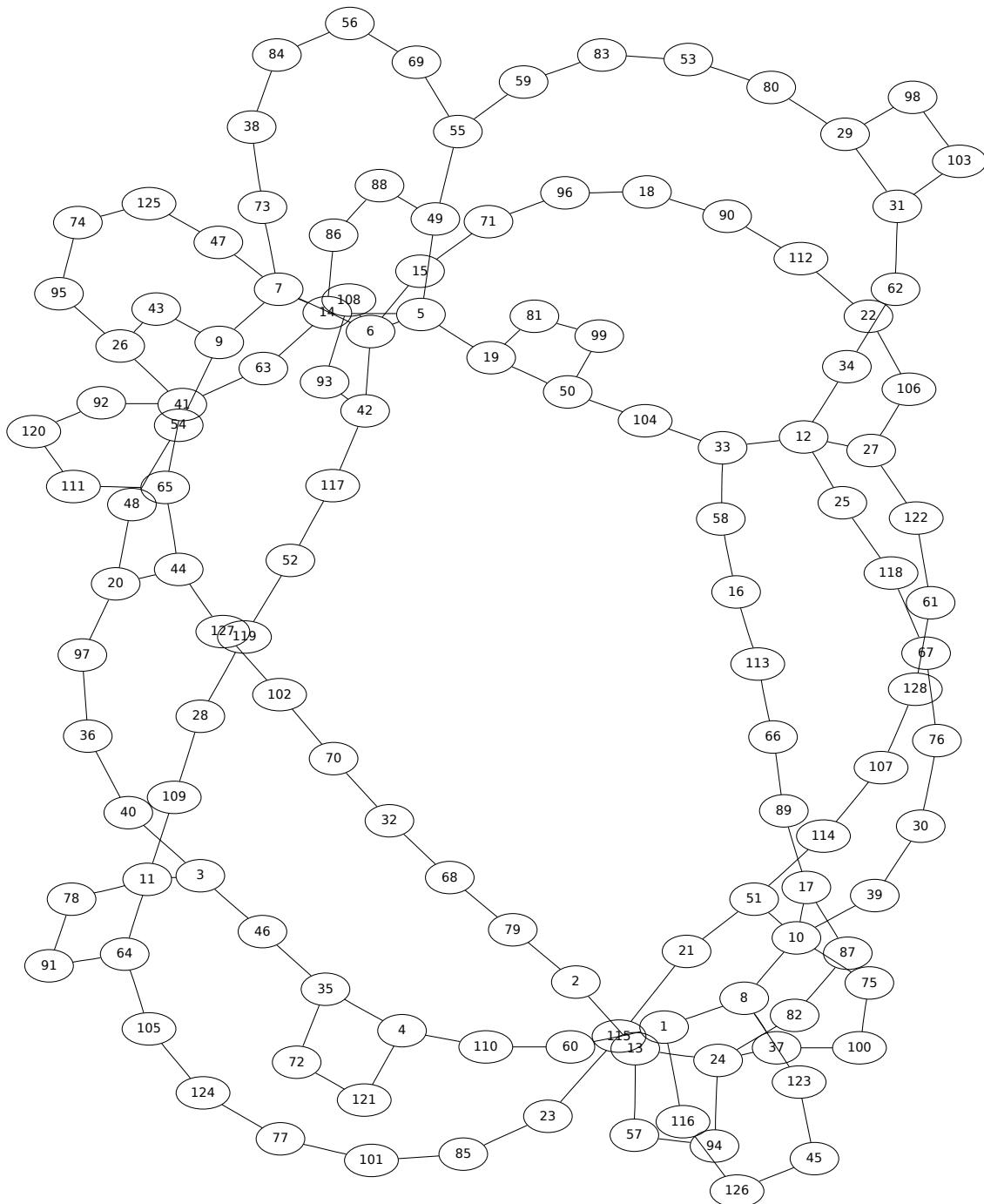


Figura 6.11: Exemplo de uma criação de rede mal sucedida a partir de uma base de dados com estados dos três modos de locomoção e com o fator de poda em 1,2 e limiar de atividade de 0,65.

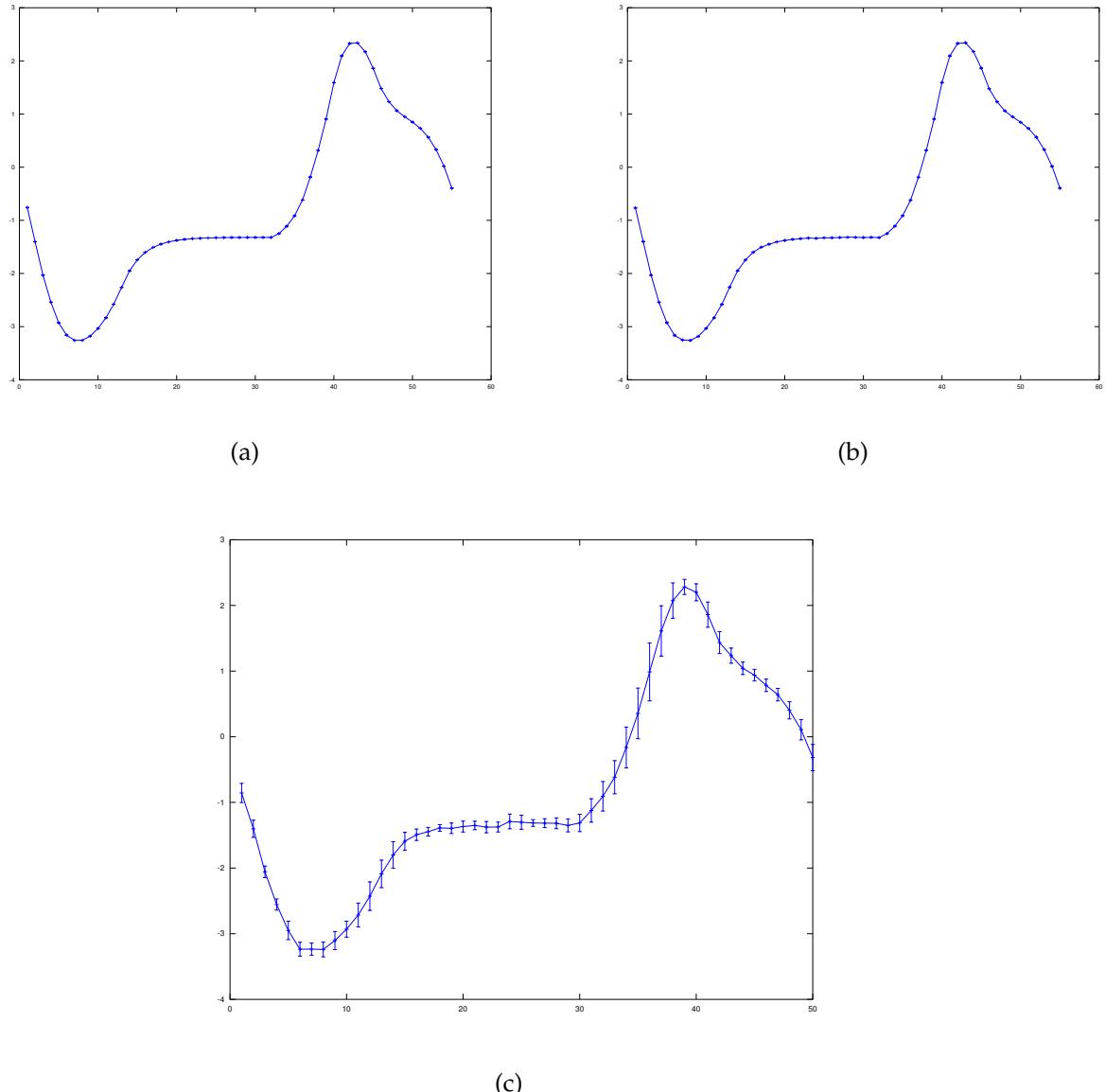


Figura 6.12: Os gráficos da variação do sinal aplicado as articulações *beta* no modo de locomoção lento em três níveis de ruído com fator de poda em 0,8: (a)  $\sigma = 0,001$  e limiar de atividade em 0,9; (b)  $\sigma = 0,01$  e limiar de atividade em 0,89; (c)  $\sigma = 0,1$  e limiar de atividade em 0,46.

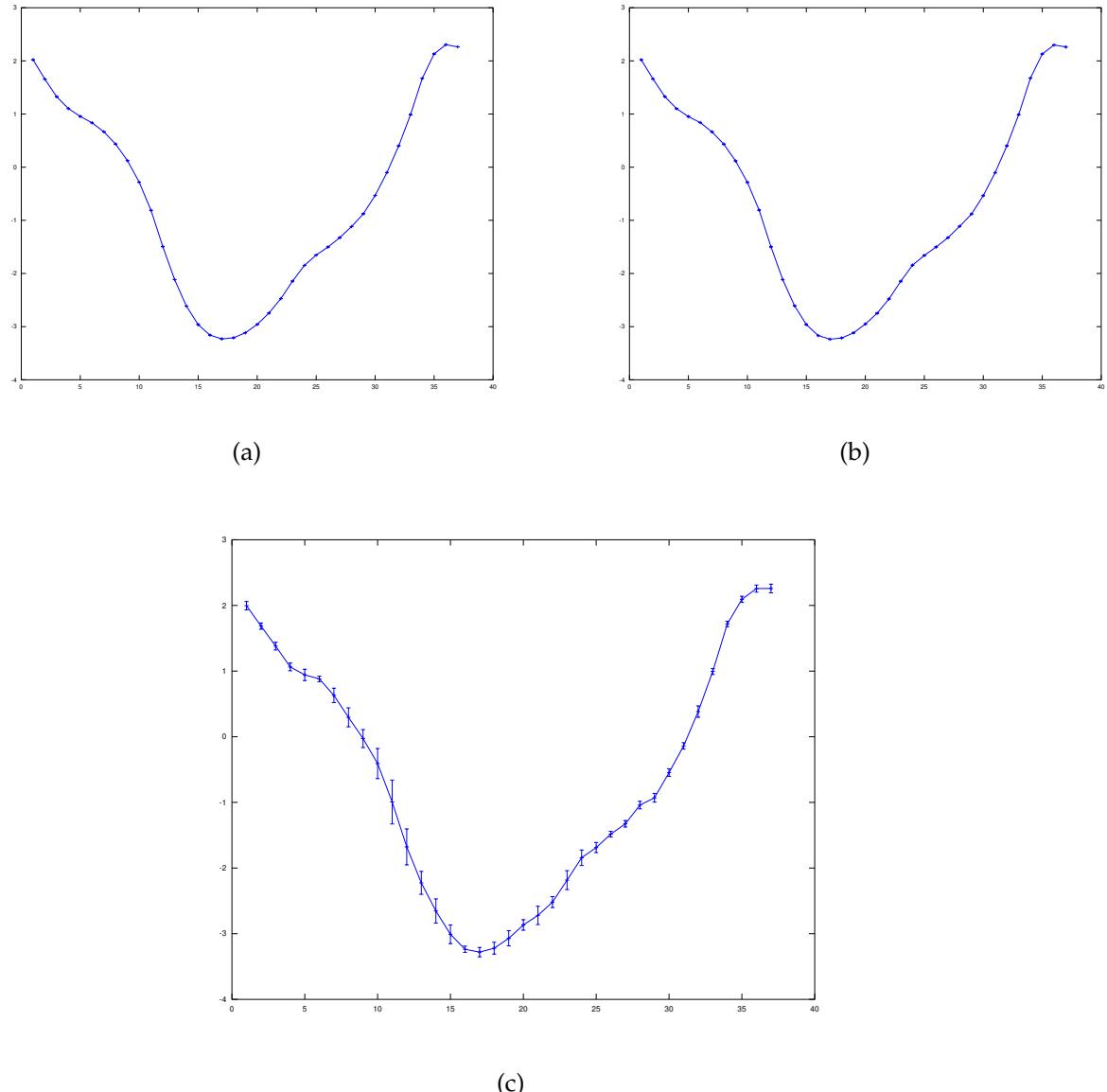


Figura 6.13: Os gráficos da variação do sinal aplicado as articulações *beta* no modo de locomoção médio em três níveis de ruído com fator de poda em 0,8: (a)  $\sigma = 0,001$  e limiar de atividade em 0,54; (b)  $\sigma = 0,01$  e limiar de atividade em 0,55; (c)  $\sigma = 0,1$  e limiar de atividade em 0,465.

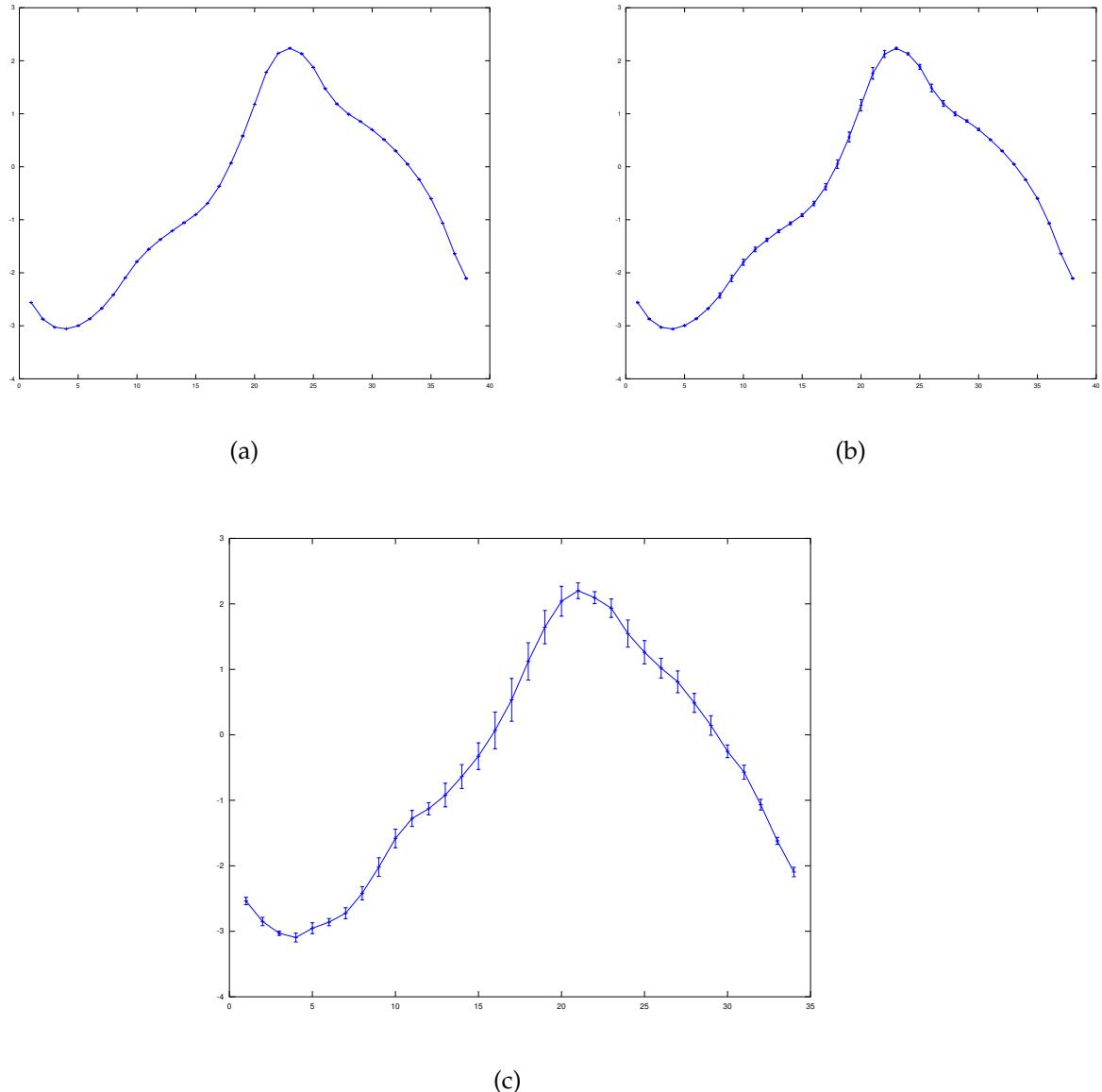


Figura 6.14: Os gráficos da variação do sinal aplicado as articulações *beta* no modo de locomoção rápido em três níveis de ruído com fator de poda em 0,8: (a)  $\sigma = 0,001$  e limiar de atividade em 0,7; (b)  $\sigma = 0,01$  e limiar de atividade em 0,66; (c)  $\sigma = 0,1$  e limiar de atividade em 0,48.

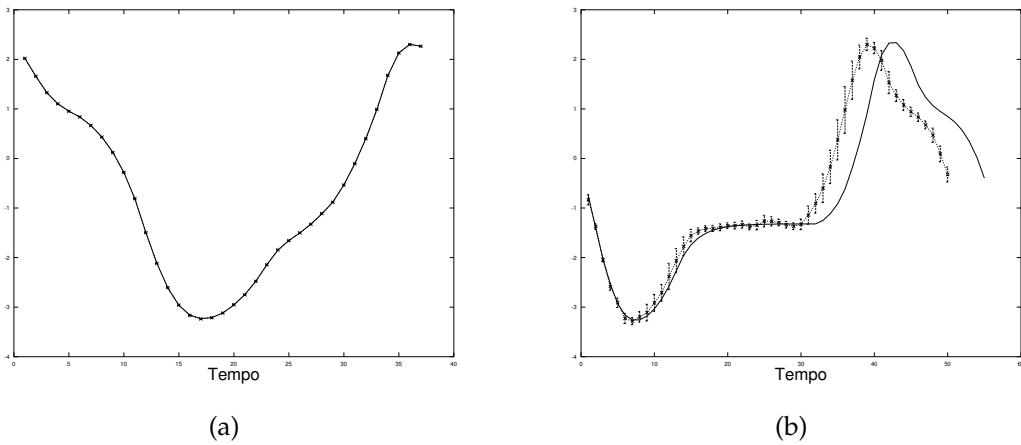


Figura 6.15: Resultados para os níveis mais altos de ruído: (a) o ruído criado com a função gaussiana com desvio padrão de 0,01 no modo de locomoção médio; (b) ruído criado com a função gaussiana com desvio padrão 0,1 no modo de locomoção lento.

Tabela 6.4: *Setup* experimental para os três modos de locomoção

Configuração	Locomoção	Ruído	Limiar de Atividade	DTW
1	Lento	0,001	0,900	$6,0103 \times 10^{-6}$
2	Lento	0,010	0,890	$5,5686 \times 10^{-4}$
3	Lento	0,100	0,460	$1,6560 \times 10^{-1}$
4	Médio	0,001	0,540	$7,0740 \times 10^{-6}$
5	Médio	0,010	0,550	$6,6158 \times 10^{-4}$
6	Médio	0,100	0,465	$1,9200 \times 10^{-1}$
7	Rápido	0,001	0,700	$4,3887 \times 10^{-6}$
8	Rápido	0,010	0,660	$3,2041 \times 10^{-3}$
9	Rápido	0,100	0,480	$1,8443 \times 10^{-1}$

Em todas as situações o CCMT conseguiu recuperar a trajetória original. Particularmente, nas situações em que o ruído aplicado era mais elevado ( $\sigma = 0,1$ ), linhas 3, 6 e 9 da Tabela 6.4, o CCMT conseguiu recuperar a trajetória original e ainda manteve a distância DTW próxima de zero. Com  $\sigma$  em 0,01 e 0,001, o CCMT gerou trajetórias muito próximas às originais valores de DTW menores que  $10^{-4}$ , ver gráficos 6.12a, 6.12b, 6.13a, 6.13b, 6.14a, 6.14b e Tabela 6.4. Por exemplo, para uma trajetória com dados ruidosos gerados com  $\sigma$  em 0,01 e a mesma trajetória sem ruído os gráficos ficam sobrepostos, ver Figura 6.15a.

### 6.1.5 Gerenciamento de Locomoção

Esta seção descreve os resultados obtidos com o STRAGIC no gerenciamento de transição entre modos de locomoção. As três bases de dados dos modos de locomoção lento, médio e rápido foram agrupadas em uma única base de dados. STRAGIC identificou e construiu uma sub-rede neural para cada um dos três modos de locomoção, semelhante ao experimento da seção 6.1.3.

As transições foram realizadas utilizando os seguintes critérios: transição imediata e transição mais suave. Na transição imediata o próximo estado escolhido é o estado da trajetória alvo que possui menor distância euclidiana do estado atual. A transição mais suave é aquela que possui menor distância euclidiana de todas as conexões viáveis entre os nodos da trajetória alvo e de destino.

Depois das três redes criadas o STRAGIC gerou as interconexões entre nodos de duas trajetórias diferentes. O STRAGIC executa a transição entre os modos de locomoção através dos estados mais próximos das trajetórias que representam modos de locomoção. Para sair do modo lento para o modo rápido é preciso passar pelo modo moderado e em seguida para o rápido, pois não existem conexões entre os modos rápido e lento diretamente.

As transições imediatas e as transições mais suaves foram testadas para diferentes mudanças de modos de locomoção. A Figura 6.16 mostra os gráficos das oscilações aplicadas às articulações  $\beta$  de cada membro nas transições: lento para médio, Gráfico 6.16a; médio para rápido, Gráfico 6.16b; rápido para médio, Gráfico 6.16c; médio para lento, Gráfico 6.16d. Em todas os gráficos da Figura 6.16, a transição ocorreu no passo 100 da simulação, indicada pela linha vertical pontilhada. As transições aconteciam no momento que o STRAGIC recebia um comando “UP” ou “DOWN”. O comando “UP” executa a transição do modo de locomoção atual para o modo de locomoção que possui a velocidade superior mais próxima, o comando “DOWN” faz o inverso, executa a transição para a velocidade inferior mais próxima. Caso a velocidade solicitada não exista o STRAGIC continua no modo de locomoção atual.

A Figura 6.17 mostra os gráficos das transições mais suaves: lento para médio ocorrida no passo 102, gráfico 6.17a; médio para rápido ocorrida no passo 57, gráfico 6.17b; rápido para médio ocorrida no passo 56, gráfico 6.17c; médio para lento ocorrida no passo 69, gráfico 6.17d. Em cada uma destas situações a velocidade atual foi mantida até o passo de simulação 40, em seguida o comando de executar a melhor transição foi enviado.

A conexão que representa a transição imediata pode coincidir com a transição mais

suave. Entretanto, o algoritmo utilizado para construir as interações garante que a transição mais suave é a que possui a menor distância entre os dois estados que fazem parte das interconexões.

## 6.2 Dados Reais

Este experimento testa a capacidade do STRAGIC em ligar com dados obtidos a partir da locomoção de um animal. O animal escolhido foi um cachorro de médio porte, os dados sobre a locomoção foram extraídos de um vídeo, obtido no *youtube*. O vídeo mostra um cachorro andando numa esteira, como visto na Figura 6.20. A base de dados<sup>1</sup> armazena a posição  $(x, y)$  de cada articulação dos membros esquerdos do cachorro. O ponto  $(0, 0)$  está localizado no canto superior esquerdo da imagem, sendo que todos os outros pontos da imagem assumem valores positivos. A Figura 6.19 serve de referência para identificar as articulações dos membros do cachorro nas imagens obtidas a partir do vídeo. As articulações foram marcadas com a cor verde, para facilitar a segmentação já que a cor verde aparentemente não está presente nos quadros originais do vídeo.

A base de dados contém as coordenadas  $(x, y)$  das articulações do lado esquerdo do animal coletadas de um passo do cachorro, contabilizando 17 quadros convertidos em 17 amostras. Os quadros foram convertidos em imagens do tipo jpeg, ver Figura 6.18, as articulações de cada imagem marcadas manualmente com uma ferramenta livre de edição de imagem, Gimp<sup>2</sup>(*GNU Image Manipulation Program*). O Algoritmo de segmentação de imagem e detecção das regiões marcadas é descrito a seguir:

1. Carregue um quadro do vídeo;
2. Faça uma varredura na imagem avaliando cada pixel, começando do ponto  $(0, 0)$ ;
3. Caso o pixel encontrado seja verde, faça:
  - (a) Verifique se possui algum pixel vizinho verde;
  - (b) Se falso, crie um grupo para este pixel;
  - (c) Se verdadeiro, adicione este pixel ao grupo do seu vizinho;
4. Grupos com um tamanho abaixo de um determinado limiar são considerados ruídos e descartados;

---

<sup>1</sup><http://orivaldo.net/web/dissertacao/experimentos/video.dat>

<sup>2</sup><http://www.gimp.org/>

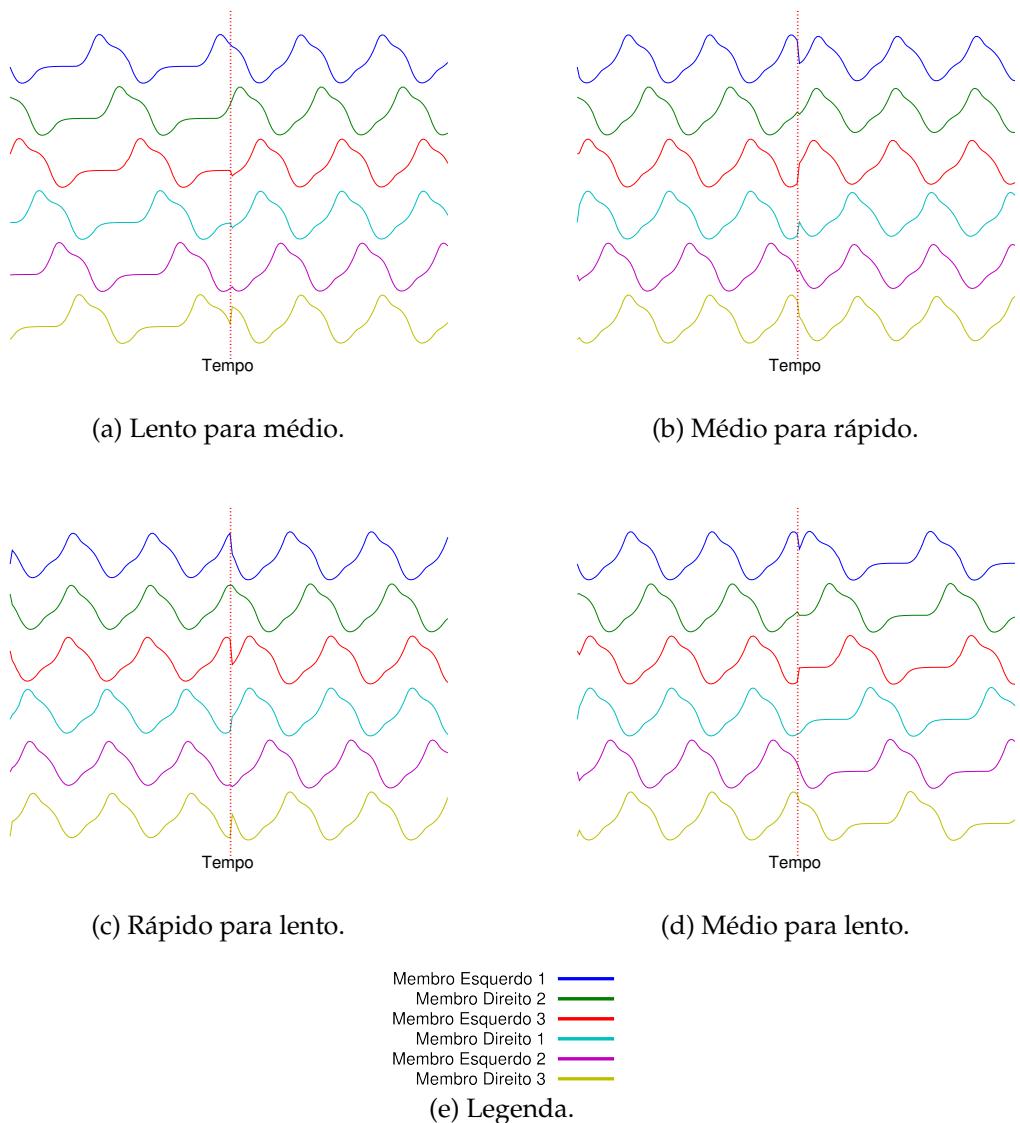


Figura 6.16: Transições imediatas entre os modos de locomoção.

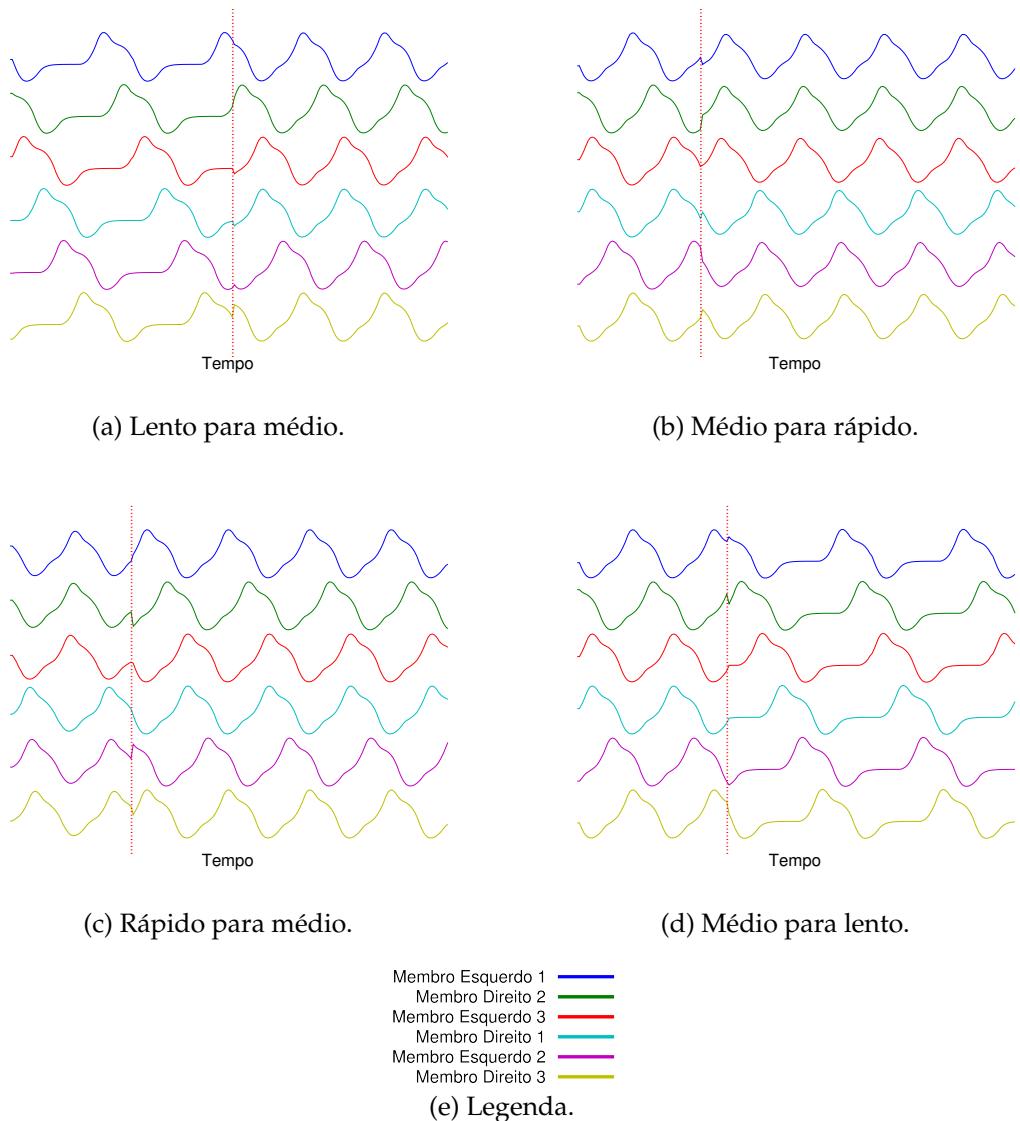


Figura 6.17: Melhor transição entre modos de locomoção.

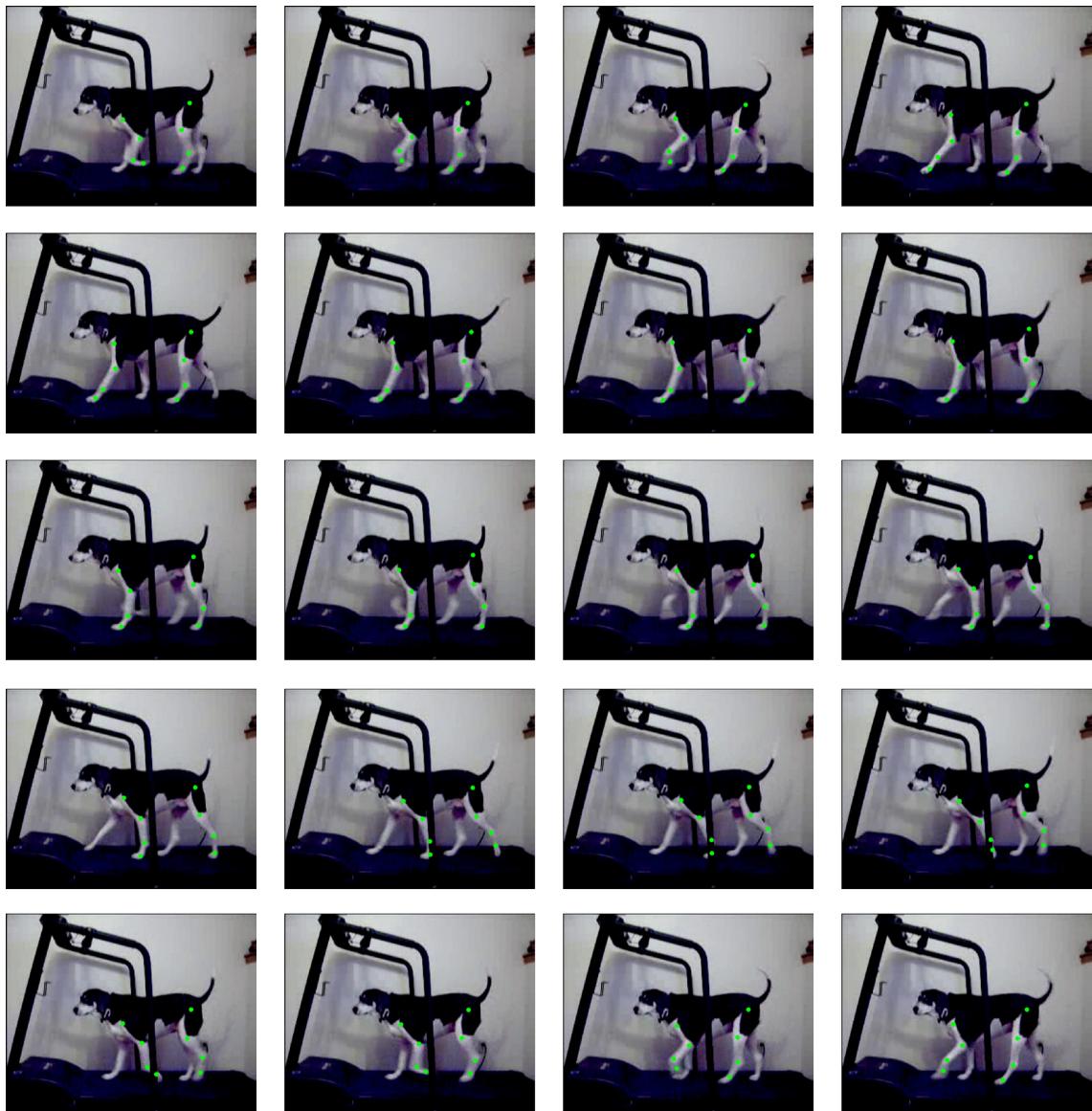


Figura 6.18: Quadros da locomoção de um cachorro com as articulações marcadas com verde.

Tabela 6.5: Taxa de êxito ao criar a rede.

Fator de Poda	Taxa de Êxito Média (%)	Desvio Padrão
0,4	17,567	1,14000
0,8	19,960	1,02590
1,2	0,220	0,17403

5. Calcule a coordenada média de cada grupo e armazene em uma linha do arquivo de dados;
6. Volte ao passo 1, se ainda existir quadro para avaliar;

O STRAGIC foi configurado como descrito no início da Seção 6.1. Mas, com um limiar de atividade alto o suficiente para permitir que todos os estados fossem representados pelos nodos, valor do limiar de atividade foi de 0,000001. O critério de parada utilizado foi a rede não crescer durante 15 iterações consecutivas.

A Tabela 6.5 mostra a taxa de êxito do STRAGIC ao criar uma rede capaz de gerar a trajetória original. A taxa de êxito é definida como a porcentagem do total de execuções em que as redes resultantes são capazes de gerar um trajetória de estado idênticas a original. O parâmetro que influencia a taxa de êxito é o fator de poda, três valores diferentes do fator de poda foram avaliados, Tabela 6.5. Para obter esta taxa de êxito, o STRAGIC foi executado 1000 vezes. Para obter a taxa de êxito média, 15 taxas de êxitos foram geradas. A melhor configuração dos parâmetros obteve uma taxa de êxito média igual a 19,96 porcento.

### 6.2.1 Discussão sobre Taxa de Êxito

Nos experimentos com dados artificiais a taxa de êxito foi de 100%, isto significa dizer que em todas as execuções do STRAGIC com dados artificiais as redes resultantes eram capazes de gerar uma trajetória de estados semelhante à original. No entanto, as trajetórias resultantes nem sempre eram idênticas ( $DTW = 0$ ) as originais.

No experimento com dados reais os resultados não foram como esperado, já que a taxa de êxito foi muito baixa. Para o STRAGIC gerar um trajetória cíclica com sucesso um requisito básico é que todos os nós da rede tenham no mínimo dois vizinhos. Algumas configurações foram testadas e a melhor configuração rodada milhares de vezes, obtendo assim uma taxa de êxito em torno de 20%.

As falhas acontecem quando o CCMT não cria uma conexão entre dois estados consecutivos na trajetória real, por estes dois estados estarem mais próximos, segundo

## 6.2. DADOS REAIS

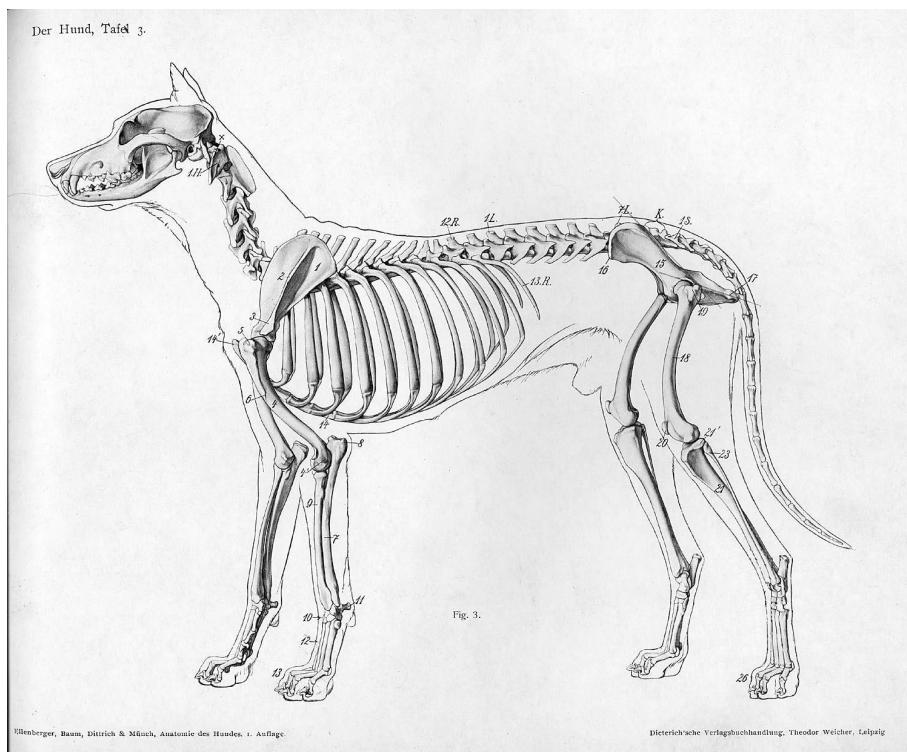


Figura 6.19: Anatomia lateral do esqueleto de um cachorro usada como referência para marcar a posição das articulações, fonte Wikipédia.



Figura 6.20: Marcação das articulações do cachorro.

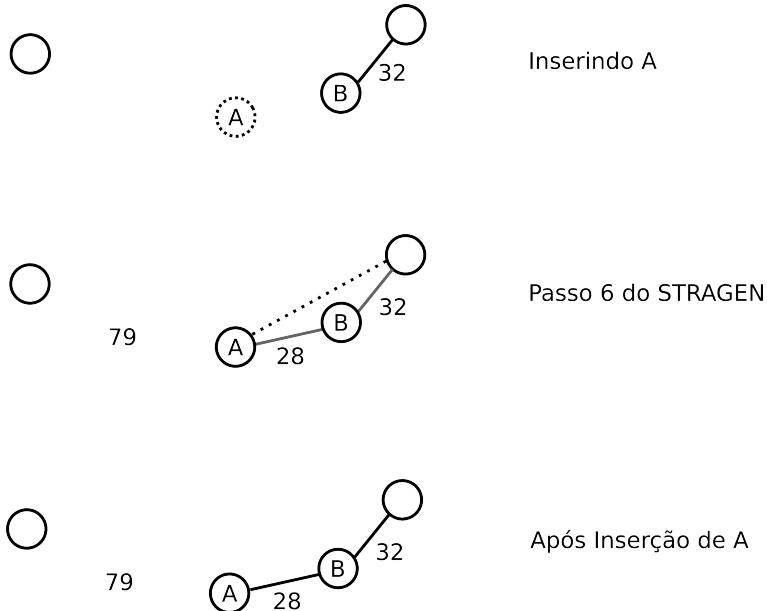


Figura 6.21: Situação em que o CCMT falha ao criar uma conexão.

a distância Euclidiana, de outros estados da trajetória real. Isto acontece por que o movimento da pata do animal durante um passo possui diferentes velocidades. Como os quadros do vídeo são gerados em uma velocidade constante, a diferença da posição espacial da pata do animal em quadros consecutivos pode variar muito. Quadros em que a velocidade de deslocamento da pata do robô é pequena a distância euclidiana entre a posição das patas também será pequena. Assim, estados de baixa velocidade estão mais próximos de se do que dos estados de alta velocidade.

Um exemplo de uma situação em que o CCMT pode falhar está presente nas Figuras 6.21 e 6.22. A Figura 6.21 ilustra uma situação em que o CCMT falha ao criar uma conexão entre dois nodos que representam estados consecutivos na trajetória real. Já a Figura 6.22 mostra os mesmos estados só que devido à ordem em que eles foram inseridos o CCMT conseguiu criar a conexão, mesmo tendo uma distância maior do que a das outras conexões.

## 6.3 Considerações

Este capítulo apresentou e avaliou os experimentos com dados artificiais e reais. Os resultados obtidos com os dados artificiais demonstram que o STRAGIC é capaz de controlar a locomoção de um robô simulado e gerenciar a transição entre modos de locomoção. Além disso, os resultados indicam que o STRAGIC comporta-se bem diante de dados ruidosos. Com dados reais, o desempenho não foi tão bom como nos dados

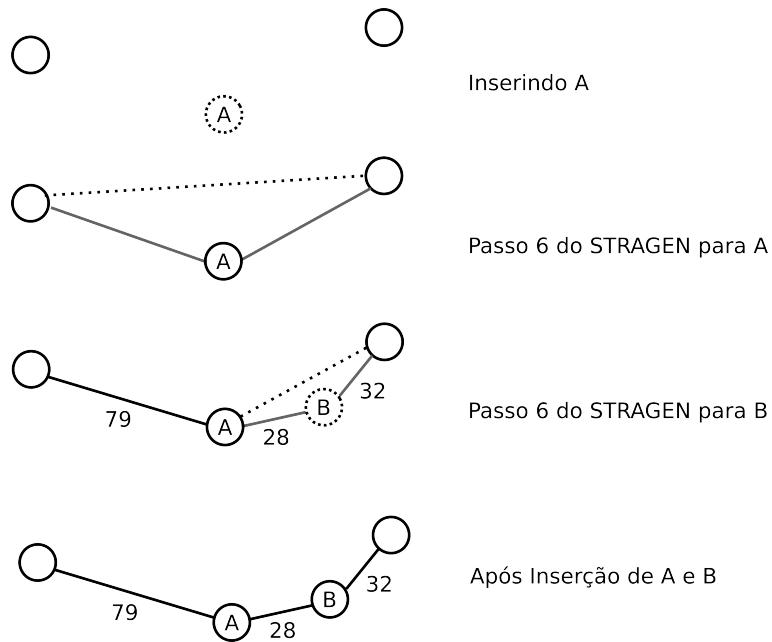


Figura 6.22: Situação em que o CCMT consegue criar uma conexão que possui uma distância maior.

artificiais, já que a taxa de êxito para os dados reais foi em torno de 20 %, enquanto que nos dados artificiais a taxa de êxito foi 100%. Investigando as possíveis causas para o valor baixo da taxa de êxito com dados reais surgiu algumas ideias para melhorar o CCMT do STRAGIC, discutidas em mais detalhes na Seção 7.1.

# 7

## Conclusão e Trabalhos Futuros

Esta dissertação tratou do controle de locomoção de robôs com membros inferiores por meio de redes neurais. Para tanto, foi realizado um estudo de dois modelos de redes neurais e a proposição de uma nova abordagem para o controle de locomoção de robôs com membros. A grande maioria dos trabalhos encontrados na literatura sobre controle de locomoção com redes neurais utilizam alguma abordagem relacionada a CPGs. Duas destas abordagens receberam destaque. A abordagem baseada na CNN de [Arena et al. \(2004\)](#), por possuir uma relação com mapas auto-organizáveis e a abordagem biologicamente inspirada de [Ijspeert et al. \(2007\)](#) por ser um tipo de abordagem muito referenciada na literatura de controle de locomoção com redes neurais.

O modelo proposto, o STRAGIC, é baseado num mapa auto-organizável de topologia variável chamado de STRAGEN. Sem contar com o STRAGEN, todo o projeto do STRAGIC foi realizado durante o trabalho nesta dissertação. O STRAGIC foi separado em dois módulos uma para o controle de locomoção e outro para o gerenciamento do modo de locomoção.

o STRAGIC é capaz de construir trajetórias de estado até mesmo quando as amostras são originadas de uma fonte ruidosa. A construção dos movimentos pode ser realizada através do balbuciamento motor [Benante e Araujo \(2007\)](#), um procedimento que possui plausibilidade no modo que os bebês aprendem seus primeiros movimentos. Os nodos do STRAGIC podem tratar diferentes tipos de informação, pois a dinâmica da rede pode ser modificada através da escolha do critério de vizinhança. O STRAGIC tem capacidade de lidar com dados que caracterizam a locomoção (ex. postura) não importando a abordagem utilizada para gerar estes dados.

Uma vantagem de trabalhar com estados que contêm informações que descrevem a postura do robô é que o STRAGIC não precisa definir a defasagem entre membros como nos modelos vistos no Capítulo 3. A informação da defasagem já está embutida

indiretamente nas informações dos estados.

A avaliação dos experimentos com dados gerados artificialmente é positiva, visto que o STRAGIC foi capaz de construir trajetórias de estados para o controle de diferentes modos de locomoção até mesmo quanto estes dados possuíam ruído e com uma taxa de êxito de 100%. Além disso, o STRAGIC foi capaz de executar a transição entre modos de locomoção com um simples comando de controle. No entanto, com os dados extraídos da locomoção de um animal real o STRAGIC apresentou um resultado abaixo do esperado com uma taxa de exito em torno de 20%. Este desempenho abaixo do esperado sugere mudanças no STRAGIC, mais especificamente no STRAGEN que responsável diretamente pelo controle de locomoção. Algumas sugestões de melhorias no STRAGEN são analisadas na Seção 7.1.

Alguns fatores provavelmente contribuíram para o exito com dados articiais do STRAGIC e consequentemente do STRAGEN. A alta dimensionalidade da amostra contribuiu para gerar uma distribuição dos dados tal que a distância entre amostras ficassem equivalente a sequência de estados da locomoção. O modo como os valores dos ângulos variam em cada modo de locomoção possibilitou ao modelo gerar as diferentes trajetórias até mesmo quando os dados destas trajetórias estavam todos misturadas em um única base de dados. Assim, o STRAGEN foi capaz de capturar uma propriedade emergente, o sincronismo entre membros para diferentes modos de locomoção, já que não era uma informação explicita nos dados de treinamento.

## 7.1 Contribuições para o STRAGEN

Os resultados dos experimentos com dados reais sugerem que a maioria das redes geradas não é capaz de criar uma trajetória cíclica, deste modo falhando ao controlar a locomoção do robô. Este problema acontece quando a rede resultante possui neurônios com um só vizinho, impossibilitando o algoritmo de geração de trajetórias construir trajetórias fechadas.

Além do passo de inserção de novos nodos, existem dois outros passos no algoritmo STRAGEN onde as conexões podem ser modificadas. Um passo é o de adaptação (passo 7) e o outro é o passo de poda (passo 9). Durante a fase de treinamento quando todas as amostras da base de dados são convertidas em nodos na rede, o passo de adaptação não provoca nenhuma mudança na rede. Já que a ação executada por este passo é mover o neurônio vencedor em direção a amostra de entrada, mas o neurônio vencedor e a amostra de entrada são iguais, na situação descrita.

De acordo com análise do comportamento do passo 7 e do passo 9, uma modificação

sugerida no passo de adaptação é a seguinte, caso o nó vencedor possua apenas um vizinho então, crie uma conexão entre o neurônio vencedor e o neurônio mais próximo que possuir apenas um vizinho. Assim, evitaria que as redes resultantes possuíssem nodos com apenas um vizinho. No passo de poda, a modificação sugerida seria no sentido de não retirar uma quantidade  $l$  de nós que possuem as menores distâncias para o nó vencedor ao invés de amarrar a quantidade de nós excluídos ao fator de poda.

O comportamento atual do passo de poda permite que a rede possua nós com diferentes quantidades de vizinhos. Deste modo, um nodo com apenas um vizinho inviabiliza o controle de locomoção e nós com mais de dois vizinhos dificulta a construção das trajetórias cíclicas. Além disto, o passo de poda pode corrigir eventuais conexões criadas incorretamente no passo de adaptação. Já que a poda retira conexões que possuem os valores mais elevados do desvio padrão.

A modificação no passo garante que a rede resultante possua uma estrutura homogênea ou que tenha um número máximo de vizinhos. No caso do problema de controle de locomoção é interessante que a rede seja homogênea, isto é, cada nó da rede possuindo dois vizinhos. Uma estrutura homogênea facilita a construção de trajetórias cíclicas.

## 7.2 Trabalhos Futuros

Com os resultados dos experimentos ficou claro que algumas questões ainda precisam ser tratadas no STRAGIC:

- Realizar experimentos para diferentes tipos de robôs com membros com o objetivo de testar a capacidade do modelo em lidar com diferente anatomias de robôs caminhantes. Verificando assim, a capacidade do modelo proposto em controlar diferentes robôs independente da quantidade de membros e da anatomia.
- Elaborar novos experimentos com dados reais e aprimorar a técnicas de aquisição de dados.
- Durante a realização dos experimentos com dados artificiais com ruído ou até mesmo sem ruído o limiar de atividade precisou ser ajustado para maximizar a taxa de exito. Uma possível contribuição para o modelo é gerar um mecanismo ajuste automático do limiar de atividade para gerar a melhor taxa de êxito ou uma taxa ótima.

- Aprofundar os estudos com osciladores neurais biologicamente inspirados implementá-los e compará-los com o STRAGIC.
- Investigar uma maneira de unir a modelagem matemática de CPGs e os mapas auto-organizáveis.
- Aprofundar a abordagem proposta no sentido de buscar plausibilidade biológica no funcionamento do sistema nervoso. Entender e modelar biologicamente, na medida do possível, o controle de um membro individualmente bem como coordenação de todos os membros.
- Implementar as modificações no STRAGEN propostas na Seção 7.1 realizar experimentos para avaliar o resultado destas modificações.
- Sobre a instalação do ambiente de simulação, devido as dificuldades iniciais de instalação principalmente em relação às dependências de bibliotecas. Muitas bibliotecas não estão instaladas por padrão no sistema operacional utilizado e não é trivial encontrar qual biblioteca resolve um erro apresentado durante a compilação. Diante disto, um tarefa importante para facilitar novas instalações é gerar um pacote de instalação (exemplo pacote Debian) do Gazebo para automatizar a instalação do simulador sem necessitar compilar o código fonte.

# Referências Bibliográficas

- Acebrón, J. A., Bonilla, L. L., Pérez Vicente, C. J., Ritort, F., and Spigler, R. (2005). The kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, **77**(1), 137–185.
- Arena, P., Caponetto, R., Fortuna, L., and Manganaro, G. (1997). Cellular neural networks to explore complexity. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, **1**, 120 – 136.
- Arena, P., Baglio, S., Fortuna, L., and Manganaro, G. (1998). Self-organization in a two-layer cnn. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, **45**(2), 157–162.
- Arena, P., Fortuna, L., and Branciforte, M. (1999). Reaction-diffusion cnn algorithms to generate and control artificial locomotion. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, **46**(2), 253–260.
- Arena, P., Fortuna, L., Frasca, M., and Marchese, C. (2001). Multi-template approach to artificial locomotion control. *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, **3**, 6 – 9.
- Arena, P., Fortuna, L., and Frasca, M. (2002). Multi-template approach to realize central pattern generators for artificial locomotion control. *International Journal of Circuit Theory and Applications*, **30**, 441 – 458.
- Arena, P., Fortuna, L., Frasca, M., and Sicurella, G. (2004). An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, **34**(4), 1823–1837.
- Ayers, J. and Witting, J. (2007). Biomimetic approaches to the control of underwater walking machines. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **365**(1850), 273–295.
- Bekey, G. A. (2005). *Autonomous Robots From Biological Inspiration to Implementation and Control*. Massachusetts Institute of Technology.
- Benante, R. (2008). *STRAGEN - State Trajectories Generator*. Ph.D. thesis, UFPE.
- Benante, R. and Araujo, A. (2007). Self-organizing maps to generate state trajectories of manipulators. *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, **1**, 1590–1595.

- Billard, A. and Ijspeert, A. J. (2000). Biologically inspired neural controllers for motor control in a quadruped robot. In *IJCNN '00: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 6*, page 6637, Washington, DC, USA. IEEE Computer Society.
- Bonet, B. and Geffner, H. (2001a). Planning and control in artificial intelligence: A unifying perspective. *Applied Intelligence*, **14**(3), 237 – 252.
- Bonet, B. and Geffner, H. (2001b). Planning as heuristic search. *Artificial Intelligence*, **129**, 5–33.
- Boyce, W. E. and Diprima, R. C. (2006). *Equações Diferenciais Elementares e Problemas de Valores de Contorno*. LTC, 8 edition.
- Chua, L. and Roska, T. (1993). The cnn paradigm. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, **40**(3), 147–156.
- Chua, L. and Yang, L. (1988a). Cellular neural networks: applications. *Circuits and Systems, IEEE Transactions on*, **35**(10), 1273–1290.
- Chua, L. and Yang, L. (1988b). Cellular neural networks: theory. *Circuits and Systems, IEEE Transactions on*, **35**(10), 1257–1272.
- Chua, L., Hasler, M., Moschytz, G., and Neirynck, J. (1995). Autonomous cellular neural networks: a unified paradigm for pattern formation and active wave propagation. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, **42**(10), 559–577.
- Durr, V., Krause, A., Schmitz, J., and Cruse, H. (2003). Neuroethological concepts and their transfer to walking machines. *The International Journal of Robotics Research*, **22**(3-4), 151.
- FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, **1**, 445 – 466.
- Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, **7**(9), 1441 – 1460.
- Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press.

- Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, **117**, 500 – 544.
- Holmes, P., Full, R. J., Koditschek, D., and Guckenheimer, J. (2006). The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, **48**(2), 207–304.
- Ijspeert, A. (2001). A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, **84**(5), 331–348.
- Ijspeert, A., Hallam, J., and Willshaw, D. (1999). Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, **7**(2), 151–172.
- Ijspeert, A., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, **315**(5817), 1416–1420.
- Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, **21**(4), 642–653.
- Ijspeert, A. J. and Kodjabachian, J. (1999). Evolution and development of a central pattern generator for the swimming of a lamprey. *Artitifical Life*, **5**(3), 247–269.
- Jing, X.-J. (2005). Behavior dynamics based motion planning of mobile robots in uncertain dynamic environments. *Robotics and Autonomous Systems*, **53**(2), 99–123.
- Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In *In First SIAM International Conference on Data Mining SDMâ2001*.
- Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, **43**, 59–69.
- Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, **21**(1-3), 1 – 6.
- Kohonen, T. and Hari, R. (1999). Where the abstract feature maps of the brain might come from. *Trends in Neurosciences*, **22**, 135–139.

---

## REFERÊNCIAS BIBLIOGRÁFICAS

---

- Marsland, S., Shapiro, J., and Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Netw.*, **15**(8-9), 1041–1058.
- Matsuoka, K. (1987). Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, **56**, 345 – 353.
- Maufroy, C., Kimura, H., and Takase, K. (2008). Towards a general neural controller for quadrupedal locomotion. *Neural Networks*, **21**, 667 – 681.
- Maufroy, C., Kimura, H., and Takase, K. (2010). Integration of posture and rhythmic motion controls in quadrupedal dynamic walking using phase modulations based on leg loading/unloading. *Autonomous Robots*, **28**, 331 – 353.
- McMahon, T. A. (1984). *Muscles, Reflexes, and Locomotion*. Princeton University Press.
- Myers, C., Rabiner, L., and Rosenberg, A. (1980). Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **28**(6), 623 – 635.
- Nagumo, J., Arimoto, S., and Yoshizawa, S. (162). An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, **50**, 2061 – 2070.
- Nakamura, Y., Mori, T., Sato, M. A., and Ishii, S. (2007). Reinforcement learning for a biped robot based on a cpg-actor-critic method. *Neural Netw.*, **20**(6), 723–735.
- Örjan Ekeberg (1993). A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, **69**, 363 – 374.
- Reeve, R. and Hallam, J. (2005). An analysis of neural models for walking control. *Neural Networks, IEEE Transactions on*, **16**(3), 733–742.
- Righetti, L. and Ijspeert, A. (2006). Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, pages 191–198, Philadelphia, USA.
- Ritter, H., Martinet, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps; An Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Santos, P. G. D., Cobano, J., Garcia, E., Estremera, J., and Armada, M. (2007). A six-legged robot-based system for humanitarian demining missions. *Mechatronics*, **17**(8), 417 – 430.

---

## REFERÊNCIAS BIBLIOGRÁFICAS

---

- Senin, P. (2008). Dynamic time warping algorithm review. Technical report, Information and Computer Science Department - University of Hawaii at Manoa.
- Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. The MIT Press.
- Song, S.-M. and Waldron, K. J. (1989). *Machines that walk: the adaptive suspension vehicle*. Massachusetts Institute of Technology.
- Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2006). *Robot Modeling and Control*. John Wiley & Sons, first edition.
- Sproewitz, A., Moeckel, R., Maye, J., and Ijspeert, A. J. (2008). Learning to move in modular robots using central pattern generators and online optimization. *International Journal of Robotics Research*, **27**(3-4), 423–443.
- Stein, R. B., Leung, K. V., Mangeron, D., and Oguztöreli, M. N. (1973). Improved neuronal models for studying neural networks. *Biological Cybernetics*, **15**, 1–9.
- Wu, Q. D., Liu, C. J., Zhang, J. Q., and Chen, Q. J. (2009). Survey of locomotion control of legged robots inspired by biological concept. *Science in China Series*, **52**, 1715 – 1729.
- Zou, F. and Nossek, J. (1993). Bifurcation and chaos in cellular neural networks. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, **40**(3), 166–173.

# **Appendices**

# A

## Comportamento Dinâmico de um Sistema Autônomo

O comportamento dinâmico de um sistema autônomo de duas células é descrito pelas equações de estado a seguir:

$$\begin{aligned}\dot{x}_1 &= -x_1 + (1 + \mu) \cdot y_1 - s \cdot y_2 + i_1; \\ \dot{x}_2 &= -x_2 + s \cdot y_1 + (1 + \mu) \cdot y_2 + i_2;\end{aligned}\quad (\text{A.1})$$

Este sistema de equações possui dois termos de bias,  $i_1$  e  $i_2$ , um para cada equação de estado. Para uma determinado valor dos parâmetros  $\mu$  e  $s$  este sistema não-linear oscila estável dentro de um ciclo limite no plano de fase. Este plano de fase possui como eixo das abscissas o  $x_1$  e o das ordenadas  $x_2$  [Arena et al. \(1997\)](#).

Na região linear  $R_l$  nenhuma variável de estado está saturada:

$$R_l = \{x \in \mathbb{R}^2 : |x_i| < 1, i = 1, 2\}; \quad (\text{A.2})$$

Nesta região linear a matriz Jacobina  $J_l$  identifica o ponto de equilíbrio:

$$J_l = \begin{pmatrix} \mu & -s \\ s & \mu \end{pmatrix}; \quad (\text{A.3})$$

Para que o ponto de equilíbrio esteja sempre dentro da região linear os parâmetros  $\mu$  e  $s$  terão sempre valores positivos [Arena et al. \(1997\)](#).

O sistema de equações não lineares A.1 de uma célula CNN, para valores pequenos dos estados internos, comporta-se semelhante a um sistema linear com autovalores complexos. Isto porque, para  $x_1$  e  $x_2 \in [-1, 1]$  a função de saída (ver figura 3.5) tem

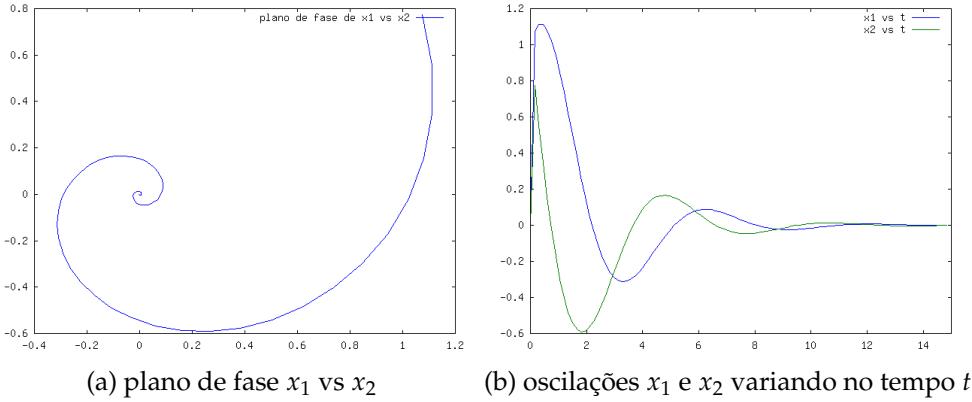


Figura A.1: Plano de fase e oscilações no tempo, para  $\mu$  assumindo valores negativos.

o comportamento exatamente igual ao estado interno, assim  $y_i = x_i$  para  $i \in \{1, 2\}$ . Considerando  $i_1 = 0$  e  $i_2 = 0$  é possível chegar à equação A.4 na forma matricial a partir do sistema de equações A.1.

$$\dot{\mathbf{x}} = \begin{pmatrix} \mu & -s \\ s & \mu \end{pmatrix} \mathbf{x} \quad (\text{A.4})$$

Ao calcular a solução do sistema linear A.4 os autovalores encontrados serão do tipo  $\mu \pm is$ , onde  $\mu$  e  $s$  são reais,  $\dot{\mathbf{x}} = [\dot{x}_1 \quad \dot{x}_2]^T$  e  $\mathbf{x} = [x_1 \quad x_2]^T$ . O mesmo sistema A.4 pode ser reescrito da forma escalar:

$$\dot{x}_1 = \mu x_1 - sx_2, \quad \dot{x}_2 = \mu x_1 + sx_2. \quad (\text{A.5})$$

Introduzindo coordenadas polares  $r, \theta$  dadas por

$$r^2 = x_1^2 + x_2^2, \quad \operatorname{tg}\theta = x_1/x_2.$$

Diferenciando essas equações, resultam em

$$r\dot{r} = x_1\dot{x}_1 + x_2\dot{x}_2, \quad (\sec^2\theta)\dot{\theta} = (x_1\dot{x}_2 + x_2\dot{x}_1)/x_1^2. \quad (\text{A.6})$$

Substituindo as equações A.5 na primeira das equações A.6, obtém-se

$$\dot{r} = \mu r, \quad (\text{A.7})$$

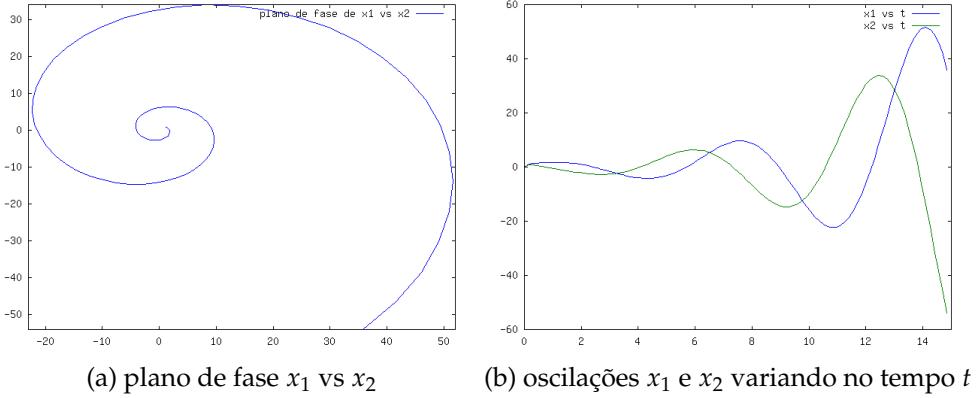


Figura A.2: Plano de fase e oscilações no tempo para  $\mu$  assumindo valores positivos.

e resolvendo a diferencial da equação A.7 acima, resulta em:

$$r = ce^{\mu t}, \quad (\text{A.8})$$

onde  $c$  é uma constante. Repetindo o procedimento anterior, substituindo as equações A.5 na segunda equação de A.6 e usando  $\sec^2 \theta = \frac{1}{\cos^2 \theta} = \frac{r^2}{x_1^2}$  da trigonometria, tem-se

$$\dot{\theta} = -s. \quad (\text{A.9})$$

Logo,

$$\theta = -st + \theta_0, \quad (\text{A.10})$$

onde  $\theta_0$  é o valor de  $\theta$  quando  $t = 0$  Boyce e Diprima (2006).

As equações A.8 e A.10 são equações paramétricas em coordenadas polares das trajetórias do sistema A.4. Analisando estas equações observa-se que  $\theta$  está inversamente relacionado com  $s$  e raio  $r$  está diretamente relacionado com  $\mu$ . Estas equações definem a trajetória de uma espiral que se afasta ou se aproxima da origem e que cresce no sentido horário ou anti-horário. Considerando  $s > 0$ , para valores grande de  $s$ ,  $\theta$  inicialmente assume valores pequenos. Para valores positivos de  $\mu$  o raio cresce quando o tempo tende ao infinito (figura A.2), já para valores negativos de  $\mu$  o raio tende a zero, com o passar do tempo, ver figura A.1. Logo, se um dos autovalores possuírem parte real positiva, então o ponto de equilíbrio é instável. Caso contrário, se todos os autovalores possuírem parte real negativa, então o ponto de equilíbrio é assintoticamente estável Zou e Nossek (1993). A figura A.2 mostra o plano de fase e as oscilações para um autovalor cuja parte real é positiva, já a figura A.1 mostra a situação em que a parte real é negativa.

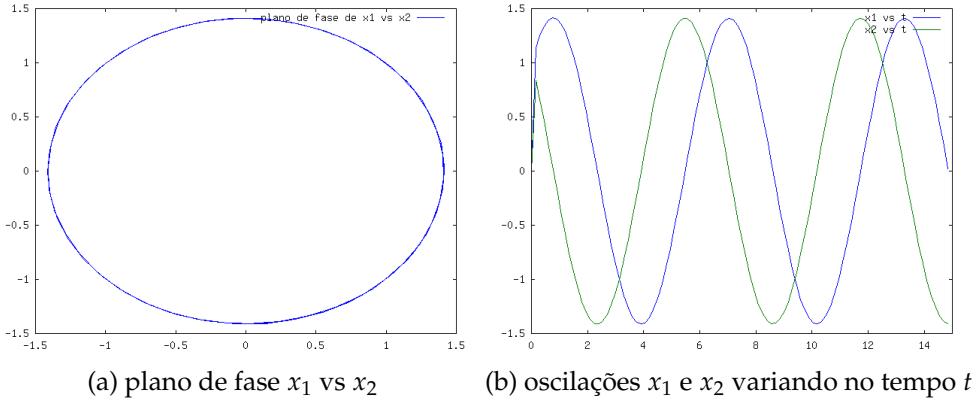


Figura A.3: Plano de fase e oscilações no tempo para  $\mu = 0$ .

No caso em que  $\mu = 0$  o sistema possuirá apenas autovalores imaginários puros. A primeira equação em A.8 passará a ser  $r = c$ , onde  $c$  é uma constante. Logo, as trajetórias se tornam círculos centrados na origem, percorridos no sentido horário se  $s > 0$  e no sentido anti-horário caso  $s < 0$ . Quando os autovalores são imaginários puros as trajetórias são elipses centradas na origem. Esta situação é ilustrada na figura A.3.

Quando os valores do estado interno  $x_i$  saem do intervalo  $[-1,1]$  a CNN entra numa região de saturação  $R_s$ . Nesta região  $R_s$ , se existir um ponto de equilíbrio então a região toda pertence à bacia atratora do ponto de equilíbrio. Ou seja, qualquer trajetória dentro de uma região de saturação irá convergir assintoticamente para o ponto de equilíbrio correspondente a esta região Zou e Nossek (1993).

A região saturada  $R_s$  é definida por:

$$R_s = \{x \in \mathbb{R}^2 : |x_i| \geq 1, i = 1, 2\}; \quad (\text{A.11})$$

# B

## O Simulador

O projeto Gazebo é um simulador de alta fidelidade de ambientes dinâmicos ao ar livre. Ele surgiu para complementar os projetos *Player* e *Stage*. O *Player* é um servidor de dispositivos em rede e o *Stage* é apropriado para simular uma enorme população de robôs moveis em complexos ambientes 2D. Gazebo é projetado para reproduzir com precisão ambientes dinâmicos. Todos os objetos simulados possuem massa, velocidade, atrito, e outros atributos que tornam seu comportamento realístico quando empurrados, puxados, caídos ou carregados (Koenig e Howard, 2004).

Os robôs são estruturas dinâmicas compostas de corpos rígidos conectados através de articulações. Forças lineares e angulares podem ser aplicadas a superfícies e articulações para gerar locomoção e interação com o ambiente. Este ambiente pode conter paisagens, construções estruturadas e outros objetos criados pelos usuário. Quase todos os aspectos da simulação são controláveis, desde as condições de iluminação até coeficientes de atrito (Koenig e Howard, 2004).

Gazebo é software de código aberto, logo disponível livremente. Ele oferece um ambiente rico o suficiente para rapidamente desenvolver e testar sistemas multi-robôs. Sendo, uma ferramenta simples, escalável e eficiente que tem o potencial de levar o campo de pesquisa em robótica a um número maior de pesquisadores (Koenig e Howard, 2004).

Gazebo é compatível com o servidor de dispositivos, *Player*, desde o desenvolvimento de sua base. O hardware simulado no Gazebo é projetado para refletir com precisão o comportamento do dispositivo físico equivalente. Como resultado, um programa cliente vê uma interface idêntica comparando o robô real e o simulado (Koenig e Howard, 2004).

Gazebo é geralmente usado em conjunto com o *Player*. O *Player* trata o Gazebo como um dispositivo normal capaz de enviar e receber dados. Do ponto de vista do usuário,

a interface para os modelos simulados no Gazebo são equivalentes a dos dispositivos reais. A interface do Gazebo não é limitada ao *Player* apenas. Uma biblioteca de baixo-nível fornece um mecanismo para dispositivos robóticos de terceiros interagirem com o Gazebo (Koenig e Howard, 2004).

Gazebo dá suporte a uma API simples: para adição de robôs, atuadores, sensores e objetos arbitrários; e pontos de acesso para interação com os programas clientes. Uma camada abaixo desta API encontra-se as bibliotecas que manipulam a simulação física e a visualização. As bibliotecas de terceiros utilizadas no simulador foram escolhidas baseadas em seu status de código aberto, base de usuário ativa e maturidade (Koenig e Howard, 2004).

As interfaces fornecem o meio pelo qual os programas clientes podem acessar e controlar modelos. Comandos enviados para uma interface pode instruir um modelo para mover uma articulação, mudar a configuração de sensores, ou requisitar dados. Um robô não pode executar tarefas sem sensores. Um sensor no Gazebo é um dispositivo abstrato sem uma representação física. Existem três sensores implementados: odometro, de distância, e câmera (Koenig e Howard, 2004).

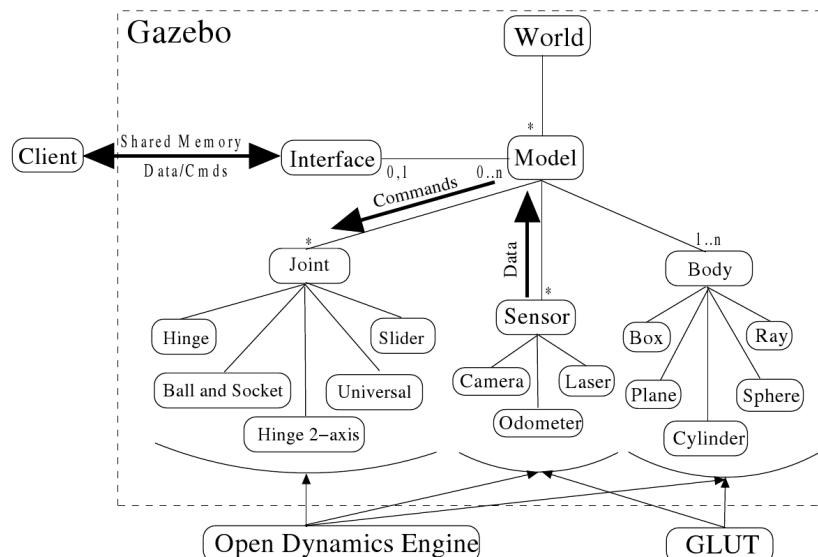


Figura B.1: Estrutura Geral dos componentes do Gazebo obtida em Koenig e Howard (2004)

A arquitetura do simulador é ilustrada na figura B.1. O componente *World* representa o conjunto de todos os modelos e as condições do ambiente como iluminação e gravidade. As bibliotecas de terceiros fazem interface com o gazebo no mais baixo nível da arquitetura. Com isto, evita-se que os modelos criados para o gazebo se tornem dependentes de ferramentas específicas que podem mudar no futuro. Os clientes enviam

---

comandos e recebem dados através de uma interface de memória compartilhada. Um modelo pode ter muitas interfaces para funções como controle de articulações e captura de imagens da câmera [Koenig e Howard \(2004\)](#).

A ODE (*Open Dynamics Engine*) é o motor de física mas amplamente utilizado na comunidade de código aberto (*open source*). Ele é projetado para simular a dinâmica e a cinemática relacionadas a corpos rígidos articulados. Este motor inclui muitos recursos, tais como diversas articulações, detecção de colisão, massa, funções de rotação e muitas geometrias. Gazebo utiliza destes recursos fornecendo uma camada de abstração situada entre a ODE e seus modelos. Esta camada facilita a criação tanto de objetos simples quanto abstratos mantendo toda as funcionalidades fornecidas pela ODE [Koenig e Howard \(2004\)](#).

Para a visualização o Gazebo utiliza a OpenGL e a GLUT (*OpenGL Utility Toolkit*) por atender o requisito de ser rápido e sofisticado. A OpenGL é uma biblioteca padrão para a criação de aplicações 2D e 3D interativas. Ela é uma plataforma independente, altamente escalável, estável, e evolui continuamente. GLUT é um conjunto de componentes básicos para construção de interface gráfica (janelas) para as aplicações OpenGL. Ele também fornece um mecanismo para interação com o usuário através de mouse e teclado [Koenig e Howard \(2004\)](#).

Um ambiente completo é essencialmente uma coleção de modelos e sensores. O chão e as construções representam modelos estacionários enquanto o robô e outros objetos são dinâmicos. Os componentes gerais envolvidos na simulação são: modelos, corpos e articulações. Um modelo é um objeto que mantém uma representação física. Ele engloba desde uma simples forma geométrica a complexos robôs. Os modelos são compostos de no mínimo um corpo rígido, zero ou mais articulações, sensores, e interfaces para facilitar o fluxo de dados. Os corpos representam os blocos básicos de um modelo. Sua representação física assume formas geométricas como caixas, esferas, cilindros, planos e linhas. Cada corpo possui massa, atrito, fator de elasticidade e propriedades de renderização (cor, textura, transparência, etc.). As articulações fornecem o mecanismo para conectar corpos e formar relacionamentos cinemáticos e dinâmicos. Várias articulações estão disponíveis incluindo articulações de rolagem. Um cuidado especial precisa ser tomado quando muitas articulações são conectadas pois, a simulação pode perder a estabilidade se parâmetros incorretos forem escolhidos [Koenig e Howard \(2004\)](#).

A criação de modelos é feita a mão. O processo começa com a escolha dos corpos apropriados e das articulações necessárias para construir um modelo correto em aparência e funcionalidade. O passo seguinte é juntar corpos utilizando articulações. O

---

---

resultado final é uma representação física completa de um modelo Koenig e Howard (2004).