Inteligência Artificial Aplicada no Controle do Caminhar e na Evolução da Morfologia de Robôs Móveis Simulados

Milton Roberto Heinen
UFRGS, Informatics Institute
Porto Alegre, RS, Brazil, 91501-970
mrheinen@inf.ufrgs.br

and

Fernando Santos Osório
UNISINOS, Applied Computing
São Leopoldo, RS, Brazil, 93022-000
fosorio@unisinos.br

Abstract

This paper describes our research and experiments with autonomous robots, in which we use neural networks to generate and control stable gaits of simulated legged robots. The experiments were conducted into a physically based virtual reality simulation environment. In our approach, the gait control is accomplished using an Elman network, and the synaptic weights are evolved using Genetic Algorithms. The proposed model also allows the evolution of robots' morphology together with the control parameters. The model validation was performed by several experiments, and the results showed that it was possible to generate stable gaits in an efficient manner using the proposed neural network model.

1 Introdução

Os robôs móveis autônomos tem atraído a atenção de um grande número de pesquisadores, devido ao desafio que este novo domínio de pesquisas propõe: dotar sistemas de uma capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos [30]. Atualmente os robôs móveis atuam em diferentes áreas, como desarmamento de bombas, exploração de ambientes hostis, e a condução de veículos de forma semi-autônoma [20]. A maioria dos robôs móveis desenvolvidos até o momento se deslocam através do uso de rodas, o que facilita bastante o controle, mas impede que eles sejam capazes de se deslocarem em ambientes irregulares que possuam desníveis e degraus [23]. Assim, para que um robô móvel possa se deslocar em ambientes irregulares, ele precisaria ser dotado do mesmo mecanismo de locomoção utilizado pelos seres humanos e a um grande número de seres vivos, ou seja, ele precisaria de pernas ou articulações [2, 35].

Mas o desenvolvimento de robôs com pernas que consigam se deslocar livremente em ambientes irregulares é uma tarefa bastante árdua, que exige a configuração de diversos parâmetros relativos ao caminhar. A configuração manual destes parâmetros exige muitas horas de um especialista humano, e os resultados obtidos são sub-ótimos e dependentes da arquitetura específica do robô [5]. Desta forma, seria bastante útil realizar a configuração do caminhar de forma automática, através da utilização de técnicas de aprendizado de máquina (*machine learning* – ML) [32]. Uma das técnicas de ML mais adequadas para solucionar este tipo de problema são os algoritmos genéticos (*genetic algorithms* – GA) [12], pois segundo a teoria da evolução natural das espécies [6], os mecanismos de locomoção utilizados pelos seres vivos são um resultado da evolução natural, o que torna o uso de GAs uma solução biologicamente inspirada [35]. Do ponto de vista computacional, os GAs também são bastante adequados, pois conseguem realizar uma busca multi-critério em um espaço multi-dimensional, e não necessitam de informações locais para a correção do erro nem do cálculo do gradiente [31].

Mas apesar de serem bastante adequados para a configuração automática do caminhar, os algoritmos genéticos não são muito eficientes de serem utilizados diretamente em robôs reais. Isto ocorre porque

os GAs são técnicas de aprendizado de máquina muito "pesadas" do ponto de vista computacional, que exigem centenas de gerações até que se chegue a uma solução razoável [43]. De fato, se forem utilizadas 100 gerações no algoritmo genético, e uma população de 50 indivíduos (valores típicos), será necessário que sejam realizados 5000 testes com o robô real. Se cada teste levar um minuto, serão necessárias 83,33 horas para a realização de um único experimento. Este uso prolongado do robô, além de causar um desgaste excessivo dos componentes, necessita de supervisão humana para tarefas como o reposicionamento e a troca ou recarga das baterias [5]. Desta forma, o uso de simulação baseada em física surge como uma alternativa viável para tornar a configuração do caminhar de um robô móvel mais eficiente, pois ao invés dos experimentos serem realizados diretamente em um robô real, eles podem ser realizados primeiramente em um robô simulado, e após o término do aprendizado a configuração aprendida pode ser portada para o robô real, o que economiza muitas horas de treinamento e evita o desgaste dos equipamentos [43].

O objetivo deste artigo é descrever o simulador LegGen [15, 16, 17, 18, 19], que é um simulador capaz de realizar a configuração do caminhar de robôs com pernas de forma automática através do uso de técnicas de aprendizado de máquina. Em nossos trabalhos anteriores [16], o controle do caminhar foi realizado através de um autômato finito cujos parâmetros forma evoluídos através de de algoritmos genéticos. Também foi realizado um estudo comparativo entre diferentes modelos de robôs de quatro e seis pernas [19] e entre diferentes funções de *fitness* [17]. Neste artigo, o controle das juntas do robô é realizado através de redes neurais artificiais (*artificial neural networks* – ANN) [14] cujos pesos sinápticos são evoluídos através de GAs. Além disto, a morfologia do robô é evoluída em conjunto com os pesos da rede neural.

Este artigo está estruturado da seguinte forma: A Seção 2 descreve diversos trabalhos do estado da arte da área em questão; A Seção 3 descreve as técnicas de aprendizado de máquina utilizadas; A Seção 4 descreve o uso de simulação baseada em física e a biblioteca ODE; A Seção 5 descreve o modelo proposto, o protótipo implementado, e o robô utilizado nas simulações; A Seção 6 descreve os experimentos realizados e os resultados obtidos; Por último, a Seção 7 traz as conclusões finais e as perspectivas futuras.

2 Trabalhos relacionados

O controle de locomoção em robôs com pernas é um problema de busca em um espaço de estados multi-dimensional que vem desafiando os pesquisadores a várias décadas [2]. Este controle requer a especificação e a coordenação dos movimentos de todas as pernas do robô, enquanto são considerados fatores como a estabilidade e a fricção em relação a superfície de contato (solo) [24]. Esta área de pesquisas possui uma clara ligação com o controle de locomoção realizado pelos animais, e muitos das pesquisas realizadas até o momento se inspiram no caminhar realizado por animais como os mamíferos e os insetos [37]. Como trabalhos pioneiros desta área podemos destacar os primeiros robôs com pernas realmente independentes, como o "Phony Pony" desenvolvido por Frank e McGhee [29], onde cada junta foi controlada por uma simples máquina de estados finita, até o controle algorítmico bem sucedido desenvolvido por Raibert [36], que era capaz de manter a estabilidade dinâmica em robôs de uma (*monopod*), duas e quatro pernas.

Na área de controle inteligente de robôs com pernas, os primeiros trabalhos datam do final dos anos 80 e início dos anos 90, como por exemplo o trabalho de Lewis [28], que utilizou algoritmos genéticos para a evolução dos controladores de um robô de seis pernas (*hexapod*). Neste trabalho, o controlador foi evoluído em um robô cujo caminhar era inspirado no caminhar dos insetos. Através de vários estágios de evolução, seu comportamento foi sendo modificado até atingir um caminhar razoavelmente satisfatório. Bongard [3] evoluiu os parâmetros de uma rede neural Artificial dinâmica utilizada para controlar diversos tipos de robôs simulados. Busch [4] utilizou Programação Genética para evoluir os parâmetros de controle de diversos tipos de robôs, simulados utilizando o pacote de software DynaMechs[1]. Jacob [22] utilizou Aprendizado por Reforço para o controle de um robô de quatro pernas (*tetrapod*) simulado através da biblioteca de software ODE. Reeve [37] utilizou algoritmos genéticos para a evolução dos parâmetros de diversos modelos de redes neurais utilizadas para o controle de diversos *tetrapods* simulados utilizando o DynaMechs.

Na maioria das abordagens descritas acima, a função de *fitness* utilizada foi a distância percorrida pelo robô durante um certo período de tempo. Embora esta função de *fitness* seja largamente utilizada, ela pode fazer com que a evolução privilegie formas de caminhar pouco estáveis em detrimento de soluções um pouco mais lentas porém muito mais estáveis [13]. Em nossos estudos, além da distância percorrida pelo robô, foram utilizadas como critério de *fitness* informações sensoriais, provenientes de um giroscópio simulado, a fim de se garantir que os caminhares obtidos fossem tanto rápidos quanto estáveis [15].

[1]DynaMechs – http://dynamechs.sourceforge.net/

2.1 Vida artificial

Vida artificial (*Artificial Life* - ALife) é o nome dado à disciplina que estuda a vida natural através da tentativa de recriar fenômenos biológicos em computadores ou outros meios artificiais [25]. Complementa a abordagem analítica tradicional da biologia com uma abordagem sintética onde, ao invés de estudar os fenômenos biológicos através de ver como funcionam os organismos vivos já constituídos, cria um sistema que se comporta como um organismo vivo [27, 35]. As tentativas de recriar os fenômenos biológicos de maneira artificial podem resultar não só na melhor compreensão teórica dos fenômenos estudados, como também em aplicações práticas nas áreas de robótica, medicina, nanotecnologia e engenharia [33].

Um dos pioneiros na área de vida artificial foi Karl Sims, que em [40, 41] utilizou um ambiente tridimensional baseado em física para a evolução de criaturas virtuais. Nestas criaturas, o sistema de controle foi evoluído em conjunto com a morfologia, o que é biologicamente plausível, pois segundo Pfeifer [35], na natureza o sistema nervoso evoluiu em conjunto com a morfologia dos seres vivos. O genótipo utilizado é constituído por um grafo direcionado, que determina as conexões entre os diversos segmentos da criatura virtual. A Figura 1 mostra algumas criaturas virtuais de Karl Sims evoluídas para caminhar.

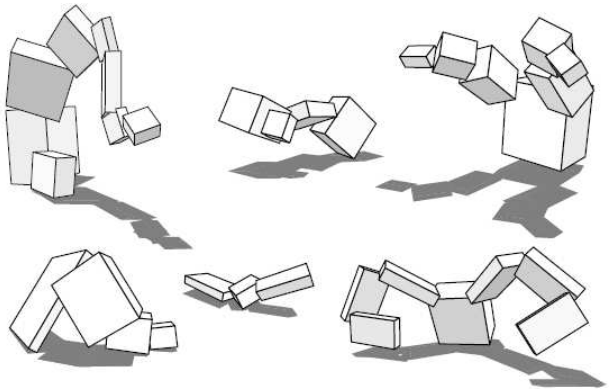


Figure 1: Criaturas virtuais de Sims [41]

Quando a morfologia é evoluída em conjunto com o sistema de controle, o espaço de busca cresce exponencialmente, o que dificulta a evolução e exige um maior poder computacional. Para manter a complexidade em níveis aceitáveis, são necessárias restrições no modelo. Nas criaturas de Karl Sims, uma das restrições impostas é o formato dos segmentos – as criaturas são constituídas de corpos rígidos (cubos) de tamanhos variados. Um modelo sem restrições dificilmente irá convergir para uma solução aceitável [35]. Já em um modelo com muitas restrições haverá menos variabilidade nas criaturas evoluídas.

3 Inteligência artificial e aprendizado de máquina

Segundo Mitchell [32], um programa aprende quando a sua performance melhora com a experiência em uma determinada tarefa. Dentre as diversas técnicas de aprendizado de máquina que podem ser utilizadas na tarefa em questão, neste trabalho optou-se pelos algoritmos genéticos e pelas redes neurais artificiais.

3.1 Algoritmos genéticos

Os algoritmos genéticos são métodos de busca estocástica baseados na Teoria da Evolução Natural das Espécies [6], criados por John Holland nos anos 60 [21]. Os algoritmos genéticos trabalham com uma população de soluções iniciais, chamadas cromossomos, que através de diversas operações vão sendo evoluídas até que se chegue a uma solução que melhor atenda a algum critério específico de avaliação. Para que isto ocorra, a cada geração os cromossomos são avaliados segundo uma função que mede o seu nível de aptidão, chamada de função de *fitness*. Os cromossomos que tiverem o melhor *fitness* são selecionados para darem origem a próxima geração, através de operações como cruzamentos e mutações. Desta forma, a tendência é que a cada geração o conjunto de soluções seja melhorado até atender aos objetivos desejados [31].

Para a implementação dos algoritmos genéticos no protótipo do modelo proposto, foi selecionada a

biblioteca de software GAlib[2], desenvolvida por Matthew Wall do MIT. A GAlib foi selecionada por ser uma das mais completas, eficientes conhecidas bibliotecas de software para a simulação de algoritmos genéticos, e também por ser uma solução gratuita e de código aberto, baseada em C++.

No simulador implementado, foi utilizado o algoritmo genético com populações sobrepostas proposto por [7], e foram adotados genomas do tipo real. Para se reduzir o espaço de busca, foram utilizados alelos para limitar o conjunto de valores gerados para cada atributo. O tipo de cruzamento escolhido foi o cruzamento em um ponto, e o esquema de seleção adotado foi o *stochastic remainder sampling selector*, que segundo [12] possui um desempenho superior ao esquema da roleta (*roulette wheel*). O método de escala do *fitness* utilizado foi o *sigma truncation*, que permite que o *fitness* assuma valores negativos.

3.2 Redes neurais artificiais

Através de um modelo abstrato e simplificado dos neurônios humanos é possível desenvolver um simulador que seja capaz de classificar, generalizar e aprender funções desconhecidas. Um dos modelos de aprendizado neural mais utilizados na atualidade é o modelo denominado *backpropagation* [39]. Para utilizar o algoritmo *backpropagation*, é necessário um conjunto de dados de exemplos de padrões com as respostas esperadas (padrões e classes correspondentes), que é dividido em uma base de aprendizado e uma base de validação (teste de generalização). Este tipo de aprendizado é conhecido como aprendizado supervisionado com validação cruzada [14].

Uma das principais limitações do algoritmo *backpropagation* é a necessidade de uma base de treinamento com as saídas desejadas, o que nem sempre é possível de ser obtido. Em problemas de robótica autônoma, por exemplo, não é comum se ter informações locais para a correção dos pesos, mas apenas uma medida que informa o desempenho de cada solução em relação as outras [38, 18]. Esta medida de desempenho não é suficiente para o cálculo do erro através do algoritmo *backpropagation*, mas geralmente é suficiente para a definição de uma função de *fitness*.

Uma alternativa viável para resolver este problema seria a utilização das redes neurais em conjunto com os algoritmos genéticos, de forma que os pesos sinápticos fossem evoluídos pelo GA, o que dispensaria o uso do algoritmo *backpropagation*. Assim, a combinação das duas técnicas permite que se utilize todo o poder de representação das redes neurais artificiais como um aproximador universal de funções [14] mesmo em problemas onde não é possível se obter de antemão informações locais para a descida do gradiente.

4 Simulação de robôs móveis

Quando se deseja realizar experimentos em robótica móvel, duas alternativas são possíveis: (i) realizar os experimentos diretamente em um robô real; ou (ii) realizar os experimentos utilizando um robô simulado em um ambiente virtual realista [35]. A utilização de um robô real possui a vantagem de tornar reais os resultados obtidos, mas o uso de simulação possui as seguintes vantagens [26]:

- Na simulação não existe o risco de se danificar o robô;
- A troca ou recarga de baterias e a manutenção do robô não são necessárias;
- O reposicionamento do robô pode ser realizado sem a intervenção humana;
- O relógio da simulação pode ser acelerado, reduzindo assim o tempo de aprendizado;
- Pode-se testar várias arquiteturas e modelos diferentes de robôs antes da construção física, e assim descobrir com antecedência qual modelo de robô é mais eficiente.

Por estes motivos, optou-se por realizar os experimentos utilizando robôs simulados em um ambiente virtual que implementa as leis da física através da biblioteca ODE, que é uma biblioteca de software que permite a realização de simulações da dinâmica de corpos rígidos articulados com bastante realismo.

4.1 Simulação baseada em física

Para que uma simulação de robôs móveis seja realista, diversos elementos do mundo real precisam estar presentes no modelo de simulação, para que os corpos se comportem de forma similar à realidade. Em especial, é necessário que um robô sofra quedas se não for bem controlado ou se não estiver bem posicionado, e que colida contra os objetos de forma realista. Para que isto ocorra, é necessário que as leis da física sejam modeladas no ambiente de simulação (gravidade, inércia, fricção e colisão). Atualmente existem diversas

[2]GAlib – http://www.lancet.mit.edu/ga/

bibliotecas de software disponíveis para se implementar este tipo de simulação. Uma das mais conhecidas é a ODE[3] (*Open Dynamics Engine*), descrita na próxima seção.

4.2 Biblioteca ODE

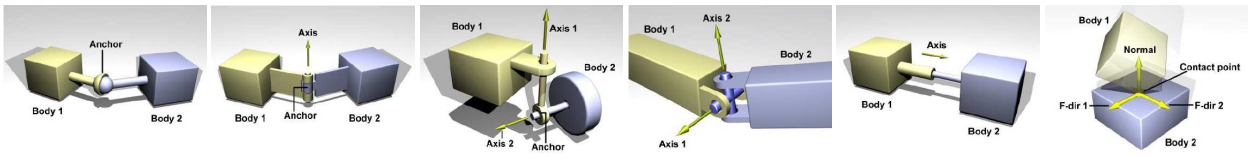
A ODE (*Open Dynamics Engine*), desenvolvida por Russel Smith [42], é uma biblioteca de software livre (*freeware* e *open source*) especialmente desenvolvida para a simulação da dinâmica de corpos rígidos articulados. Ela permite a criação de uma estrutura articulada, através da conexão de corpos rígidos de diversas formas utilizando articulações de vários tipos. A ODE foi projetada para ser utilizada de modo interativo e em simulações de tempo real, e é especialmente indicada para a simulação de objetos móveis em ambientes dinâmicos. Além disto, o usuário tem liberdade para mudar a estrutura do sistema, até mesmo durante a simulação. A ODE utiliza um integrador de primeira ordem altamente estável, que evita que os erros de simulação cresçam de forma descontrolada. Isto permite que a ODE seja rápida, robusta e estável [34].

A simulação é baseada em um método no qual as equações de movimento são derivadas através de um modelo de velocidades baseado em multiplicadores de Lagrange [43]. A ODE possui juntas do tipo contato, que permitem a utilização de restrições de não-penetração sempre que dois corpos rígidos colidem. A ODE possui um sistema de detecção de colisões nativo, que suporta as seguintes primitivas de colisão: *sphere* (esfera), *box* (caixa), *capped cylinder* (cilindro com as extremidades arredondadas) e *plane* (plano, superfície). Outras características da ODE são a distribuição de massa arbitrária aos corpos rígidos, e um modelo de fricção/contato baseado no *Dantzig LCP solver* [1].

A ODE possui uma API (*Aplication Programming Interface*), escrita em linguagem de programação C (embora a ODE tenha sido desenvolvida principalmente em C++), e algumas otimizações específicas para diferentes plataformas. Uma simulação ODE típica ocorre da seguinte forma [43]:

1. Criação do mundo dinâmico;
2. Criação dos corpos rígidos no mundo dinâmico;
3. Ajuste do estado (posição e inclinação) dos corpos rígidos;
4. Criação das articulações no mundo dinâmico;
5. Conexão das articulações aos corpos rígidos;
6. Ajuste dos parâmetros de todas as articulações;
7. Criação do mundo colisivo e dos objetos geométricos neste mundo;
8. Criação de um grupo de articulações para armazenar as juntas do tipo contato;
9. Repetir:
   (a) Aplicação de forças aos corpos conforme a necessidade;
   (b) Ajuste dos parâmetros das articulações conforme a necessidade;
   (c) Execução da rotina de detecção de colisões;
   (d) Criação de juntas do tipo contato para todos os ponto de colisão;
   (e) Execução de um passo da simulação;
   (f) Remoção de todas as juntas do tipo contato;
10. Destruição do mundo dinâmico e do mundo colisivo.

Do ponto de vista físico, um robô é simplesmente um conjunto de corpos rígidos conectados através de diversas articulações, também chamadas de juntas. Cada um destes corpos pode interagir com os demais: uma força (ou torque) aplicada a um destes corpos também afeta os demais corpos conectados a este. Além disto, todos os corpos rígidos devem sofrer a ação da gravidade [34].



(a) Ball Socket (b) Hinge (c) Hinge-2 (d) Universal (e) Slider (f) Contact

Figure 2: Articulações disponíveis na ODE [37]

[3]ODE – http://www.ode.org

Os tipos de juntas implementados na ODE são *ball-and-socket* (similar ao nosso ombro), *hinge* (dobradiça ou joelho), *hinge-2*, *fixed* (fixa), *prismatic slider* (deslizante) e *angular motor* (motores angulares). As juntas do tipo *hinge-2* são iguais a duas dobradiças ligadas em série com diferentes eixos de rotação. A Figura 2 ilustra estas articulações.

Desta forma, a ODE consegue tornar o ambiente virtual bastante realístico, pois os robôs simulados não são apenas figuras geométricas, mas interagem com o ambiente de forma coerente com as leis da física. Este realismo do ponto de vista físico é essencial nas pesquisas em robótica, onde o objetivo final é a construção de robôs reais.

5 Modelo proposto

O simulador LegGen [15, 16] é um simulador desenvolvido para realizar a configuração do caminhar em robôs simulados dotados de pernas de forma automática. Ele foi implementado utilizando a linguagem de programação C++ e as bibliotecas de software ODE e GAlib, descritas anteriormente. Este simulador recebe como entrada dois arquivos, um descrevendo o formato e as dimensões do robô e o outro descrevendo os parâmetros de simulação. A Tabela 1 mostra os parâmetros utilizados pelo simulador LegGen, com os valores utilizados nas simulações.

Table 1: Parâmetros do LegGen

Parâmetro	**Valor**
Taxa de cruzamentos	0,80
Taxa de mutação	0,08
Tamanho da população	350
Número de gerações	700
Tempo de caminhada	30

Os parâmetros *taxa de cruzamentos*, *taxa de mutação*, *tamanho da população* e *número de gerações* são usados diretamente pela biblioteca GAlib para definir o funcionamento do algoritmo genético. O parâmetro *tempo de caminhada* define o tempo em que cada indivíduo irá caminhar durante a avaliação do *fitness* (este tempo é relativo ao relógio da simulação, e não ao tempo do mundo real).

O funcionamento do simulador LegGen ocorre da seguinte forma: inicialmente o arquivo que descreve o robô é lido, e o robô é criado no ambiente virtual de acordo com as especificações presentes neste arquivo. Em seguida, os parâmetros do simulador LegGen são lidos (Tabela 1), e o algoritmo genético é inicializado e executado até que seja atingido o número de gerações desejado. A avaliação dos indivíduos ocorre da seguinte forma:

- O robô é colocado na orientação e na posição inicial do ambiente virtual;
- O genoma é lido, e a partir dele são configurados os pesos sinápticos da rede neural;
- A simulação física é realizada por um tempo determinado (30 segundos);
- Durante a simulação física, são coletadas diversas informações sensoriais;
- O *fitness* é calculado e retornado para a GAlib.

Para o cálculo da função de *fitness*, as seguintes informações sensoriais precisam ser calculadas: a distância percorrida pelo robô (D); o índice dos *bumpers* (B); e a taxa de instabilidade (G); A distância D é calculada pela fórmula:

$$D = Px_1 - Px_0 \tag{1}$$

onde D é a distância percorrida pelo robô em relação ao eixo x (movimento para frente em linha reta), Px_0 é a posição inicial e Px_1 é a posição final em relação ao eixo x. O índice dos *bumpers* B é calculado através da equação:

$$B = \sum_{i=1}^{P} \left(\frac{n_i}{N} - \frac{1}{2} \right)^2 \tag{2}$$

onde P é o número de *endpoints* (patas), n_i é a quantidade de amostras sensoriais nas quais o *endpoint* i estava em contato com o solo, e N é o número total de leituras sensoriais realizadas. Nesta função de *fitness*, o valor de B tenderá a zero quando o robô mantiver as patas no chão por aproximadamente 50% do tempo, que é o comportamento desejado durante o caminhar. Já se o robô mantiver todas as patas no chão durante

o período de simulação, o valor de B será igual a 1. O mesmo ocorrerá se o robô mantiver todas as patas no ar durante o período de simulação.

A taxa de instabilidade é calculada usando a variação das posições do robô nos eixos x, y e z. Esta variações são coletadas durante a simulação física, simulando um sensor do tipo giroscópio/acelerômetro, que é um sensor que está presente em alguns modelos de robôs [8]. A taxa de instabilidade G (Gyro) é calculada através da equação [13]:

$$G = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2 + \sum_{i=1}^{n}(y_i - \overline{y})^2 + \sum_{i=1}^{n}(z_i - \overline{z})^2}{n}} \tag{3}$$

onde n é o número de amostras coletadas, x_i, y_i e z_i são os dados coletados pelo giroscópio simulado no instante i, e $\overline{x}_x$, $\overline{x}_y$ e $\overline{x}_z$ são as médias das leituras realizadas pelo giroscópio, calculadas através das equações:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \overline{z} = \frac{\sum_{i=1}^{n} z_i}{n} \tag{4}$$

A função de *fitness* F é então calculada através da fórmula:

$$F = \frac{D}{1 + B + a \times G} \tag{5}$$

onde a é uma constante que serve para alterar a influência de G na função de *fitness*. Nos experimentos realizados, foi utilizado $a = 10$. Durante a simulação, se um robô afastar as quatro patas do chão ao mesmo tempo por mais de um segundo, a simulação deste indivíduo será imediatamente interrompida, pois é muito provável que este robô tenha sofrido alguma queda, e portanto não há necessidade de continuar a simulação deste indivíduo até o final do tempo estabelecido.

Pela análise desta função de *fitness*, se observa que o indivíduo mais bem qualificado é aquele que possui a melhor relação entre velocidade e instabilidade, de forma que as melhores soluções são aquelas mantém o compromisso entre estes dois critérios de avaliação [13].

5.1 Controle das juntas do robô

No simulador LegGen, o controle das juntas do robô é realizado através de redes neurais artificiais (*artificial neural networks* – ANN) [11, 14]. Esta abordagem possui uma limitação: não é possível se obter de antemão informações locais para o cálculo do gradiente e a correção dos erros, o que limita a utilização dos algoritmos de aprendizado supervisionado tradicionais (*backpropagation* e similares) [18]. Por isto, foram utilizados algoritmos genéticos para a evolução dos pesos sinápticos. As principais vantagens de se utilizar uma ANN no controle do caminhar são [14]: (i) são robustas em relação a situações novas e inesperadas; (ii) possuem um alto grau de generalização.

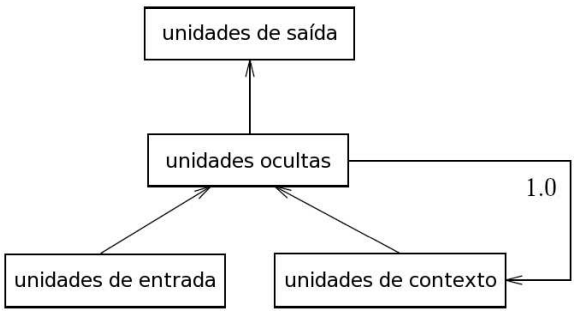
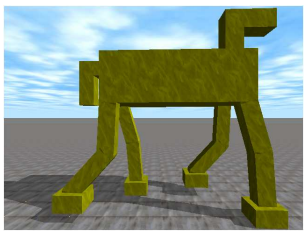


Figure 3: Rede neural recorrente do tipo Elman

Como entradas da ANN, foram utilizados os ângulos atuais das juntas do robô, normalizados entre -1 (α_{mim}) e 1 (α_{max}). Na saída da ANN, são obtidos os ângulos desejados para as juntas no instante $t + 1$, normalizados entre -1 e 1. Após alguns testes preliminares, optou-se por utilizar as redes neurais recorrentes do tipo Elman [10] (Figura 3), que são redes neurais do tipo *multi layer Perceptron* (MLP) que possuem conexões de realimentação (*feedback*) na camada oculta Estas conexões permitem que as redes de Elman aprendam a reconhecer e gerar padrões temporais. A função de ativação utilizada foi a tangente hiperbólica. O intervalo de valores possíveis para os pesos sinápticos foi limitado em $[-1, 1]$, que se mostrou bastante adequado para o problema em questão.

5.2 Robô modelado

Conforme consta em sua documentação, a biblioteca ODE possui uma complexidade computacional de ordem $O(n^2)$, onde n é o número de corpos presentes no mundo físico simulado. Deste modo, para manter a velocidade da simulação em um nível aceitável, é preciso modelar os corpos da forma mais simples possível. Por este motivo, todos o robô simulado foi modelado com objetos simples, como retângulos e cilindros, e possui apenas as articulações necessárias para a tarefa de caminhar. Para manter o projeto do robô simples, as juntas utilizadas nos membros se movimentam apenas em torno do eixo z em relação ao robô (o mesmo movimento do nosso joelho), pois as simulações realizadas até o momento foram todas com o robô a caminhando em linha reta. No futuro, o modelo será estendido para aceitar modelos de robôs com juntas mais complexas. Inicialmente foram modelados e testados diversos tipos de robôs, até que se chegou ao modelo final, mostrado na Figura 4. A dimensões deste robô são aproximadamente as de um cachorro.



	Dimensões		
Parte	x	y	z
Corpo	45,0cm	15,0cm	25,0cm
Coxa	5,0cm	15,0cm	5,0cm
Canela	5,0cm	15,0cm	5,0cm
Pata	8,0cm	5,0cm	9,0cm

Figure 4: Robô modelado

5.3 Evolução da morfologia

Segundo Pfeifer [35], na natureza a evolução do controle (sistema nervoso) não ocorre após a morfologia do corpo estar completa. Pelo contrário, este é um processo que ocorre em conjunto ao longo da evolução. Esta estratégia é muito utilizada na área de vida artificial [40, 41, 9].

Na seção anterior, foi descrito o modelo de robô a ser utilizado nos experimentos. Este robô foi modelado de forma empírica, inspirado em animais de quatro patas, com algumas simplificações. Ao se realizar a co-evolução da morfologia e dos parâmetros de controle, é possível que sejam descobertos novos modelos de robôs, sem equivalentes na natureza, que se mostrem mais eficientes na tarefa em questão [35, 33].

Assim, o simulador LegGen original foi estendido, de forma que permitisse a evolução da morfologia em conjunto com os parâmetros de controle do robô. Para que isto fosse possível, novos genes foram incluídos no genoma original do algoritmo genético. Cada segmento do robô foi codificado utilizando três valores (dimensões em x, y e z).

6 Resultados

Esta seção descreve diversos experimentos realizados utilizando o protótipo do modelo proposto. Nestes experimentos, os parâmetros descritos na Tabela 1 foram utilizados pelo GA. A Tabela 2 mostra os resultados obtidos nestes experimentos. Os valores entre a 2ª e a 5ª coluna (Somente controle) são relativos aos experimentos realizados evoluindo somente os parâmetros de controle, e os valores entre a 6ª e a 9ª coluna (Morfologia e controle) são relativos aos experimentos realizados evoluindo a morfologia em conjunto com os parâmetros de controle. Os gráficos da Figura 5 mostram a média e o intervalo de confiança (*Confidence Interval* – CI) a 95% dos valores do *fitness* (Figura 5(a)), da distância percorrida (Figura 5(b)) e da taxa de instabilidade (Figura 5(c)).

Percebe-se claramente que a evolução da morfologia em conjunto com os parâmetros de controle produziu melhores resultados do que a evolução dos parâmetros de controle de forma isolada. Assim, a evolução da morfologia pode ser considerada uma ferramenta bastante útil no projeto de robôs mais eficientes.

A Figura 6(a) mostra as morfologias que evoluíram ao final das 700 gerações, para cada experimento controlado pela tabela de ângulos. Os números no canto superior esquerdo se referem ao experimento no qual a morfologia do robô foi evoluída. A Figura 7(a) mostra a caminhada de um dos modelos de robô evoluídos[4], no caso o robô do experimento 06, e a Figura 7(b) mostra um exemplo de caminhar evoluído

[4]Vários vídeos demonstrando o caminhar dos robôs estão disponíveis em http://www.inf.unisinos.br/~osorio/leggen

Table 2: Resultados obtidos nos experimentos realizados

	Somente controle				Morfologia e controle			
E	F	D	B	G	F	D	B	G
1	16.265	29.189	0.0019	0.0793	18.802	38.028	0.0105	0.1012
2	16.635	28.306	0.0070	0.0695	17.903	32.959	0.0940	0.0747
3	16.991	27.850	0.0085	0.0631	19.839	39.525	0.0057	0.0987
4	16.678	27.915	0.0017	0.0672	17.801	37.858	0.0017	0.1125
5	16.157	28.201	0.0076	0.0738	20.093	27.409	0.0572	0.0307
6	15.965	31.126	0.0218	0.0928	15.902	32.802	0.0091	0.1054
7	17.335	29.629	0.0048	0.0704	18.869	41.132	0.0134	0.1167
8	16.654	29.038	0.0060	0.0738	18.498	36.218	0.0074	0.0951
9	16.289	30.151	0.0022	0.0849	19.078	39.162	0.0023	0.1050
10	16.227	29.808	0.0039	0.0833	15.572	37.397	0.0057	0.1396
μ	**16.520**	**29.121**	**0.0065**	**0.0758**	**18.235**	**36.249**	**0.0207**	**0.0979**
σ	**0.420**	**1.075**	**0.0059**	**0.0091**	**1.504**	**4.096**	**0.0304**	**0.0288**

para o modelo de robô da Figura 4.

A Figura 6(b) mostra as alterações da morfologia de um robô durante a evolução. Os números no canto superior esquerdo se referem a geração na qual cada modelo de robô se tornou dominante. Percebe-se que a morfologia vai aos poucos sendo melhorada, até que se chegue a um modelo de robô parecido com o da Figura 4. Cabe ressaltar que o espaço de estados permite que apareçam soluções diferentes entre si, mas igualmente eficientes, a exemplo do que ocorre nos seres vivos.

7 Conclusões e perspectivas

O objetivo deste artigo foi descrever o simulador LegGen, que é um simulador desenvolvido para realizar a configuração automática do caminhar de robôs móveis com pernas. Neste simulador, a configuração do caminhar é realizada utilizando algoritmos genéticos, que evoluem os parâmetros do caminhar (pesos sinápticos de uma rede neural) através de robôs simulados em um ambiente virtual realista. Além disso, o protótipo do modelo proposto permite a evolução da morfologia do robô em conjunto com os parâmetros de controle. Os resultados obtidos demonstram que: (i) as redes neurais artificiais são bastante eficientes para o controle das juntas do robô; (ii) o uso de algoritmos genéticos para a evolução dos pesos sinápticos surge como uma alternativa bastante atrativa de se resolver o problema da falta de informações locais para a correção dos erros e do cálculo do gradiente; (iii) a evolução da morfologia do robô em conjunto com os parâmetros de controle permite que sejam descobertos modelos de robôs mais eficientes do que os modelos projetados manualmente. As perspectivas futuras incluem tornar o caminhar possível em superfícies irregulares e a subida de escadas, bem como a construção de um robô físico conforme as especificações de um dos melhores modelos aprendidos, para assim validar o modelo em condições reais.

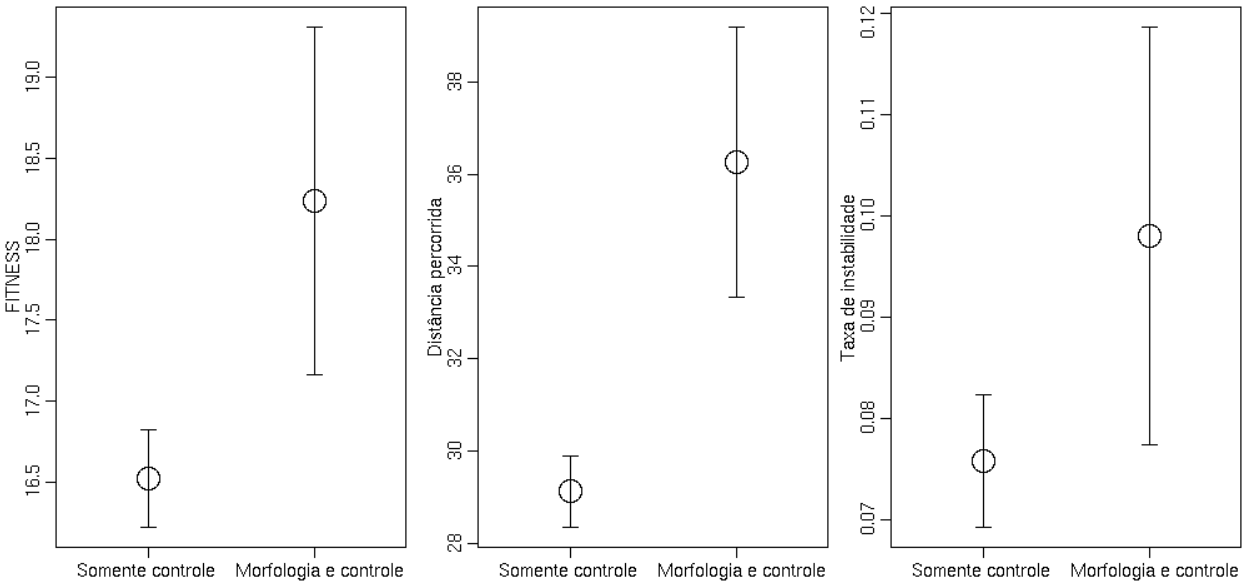
References

[1] Baraff, D., and Witkin, A. Physically based modeling: Principles and practice. Online siggraph '97 course notes, Carnegie Mellon University (CMU), Pittsburgh, CA, USA, 1997.

[2] Bekey, G. A. *Autonomous Robots: From Biological Inspiration to Implementation and Control.* The MIT Press, Cambridge, MA, USA, 2005.

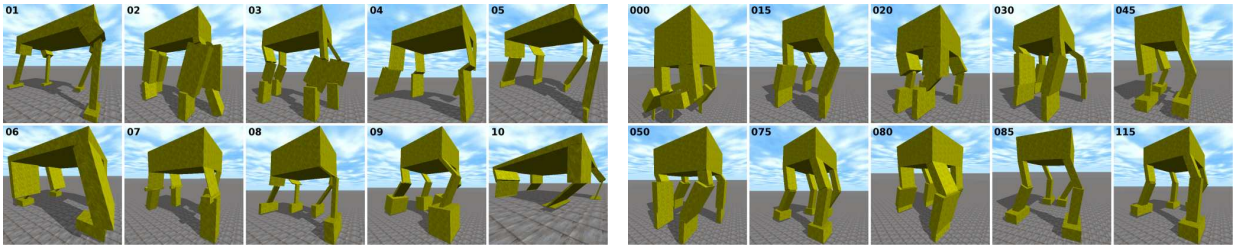
[3] Bongard, J. C., and Pfeifer, R. A method for isolating morphological effects on evolved behaviour. In *Proceedings of the 7th International Conference on Simulation of Adaptive Behaviour (SAB)* (Edinburgh, UK, Aug. 2002), The MIT Press, pp. 305–311.

[4] Busch, J., Ziegler, J., Aue, C., Ross, A., Sawitzki, D., and Banzhaf, W. Automatic generation of control programs for walking robots using genetic programming. In *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)* (Kinsale, Ireland, Apr. 2002), E. Lutton, J. A. Foster,



(a) Valores de fitness (b) Distância percorrida (c) Taxa de instabilidade

Figure 5: Gráficos que mostram a média dos resultados e intervalo de confiança a 95%



(a) Morfologias evoluídas (b) Progresso da evolução

Figure 6: Exemplos de morfologias evoluídas ao final dos experimentos

J. Miller, C. Ryan, and A. G. B. Tettamanzi, Eds., vol. 2278 of *Lecture Notes in Computer Science*, EvoNet, Springer-Verlag, pp. 258–267.

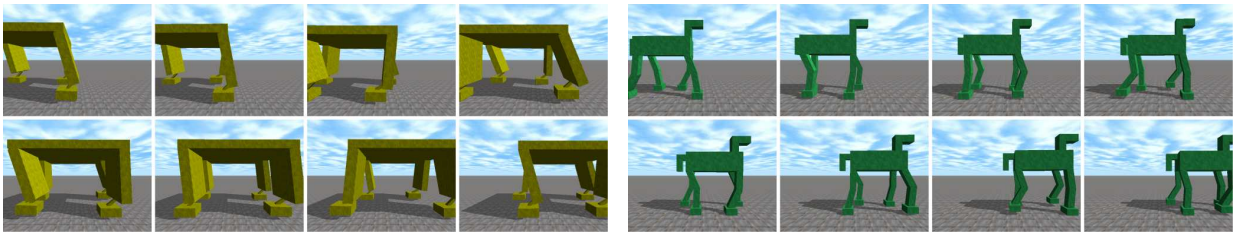
[5] Chernova, S., and Veloso, M. An evolutionary approach to gait learning for four-legged robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Sendai, Japan, Sept. 2004), IEEE Press.

[6] Darwin, C. *Origin of Species.* John Murray, London, UK, 1859.

[7] De Jong, K. A. *An Analysis of the Bahavior of a Class of Genetic Adaptative Systems.* Doctoral thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[8] Dudek, G., and Jenkin, M. *Computational Principles of Mobile Robotics.* Cambridge University Press, Cambridge, UK, 2000.

[9] Eggenberger, P. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life (ECAL'97)* (Cambridge, MA, USA, 1997), P. Husbands and I. Harvey, Eds., The MIT Press, pp. 205–213.



(a) Caminhar do robô evoluído no experimento 06

(b) Caminhar do robô modelado empiricamente

Figure 7: Exemplos de formas de caminhar evoluídas

[10] Elman, J. L. Finding structure in time. *Cognitive Science 14* (1990), 179–211.

[11] Freeman, J. A., and Skapura, D. M. *Redes Neuronales: Algoritmos, Aplicaciones y Técnicas de Programación*. Addison-Wesley / Diaz de Santos, 1993.

[12] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, USA, 1989.

[13] Golubovic, D., and Hu, H. Ga-based gait generation of sony quadruped robots. In *Proceedings of the 3th IASTED International Conference on Artificial Intelligence and Applications (AIA)* (Benalmadena, Spain, Sept. 2003), International Association of Science and Technology for Development (IASTED).

[14] Haykin, S. *Redes Neurais: Princípios e Prática*, 2 ed. Bookman, Porto Alegre, RS, Brazil, 2001.

[15] Heinen, M. R. Controle inteligente do caminhar de robôs móveis simulados. Master's thesis - applied computing, Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, RS, Brazil, 2007.

[16] Heinen, M. R., and Osório, F. S. Applying genetic algorithms to control gait of physically based simulated robots. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)* (Vancouver, Canada, July 2006), IEEE World Congress on Computational Intelligence (WCCI), IEEE Press.

[17] Heinen, M. R., and Osório, F. S. Gait control generation for physically based simulated robots using genetic algorithms. In *Proceedings of the International Joint Conference 2006, 10th Ibero-American Conference on AI (IBERAMIA), 18th Brazilian Symposium on AI (SBIA)* (Ribeirão Preto - SP, Brazil, Oct. 2006), J. Schiman, H. Coelho, and S. O. Rezende, Eds., Lecture Notes in Computer Science, International Joint Conference IBERAMIA/SBIA/SBRN 2006, Springer-Verlag.

[18] Heinen, M. R., and Osório, F. S. Neural networks applied to gait control of physically based simulated robots. In *Proceedings of the International Joint Conference 2006, 9th Brazilian Neural Networks Symposium (SBRN)* (Ribeirão Preto, SP, Brazil, Oct. 2006), A. M. P. Canuto, M. C. P. de Souto, and A. C. R. da Silva, Eds., International Joint Conference IBERAMIA/SBIA/SBRN 2006, IEEE Press.

[19] Heinen, M. R., and Osório, F. S. Evolving gait control of physically based simulated robots. *Revista de Informática Teórica e Aplicada (RITA) – to appear* (2007).

[20] Heinen, M. R., Osório, F. S., Heinen, F. J., and Kelber, C. SEVA3D: Using artificial neural networks to autonomous vehicle parking control. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)* (Vancouver, Canada, July 2006), IEEE World Congress on Computational Intelligence (WCCI), IEEE Press.

[21] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[22] Jacob, D., Polani, D., and Nehaniv, C. L. Legs than can walk: Embodiment-based modular reinforcement learning applied. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)* (Espoo, Finland, June 2005), IEEE Press, pp. 365–372.

[23] Knight, R., and Nehmzow, U. Walking robots - a survey and a research proposal. Technical Report CSM-375, University of Essex, Essex, UK, 2002.

[24] Kohl, N., and Stone, P. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (New Orleans, LA, USA, Apr. 2004), IEEE Press, pp. 2619–2624.

[25] Langton, C. *Arificial Life: An Overview.* The MIT Press, Cambridge, MA, USA, 1995.

[26] Law, A. M., and Kelton, D. W. *Simulation Modeling and Analysis.* McGraw-Hill, New York, USA, 2000.

[27] Levy, S. *Arificial Life: A Report From the Frontier Where Computers Meet Biology.* Vintage books, New York, USA, 1992.

[28] Lewis, M. A., Fagg, A. H., and Solidum, A. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Nice, France, 1992), IEEE Press, pp. 2618–2623.

[29] McGhee, R. B. Robot locomotion. *Neural Control of Locomotion* (1976), 237–264.

[30] Medeiros, A. Introdução à robótica. In *Anais do XVII Encontro Nacional de Automática* (Natal, RN, Brazil, 1998), vol. 1, 50a Reunião Anual da SBPC, pp. 56–65.

[31] Mitchell, M. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, USA, 1996.

[32] Mitchell, T. *Machine Learning.* McGrall-Hill, New York, USA, 1997.

[33] Nolfi, S., and Floreano, D. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* The MIT Press, Cambridge, MA, USA, 2000.

[34] Osório, F. S., Musse, S. R., Vieira, R., Heinen, M. R., and Paiva, D. C. *Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation*, vol. 2. Springer-Verlag, Berlin, Germany, 2006, pp. 1–45.

[35] Pfeifer, R., and Scheier, C. *Understanding Intelligence.* The MIT Press, Cambridge, MA, USA, 1999.

[36] Raibert, M. H. *Legged Robots That Balance.* The MIT Press, Cambridge, MA, USA, 1986.

[37] Reeve, R., and Hallam, J. An analysis of neural models for walking control. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 733–742.

[38] Reeve, R. E. *Generating Walking Behaviours in Legged Robots.* Ph.d. thesis, University of Edinburgh, Edinburgh, Scotland, Dec. 1999.

[39] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. *Learning Internal Representations by Error Propagation.* The MIT Press, Cambridge, MA, USA, 1986.

[40] Sims, K. Evolving 3d morphology and behavior by competition. In *Artificial Life IV Proceedings* (Cambridge, MA, USA, 1994), R. A. Brooks and P. Maes, Eds., The MIT Press, pp. 28–39.

[41] Sims, K. Evolving virtual creatures. *Computer Graphics 28* (July 1994), 15–24.

[42] Smith, R. Open dynamics engine v0.5 user guide. Last Visited: 23/02/2006, Feb. 2006.

[43] Wolff, K., and Nordin, P. Evolutionary learning from first principles of biped walking on a simulated humanoid robot. In *Proceedings of the Business and Industry Symposium of the Advanced Simulation Technologies Conference (ASTC'03)* (Orlando, FL, USA, Apr. 2003), M. Ades and L. M. Deschaine, Eds., SCS, pp. 31–36.