Co-Evolução da Morfologia e Controle de Robôs Móveis Simulados Utilizando Realidade Virtual

OMMITED FOR BLIND REVIEW

ABSTRACT

O objetivo deste artigo é exemplificar o uso da Realidade Virtual na área de Robótica Autônoma. Para isto, será descrito o simulador LegGen, que realiza a configuração automática do caminhar em robôs móveis com pernas, através do uso de Algoritmos Genéticos e simulação baseada em física. No simulador LegGen, os robôs simulados são utilizados para a evolução dos parâmetros do Algoritmo Genético, o que reduz bastante os tempos necessários para a evolução dos parâmetros, reduzindo assim os custos de desenvolvimento. Diversos experimentos foram realizados utilizando robôs virtuais de quatro e seis patas, e os resultados obtidos tornam possível a construção de um robô real utilizando o modelo de controle evoluído usando robô virtual. Além disto, são descritos diversos experimentos realizados evoluindo a morfologia do robô em conjunto com os parâmetros de controle.

Keywords

Realidade Virtual, Simulação Baseada em Física, Robótica Autônoma, Algoritmos Genéticos, Vida Artificial

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 [27]. 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 [2]. 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, pois estes ambientes possuem diversos desníveis e degraus [19]. Assim, para que um robô móvel possa se deslocar livremente em ambientes irregulares, ele precisaria ser dotado do mesmo mecanismo de locomoção utilizado pelos seres humanos e a maioria dos seres vivos, ou seja, ele precisaria de pernas [3, 31].

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 [8]. Desta forma, seria interessante realizar a configuração do caminhar de forma automática, através da utilização de alguma técnica de Aprendizado de Máquina [29]. Uma das técnicas de Aprendizado de Máquina mais adequadas para solucionar este tipo de problema são os Algoritmos Genéticos (AG) [12], pois segundo a Teoria da Evolução Natural das Espécies [9], os mecanismos de locomoção utilizados pelos seres vivos são um resultado da Evolução Natural, o que torna o uso de Algoritmos Genéticos uma solução natural para este tipo de problema [31]. Do ponto de vista computacional, os Algoritmos Genéticos também são bastante adequados para a configuração do caminhar em robôs com pernas, 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 [28].

Mas apesar de serem bastante adequados para a configuração automática do caminhar em robôs com pernas, os Algoritmos Genéticos não são muito eficientes se forem utilizados diretamente em robôs reais. Isto ocorre porque os Algoritmos Genéticos 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 [38]. 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 [8].

Desta forma, o uso de Realidade Virtual surge como uma alternativa viável para tornar o uso de Algoritmos Genéticos em robótica móvel mais eficiente, pois ao invés dos experimentos serem realizados diretamente em um robô real, eles podem ser realizados em um robô simulado, e somente após

o término do aprendizado é que a configuração aprendida é portada para o robô real, o que economiza muitas horas de treinamento e evita o desgaste dos equipamentos [38]. O objetivo deste artigo é descrever o simulador LegGen, que é um simulador capaz de realizar a configuração automática do caminhar em robôs com pernas. No simulador LegGen, os robôs são simulados dentro de um ambiente virtual bastante realista, no qual as leis da física foram implementadas através da biblioteca Open Dynamics Engine (ODE), que uma biblioteca de software baseada em C++ especialmente desenvolvida para a realização de simulações baseadas em física.

O objetivo deste artigo é descrever o simulador LegGen [4], que é um simulador capaz de realizar a configuração do caminhar de robôs com pernas de forma automática através do uso de Algoritmos Genéticos. Este artigo está estruturado da seguinte forma: A Seção 2 descreve diversos trabalhos do estado da arte na área de robótica móvel; A Seção 3 descreve a área de Vida Artificial, que possui relação com o problema em questão; A Seção 4 descreve os Algoritmos Genéticos e a biblioteca de software GAlib, que foi utilizada nas simulações; A Seção 5 descreve o uso de simulação baseada em física e a biblioteca ODE; A Seção 6 descreve diversos conceitos relativos aos robôs com pernas; A Seção 7 descreve o modelo proposto, chamado de LegGen, e os robôs utilizados nas simulações; A Seção 8 descreve os experimentos realizados e os resultados obtidos; e por último a Seção 9 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 [3]. 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) [20]. Esta é uma área de pesquisas que tem 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 [33].

O controle do caminhar em robôs com pernas é uma área de pesquisas que vem sendo explorada a bastante tempo. Como trabalhos pioneiros nesta área podemos destacar os primeiros robôs com pernas realmente independentes, como o "Phony Pony" desenvolvido por Frank e McGhee [26], onde cada junta foi controlada por uma simples máquina de estados finita, até o muito bem sucedido controle algorítmico de bípedes e quadrúpedes desenvolvido por Raibert [32].

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 [24], 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 [5] evoluiu os parâmetros de uma Rede Neural Artificial dinâmica utilizada para controlar diversos tipos de robôs simulados. Busch [7] utilizou Programação Genética para evoluir os parâmetros de controle de diversos tipos de robôs, simulados utilizando o pacote de software DynaMechs[1]. Jacob [18] utilizou Aprendizado por Reforço para o controle de um robô de quatro pernas (*tetrapod*) simulado através da biblioteca de software ODE. Reeve [33] 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 [4].

3. 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 [21]. 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 [23, 31]. 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 dos princípios biológicos nas áreas de robótica, medicina, nanotecnologia e engenharia.

Um dos pioneiros na área de vida artificial foi Karl Sims, que em [34, 35] 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 [31], 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 (a função de *fitness* é a distância percorrida).

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 [31]. Já em um modelo com muitas restrições haverá menos variabilidade nas criaturas evoluídas.

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

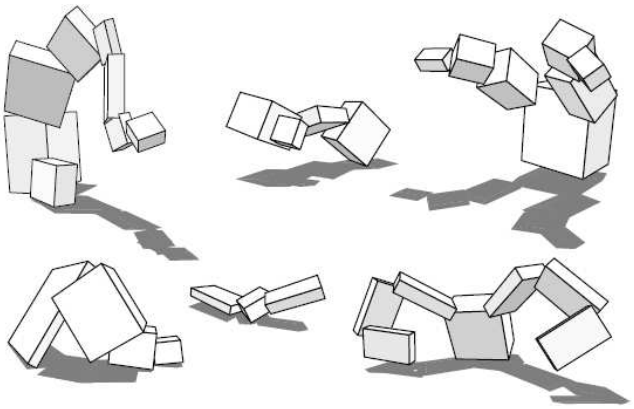


Figure 1: Criaturas virtuais de Sims

4. ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos (AG) são métodos de busca estocástica baseados na Teoria da Evolução Natural das Espécies [9], criados por John Holland nos anos 60 [16]. 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 vá sendo melhorado, até que a solução que atenda aos objetivos desejados [28].

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 na linguagem de programação C++.

No simulador implementado, foi utilizado o algoritmo genético com populações sobrepostas (*overlapping populations*) proposto por [10], e foram adotados genomas do tipo real (números de ponto flutuante). 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 selector*). O método de escala (*scaling*) do *fitness* utilizado foi o *sigma truncation*, que permite que o *fitness* assuma valores negativos.

5. 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 [31]. A utilização de um robô real possui a vantagem de tornar realísticos os resultados obtidos, mas o uso de simulação possui as seguintes vantagens [22]:

- 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.

Para o desenvolvimento de um simulador de robôs móveis, o uso de uma biblioteca de simulação baseada em física é bastante útil, como pode ser visto na próxima seção.

5.1 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 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.

5.2 Biblioteca ODE

A ODE (*Open Dynamics Engine*), desenvolvida por Russel Smith [36], 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.

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 [38]. 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

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

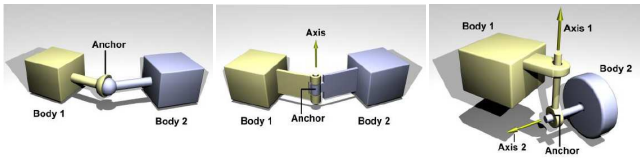
[3]ODE – http://www.ode.org

arbitrária aos corpos rígidos, e um modelo de fricção/contato baseado no *Dantzig LCP solver* [1].

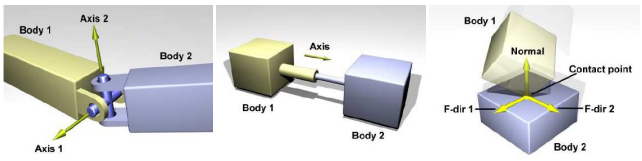
A ODE possui uma API (*Aplication Programming Interface*), escrita em linguagem de programação C (embora a ODE tenha sido desenvolvida principalmente em C++), e algumas otimizações específicas para diferentes plataformas. Uma simulação ODE típica ocorre da seguinte forma [38]:

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.



(a) Ball Socket (b) Hinge (c) Hinge-2



(d) Universal (e) Slider (f) Contact

Figure 2: Articulações disponíveis na ODE

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 mostra as articulações disponíveis na ODE.

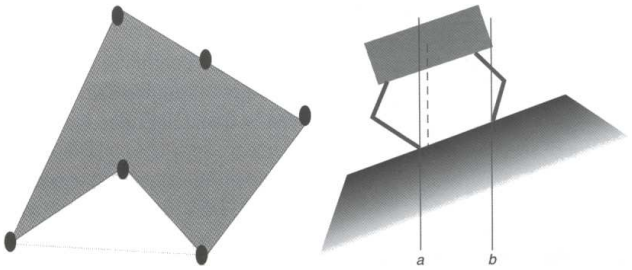
Desta forma, a ODE consegue tornar o ambiente virtual bastante realístico, pois os robôs simulados não são apenas figuras geométricas, mas interagem com o ambiente de forma coerente com as leis da física. Este realismo do ponto de vista físico é essencial nas pesquisas em robótica, onde o objetivo final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

6. ROBÔS COM PERNAS

Nesta seção serão descritos diversos conceitos relativos ao caminhar de robôs com pernas, incluindo os conceitos de estabilidade e a forma de implementação do sistema de controle das juntas.

6.1 Estabilidade

Para que um robô consiga se deslocar livremente sem sofrer quedas, é necessário que o caminhar seja estável, e esta estabilidade pode ser obtida de forma estática ou dinâmica [11]. Quando um robô se desloca de forma que o seu centro de gravidade nunca fique fora do polígono de suporte formado pelas pernas que estão em contato com o solo, é dito que o robô apresenta estabilidade estática. A Figura 3 ilustra esta situação. A Figura 3(a) mostra um exemplo de polígono de suporte formado pelas patas de um robô que estão em contato com o solo (seis patas, neste caso), e a Figura 3(b) mostra o centro de gravidade de um robô sobre o polígono de suporte. A principal vantagem da estabilidade estática é que o risco do robô sofrer quedas é menor, pois as patas que estão em contato com o solo conseguem garantir a estabilidade mesmo no caso de uma falha de energia ou se as baterias se descarregarem.



(a) (b)

Figure 3: Polígono de suporte de um robô

Se durante o caminhar o centro de gravidade do robô se deslocar periodicamente para fora do polígono de suporte, e mesmo assim o robô conseguir se movimentar de forma controlada, é dito que este robô apresenta estabilidade dinâmica. A estabilidade dinâmica é mais difícil de ser atingida, pois exige um sofisticado modelo da dinâmica do robô e do uso da inércia [11, 3].

A estabilidade depende diretamente do número de pernas do robô. Para que um robô apresente estabilidade estática, o número mínimo de pernas necessário são quatro, e o robô precisa se deslocar deixando sempre três patas em contato com o chão. Já um robô de seis pernas consegue apresentar estabilidade estática mantendo apenas a metade de suas patas em contato com o chão, o que faz com que ele possa se deslocar mais rapidamente sem correr o risco de cair.

De acordo com um estudo realizado por Muybridge [30], abordando as formas de caminhar dos cavalos, estes animais possuem oito formas diferentes de caminhar, sendo que apenas uma (a mais lenta) é estável estaticamente. No trote, que é um passo de velocidade moderada, o cavalo afasta do chão duas pernas ao mesmo tempo, e no galope, que é o andar mais rápido, existem momentos em que o cavalo fica com as quatro patas no ar.

6.2 Geração de padrões rítmicos

Os geradores centrais de padrões (*central pattern generator* - CPG) são grupos de neurônios que produzem padrões rítmicos de forma endógena, ou seja, sem a necessidade de estímulos oscilatórios ou de coordenação central. Eles são responsáveis pela produção dos padrões motores rítmicos da maioria dos seres vivos [25, 37]. Uma das principais características dos CPGs é que a geração dos padrões não depende da atuação do sistema nervoso como um todo, mas sim de pequenos grupos de neurônios (os CPGs) que trabalham de forma autônoma [17].

Nos seres vivos, os CPGs utilizados na locomoção se encontram na espinha dorsal. Estes recebem sinais relativamente simples do sistema nervoso central, que controlam a velocidade e a direção do deslocamento [6]. Para a produção dos padrões básicos, geralmente não é necessário *feedback* sensorial, embora ele seja muito importante na adaptação dos padrões de acordo com a situação enfrentada pelo ser vivo [14, 15].

6.3 Geração do caminhar

Em um robô com pernas, o caminhar pode ser gerado de diversas formas. A alternativa utilizada neste artigo consiste em uma máquina de estados, onde cada estado é determina o ângulo desejado para cada uma das juntas do robô. Nesta abordagem, o controlador continuamente realiza a leitura dos ângulos de cada uma das juntas, para verificar se elas já atingiram os valores desejados. Em robôs reais, os ângulos das juntas podem ser obtidos através da leitura de sensores (encoders) instalados nas mesmas [11]. Desta forma, o controle do caminhar é realizado da seguinte forma: inicialmente o controlador verifica se as juntas já atingiram os ângulos desejados. As juntas que não tiverem atingido são movimentadas (os motores são ativados), e quando todas elas tiverem atingido os seus respectivos ângulos, o autômato passa para o estado seguinte. Se alguma das juntas não tiver atingido o ângulo desejado após um limite de tempo, o estado é avançado independente desta.

Para que haja sincronia nos movimentos, é importante que todas as juntas atinjam os ângulos desejados praticamente ao mesmo tempo, o que é possível com a aplicação de uma velocidade angular específica para cada uma das juntas, calculada através da fórmula:

$$V_{ij} = Vr_i(\alpha_{ij} - \alpha_{ij-1}) \quad (1)$$

onde V_{ij} é a velocidade aplicada ao motor da junta i no estado j, α_{ij} é o ângulo da junta i no estado j, α_{ij-1} é o ângulo da junta i no estado anterior $(j-1)$, e Vr_i é a velocidade referencial do estado i, utilizada para controlar a velocidade do conjunto. A velocidade referencial Vr é um dos parâmetros do caminhar otimizados pelo Algoritmo Genético. Os outros parâmetros são os ângulos desejados de cada uma das juntas para cada estado. Para limitar o espaço de busca, o Algoritmo Genético gera apenas valores dentro do intervalo máximo e mínimo de cada junta.

7. MODELO PROPOSTO

O simulador LegGen 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. O simulador LegGen 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	75
Número de gerações	150
Número de estados do autômato	4
Tempo de caminhada	30
Velocidade relativa mínima	0,25
Velocidade relativa máxima	1,75

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 *número de estados do autômato* define o número de estados do autômato finito, 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), e os parâmetros *velocidade relativa mínima* e *velocidade relativa máxima* definem a faixa na qual as velocidades relativas (Vr) serão geradas pelo Algoritmo Genético.

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 é montada a tabela de controle do robô;
- A simulação física é realizada por um tempo determinado (60 segundos);
- Durante a simulação física, são coletadas as 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: distância percorrida pelo robô (D); e taxa de instabilidade (G); A distância D é calculada pela fórmula:

$$D = Px_1 - Px_0 \tag{2}$$

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.

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 [11]. A taxa de instabilidade G (Giroscópio) 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:

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

O *fitness* F é calculado através da fórmula:

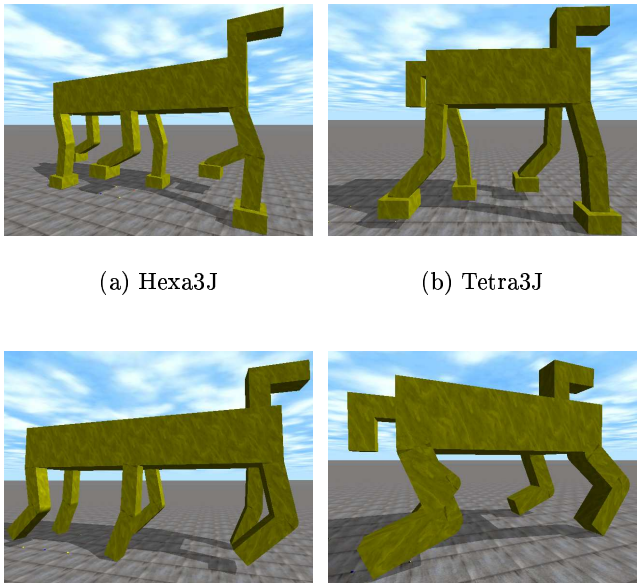
$$F = \frac{D}{1+G} \tag{5}$$

Pela análise da 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].

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.

7.1 Robôs modelados

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 os robôs simulados foram modelados com objetos simples, como retângulos e cilindros, e eles possuem apenas as articulações necessárias para a tarefa de caminhar. Assim, elementos como a cabeça e a cauda não estão presentes em nenhum dos modelos de robôs simulados. Para manter o projeto dos robôs 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 simulador será estendido para aceitar modelos de robôs com juntas mais complexas.



(a) Hexa3J (b) Tetra3J

(c) Hexa2J (d) Tetra2J

Figure 4: Modelos de robôs utilizados

Inicialmente foram modelados e testados diversos tipos de robôs, até que se chegou aos quatro modelos principais, mostrados na Figura 4. O modelo da Figura 4(a), chamado de **Hexa3J**, possui seis pernas e três partes por perna. As partes que entram em contato com o solo (pés ou patas) são mais largas que o restante das pernas, de modo a dar um maior apoio ao robô. O modelo da Figura 4(b), chamado de **Tetra3J**, é similar ao da Figura 4(a), mas possui apenas quatro pernas. Nestes dois robôs, o ângulo das juntas de cada uma das patas (α_p) é calculado através da fórmula:

$$\alpha_p = -\sum_{i=1}^{n} \alpha_i \tag{6}$$

onde α_i é o ângulo da junta i e n é o número de juntas em cada perna. Utilizando a Fórmula 6, as patas do robô ficam sempre paralelas ao solo. O modelo de robô da Figura 4(c), chamado de **Hexa2J**, é similar ao da Figura 4(a), mas possui apenas duas articulações por perna, e todos os ângulos das juntas são calculados pelo Algoritmo Genético, sem utilizar a Fórmula 6. O modelo da Figura 4(d) (**Tetra2J**) possui quatro pernas e duas articulações por perna. A Tabela 2 mostra as dimensões dos robôs em centímetros.

Table 2: Dimensões dos robôs simulados

	HexaL3J	TetraL3J	HexaL2J	TetraL2J
Corpo x	80,0	45,0	80,0	45,0
Corpo y	15,0	15,0	15,0	15,0
Corpo z	30,0	25,0	30,0	25,0
Perna x	5,0	5,0	5,0	5,0
Perna y	15,0	15,0	15,0	15,0
Perna z	5,0	5,0	5,0	5,0
Pata x	8,5	8,5	-	-
Pata y	5,0	5,0	-	-
Pata z	9,0	9,0	-	-

Similar ao que acontece na maioria dos mamíferos de quatro pernas, as patas dos robôs das Figuras 4(a) e 4(b) são um pouco mais largas que o restante das pernas, de forma a dar ao robô uma maior base de sustentação, permitindo assim uma maior estabilidade. A Tabela 3 mostra os limites máximos e mínimos das juntas dos robôs da Figura 4. Os robôs modelados possuem todas as pernas idênticas entre si.

Table 3: Limites das juntas

Robô	*Quadril*	*Joelho*	*Tornozelo*
HexaL3J	[-60°;15°]	[0°;120°]	[-90°;30°]
TetraL3J	[-60°;15°]	[0°;120°]	[-90°;30°]
HexaL2J	[-60°;15°]	[0°;120°]	–
TetraL2J	[-60°;15°]	[0°;120°]	–

8. RESULTADOS

Esta seção descreve diversos experimentos realizados utilizando o protótipo do modelo proposto. Primeiramente serão descritos os experimentos realizados visando determinar o modelo de robô mais eficiente (dentre os modelos mostrados na Figura 4(a)), e em seguida serão descritos alguns experimentos realizados nos quais a morfologia do robô foi evoluída em conjunto com os parâmetros de controle.

8.1 Escolha do modelo de robô

Esta sub-seção descreve os experimentos realizados para determinar qual o modelo de robô, dentre os mostrados na Figura 4, é mais eficiente na tarefa em questão. A idéia é selecionar um robô que consiga caminhar da forma satisfatória utilizando o menor número de juntas possível, pois quanto mais simples for o robô, mais barato se torna a sua construção futura, embora isto esteja fora do escopo deste trabalho. Desta forma, foram realizados dez experimentos distintos com cada modelo de robô. A Tabela 4 mostra os resultados obtidos nos experimentos realizados utilizando os robôs HexaL3J (Figura 4(a)) e TetraL3J (Figura 4(b)).

A primeira coluna (**E**) representa o índice do experimento. As demais colunas representam, respectivamente, os resultados obtidos nos experimentos utilizando os dois modelos de robôs. As sub-colunas desta tabela representam, respectivamente, o valor do *fitness* F, calculado através da Fórmula 5, a distância percorrida pelo robô D em 30 segundos, e a taxa de instabilidade G, calculada através da Fórmula 3. As duas últimas linhas trazem a média (μ) e o desvio padrão (σ) de cada uma das colunas. A Tabela 5 mostra os resultados

Table 4: Escolha do modelo de robô

	HexaL3J			**TetraL3J**		
E	F	D	G	F	D	G
1	13,14	31,6	0,140	14,56	28,6	0,096
2	14,80	30,8	0,107	13,69	34,9	0,154
3	12,32	47,8	0,284	12,82	25,2	0,096
4	15,40	34,6	0,123	16,05	28,0	0,075
5	15,63	32,2	0,105	12,43	31,2	0,150
6	16,67	39,2	0,135	14,11	26,6	0,088
7	12,17	43,9	0,258	13,95	33,3	0,138
8	17,86	29,6	0,065	9,98	22,6	0,126
9	15,51	33,0	0,111	10,50	25,3	0,141
10	16,28	35,3	0,116	11,41	23,5	0,106
μ	**14,98**	**35,8**	**0,145**	**12,95**	**27,9**	**0,117**
σ	**1,89**	**6,0**	**0,070**	**1,90**	**4,1**	**0,028**

obtidos utilizado os robôs HexaL2J (Figura 4(c)) e TetraL2J (Figura 4(d)).

Table 5: Escolha do modelo de robô

	HexaL2J			**TetraL2J**		
E	F	D	G	F	D	G
1	10,30	24,4	0,136	10,11	16,7	0,065
2	15,32	24,9	0,062	5,98	9,9	0,066
3	13,63	29,2	0,113	6,77	22,9	0,236
4	12,61	25,1	0,097	3,89	16,0	0,308
5	16,26	35,2	0,112	8,64	12,4	0,043
6	15,25	26,7	0,070	8,69	14,4	0,066
7	12,28	28,9	0,132	8,71	12,9	0,048
8	10,98	19,6	0,078	4,07	9,9	0,140
9	16,68	23,4	0,040	12,61	27,7	0,118
10	12,33	26,3	0,111	0,86	2,5	0,095
μ	**13,56**	**26,4**	**0,095**	**7,03**	**14,5**	**0,118**
σ	**2,22**	**4,1**	**0,032**	**3,44**	**7,0**	**0,088**

A Figura 5 mostra o gráfico de *boxplot* dos valores de F dos experimentos das Tabelas 4 e 5. Observando-se os gráficos da Figura 5, percebe-se que o modelo de robô que obteve o pior desempenho foi o TetraL2J (Figura 4(d)). A Figura 6 mostra um exemplo de caminhada realizada por este robô, no qual se pode notar que o TetraL2J apresenta sérias dificuldades para se locomover.

Com relação aos demais modelos, embora o desempenho dos robôs de seis pernas (HexaL3J e HexaL2J) seja ligeiramente melhor que o desempenho do robô TetraL3J, a diferença nos resultados não é significativa do ponto de vista estatístico. Assim, o robô selecionado para ser utilizado nos próximos experimentos foi o TetraL3J (Figura 4(b)), pois este consegue se deslocar de forma similar aos demais utilizando apenas quatro pernas, o que o torna mais simples e econômico em relação aos gastos de energia e em relação aos custos de *hardware*. A Figura 7 mostra um exemplo de caminhada realizada pelo robô TetraL3J.

A título de exemplificação, a Figura 8 mostra um exemplo de caminhada realizada pelo robô HexaL3J (Figura 4(a)), e a Figura 9 mostra um exemplo de caminhada realizada pelo robô HexaL2J (Figura 4(c)). Percebe-se que devido ao passo

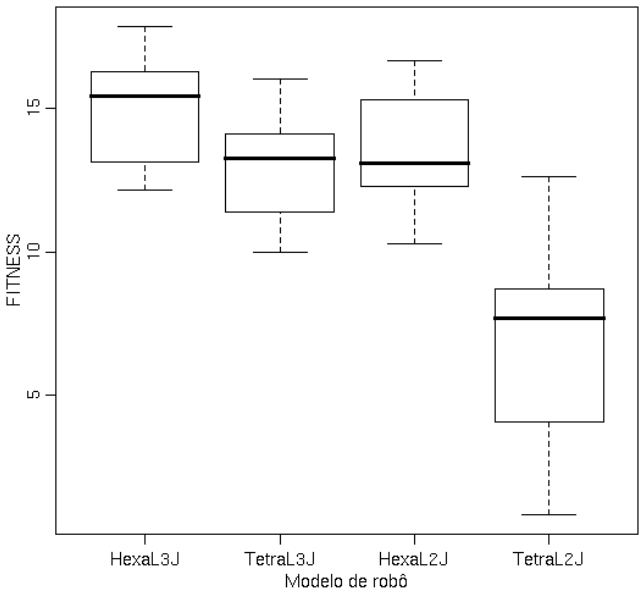


Figure 5: Gráfico de *boxplot*

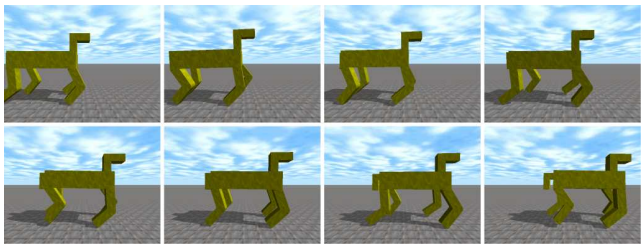


Figure 6: Caminhar realizado pelo robô TetraL2J

tripod (tripé), estes robôs conseguem se deslocar de forma rápida e eficiente, pois eles mantém a estabilidade estática durante o caminhar.

8.2 Evolução da morfologia e do controle

Esta sub-seção descreve alguns experimentos realizados, nos quais a morfologia do robô foi evoluída em conjunto com os parâmetros da estratégia de controle. Nestes experimentos, foram utilizados os mesmos parâmetros da Tabela 1 no GA, com exceção do tamanho da população, que foi aumentado para 150, e do número de gerações, que foi aumentado para 300. A Tabela 6 mostra os resultados obtidos nestes experimentos.

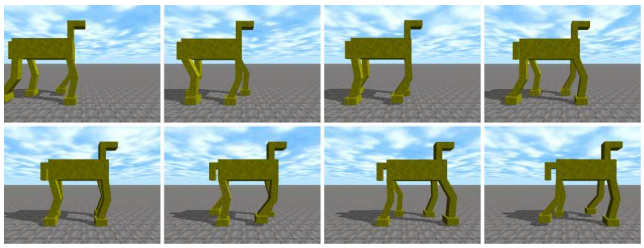


Figure 7: Caminhar realizado pelo robô TetraL3J

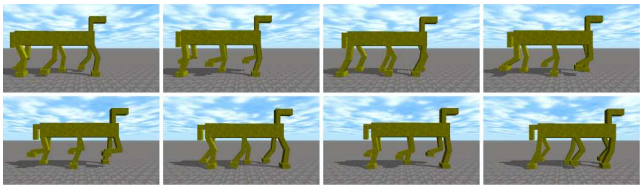


Figure 8: Caminhar realizado pelo robô HexaL3J

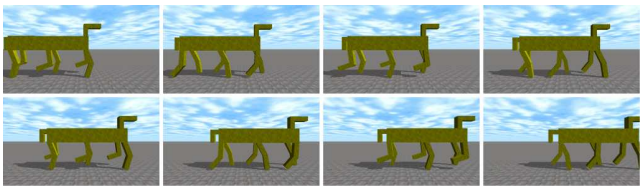


Figure 9: Caminhar realizado pelo robô HexaL2J

Os valores entre a 2ª e a 7ª coluna (Morfologia e controle) são relativos aos experimentos realizados evoluindo a morfologia em conjunto com os parâmetros de controle, e os valores entre a 8ª e a 13ª coluna (Somente controle) são relativos aos experimentos realizados evoluindo somente os parâmetros de controle. A Figura 10 mostra os gráficos de *boxplot* e o intervalo de confiança (*Confidence Interval* – CI) a 95% dos valores de *fitness*.

Percebe-se claramente que em ambas as estratégias a co-evolução da morfologia e do controle produz melhores resultados do que a evolução dos parâmetros de controle de forma isolada, embora utilizando o controlador neural as diferenças não sejam significativas do ponto de vista estatístico. Isto se deve ao fato de que o tamanho do espaço de estados do controlador neural é bem maior (40 pesos sinápticos e 12 genes da morfologia), o que influencia nos resultados.

A Figura 11 mostra as morfologias que evoluíram ao final das 300 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 12 mostra a caminhada de um modelos evoluídos, no caso o robô do experimento 06.

Table 6: Evolução da morfologia e do controle

	Morfologia e controle			Somente controle		
E	F	D	G	F	D	G
1	24,83	56,4	0,127	9,64	28,2	0,190
2	27,28	56,3	0,106	16,01	36,0	0,124
3	24,14	55,3	0,129	12,80	32,0	0,148
4	24,06	97,3	0,304	14,79	26,3	0,077
5	23,35	72,5	0,210	14,86	31,7	0,113
6	20,14	59,2	0,194	16,00	29,1	0,082
7	24,23	71,7	0,195	17,32	29,5	0,070
8	22,51	70,6	0,213	16,55	35,3	0,113
9	27,46	99,5	0,262	11,58	28,9	0,148
10	21,25	42,2	0,098	12,14	34,5	0,184
μ	**23,93**	**68,1**	**0,184**	**14,17**	**31,2**	**0,125**
σ	**2,32**	**18,5**	**0,068**	**2,50**	**3,3**	**0,042**

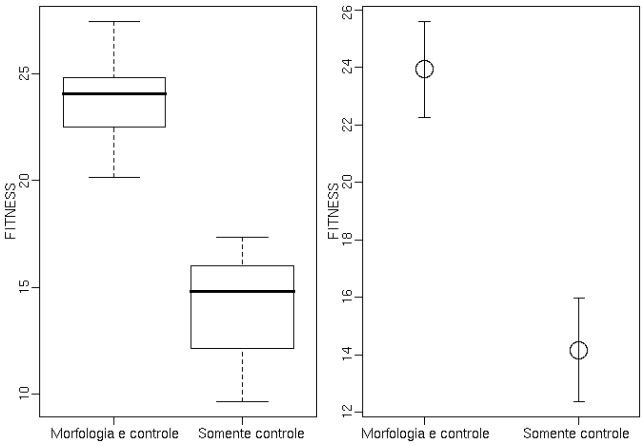


Figure 10: Gráfico de *boxplot* e CI

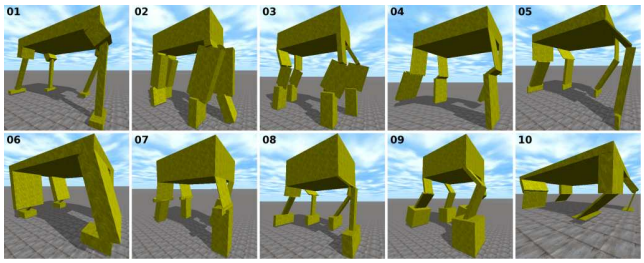


Figure 11: Morfologia final dos experimentos

A Figura 13 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(b). 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.

9. CONCLUSÕES E PERSPECTIVAS

O objetivo deste artigo foi exemplificar o uso de Realidade Virtual no desenvolvimento de sistemas de controle em robótica móvel. Para isto, foi descrito 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 através do uso de robôs simulados em um ambiente virtual que implementa as leis da física através da biblioteca ODE. Além disso, o protótipo do modelo proposto permite a evolução da morfologia do robô em conjunto com os parâmetros de controle.

Utilizando o simulador LegGen, foi possível testar diversas configurações de robôs, e assim foi possível descobrir qual o melhor modelo de robô real a ser construído no futuro. Além disso, o sistema de controle desenvolvido para o robô simulado poderá ser portado para o robô real quando este estiver construído, bastando para isto realizar alguns pequenos ajustes para adaptar o controlador evoluído.

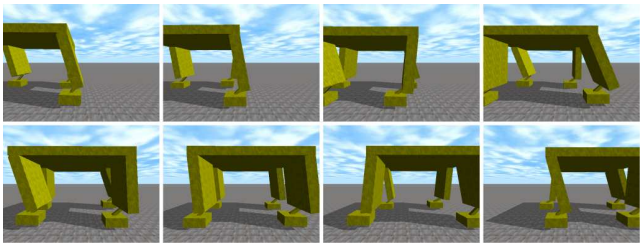


Figure 12: Robô evoluído no experimento 06

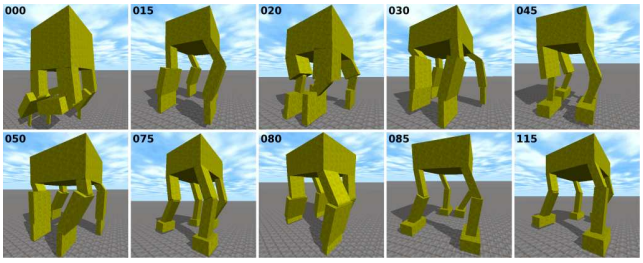


Figure 13: Progresso da evolução da morfologia

Concluímos que a Realidade Virtual é uma ferramenta bastante útil na área de robótica autônoma, pois permite que se reduzam os custos de desenvolvimento e os tempos de configuração, tornado a construção e a pesquisa de robôs móveis muito mais viável de ser executada. As perspectivas futuras incluem a extensão do simulador LegGen para ser utilizado em robôs bípedes e com juntas mais sofisticadas, para assim permitir o desenvolvimento de robôs humanóides.

10. REFERENCES

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

[2] P. Batavia, D. Pomerleau, and C. Thorpe. Applying advanced learning algorithms to alvinn. Technical Report CMU-RI-TR-96-31, Carnegie Mellon University (CMU), Pittsburgh, PA, USA, 1996.

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

[4] BlindReview. Ommited for blind review, 2006.

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

[6] J. Buchli and A. J. Ijspeert. Distributed central pattern generator model for robotics application based on phase sensitivity analysis. In A. J. Ijspeert, M. Murata, and N. Wakamiya, editors, *Biologically Inspired Approaches to Advanced Information Technology: First International Workshop, BioADIT 2004*, volume 3141 of *Lecture Notes in Computer Science*, pages 333–349, Lausanne, Switzerland, Jan. 2004. Swiss Federal Institute of Technology (EPFL), Springer-Verlag Berlin Heidelberg.

[7] J. Busch, J. Ziegler, C. Aue, A. Ross, D. Sawitzki, and

W. Banzhaf. Automatic generation of control programs for walking robots using genetic programming. In E. Lutton, J. A. Foster, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)*, volume 2278 of *Lecture Notes in Computer Science*, pages 258–267, Kinsale, Ireland, Apr. 2002. EvoNet, Springer-Verlag.

[8] S. Chernova and M. Veloso. 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.

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

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

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

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

[13] D. Golubovic and H. Hu. 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] S. Grillner. Control of locomotion in bipeds, tetrapods and fish. In V. B. Brooks, editor, *Handbook of Physiology, The Nervous System, 2, Motor Control*, pages 1179–1236. American Physiology Society, Bethesda, MD, USA, 3 edition, 1981.

[15] S. Grillner. Neurobiological bases of rhythmic motor acts in vertebrates. *Science*, 228(4696):143–149, 1985.

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

[17] S. L. Hooper. Central pattern generators. *Embryonic ELS*, (32):1–12, June 2001.

[18] D. Jacob, D. Polani, and C. L. Nehaniv. 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)*, pages 365–372, Espoo, Finland, June 2005. IEEE Press.

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

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

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

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

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

[24] M. A. Lewis, A. H. Fagg, and A. Solidum. 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)*, pages 2618–2623, Nice, France, 1992. IEEE Press.

[25] E. Marder and R. L. Calabrese. Principles of rhythmic motor pattern production. *Physiological Reviews*, 76:687–717, 1996.

[26] R. B. McGhee. Robot locomotion. *Neural Control of Locomotion*, pages 237–264, 1976.

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

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

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

[30] E. Muybridge. *Animals in Motion*. Dover Publications, Inc., New York, USA, 1957.

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

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

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

[34] K. Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV Proceedings*, pages 28–39, Cambridge, MA, USA, 1994. The MIT Press.

[35] K. Sims. Evolving virtual creatures. *Computer Graphics*, 28:15–24, July 1994.

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

[37] P. S. G. Stein, S. Grillner, A. I. Selverston, and D. G. Stuart. *Neurons, Networks, and Behavior*. The MIT Press, Cambridge, MA, USA, 1997.

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