Uso de Algoritmos Genéticos para a Configuração Automática do Caminhar em Robôs Móveis

Milton Roberto Heinen[1] **e Fernando Santos Osório**[1]

[1]Universidade do Vale do Rio dos Sinos (UNISINOS)
Computação Aplicada - PIPCA
CEP 93022-000 - São Leopoldo - RS - Brasil

mheinen@turing.unisinos.br, fosorio@unisinos.br

Resumo. *Este artigo descreve o sistema LegGen, que realiza a configuração automática do caminhar em robôs simulados dotados de pernas. No sistema LegGen, são utilizados Algoritmos Genéticos para a evolução dos parâmetros do caminhar em robôs móveis simulados através da biblioteca de simulação baseada em física Open Dynamics Engine (ODE). Diversos experimentos foram realizados utilizando robôs de quatro e seis pernas, e os resultados obtidos mostram que o sistema desenvolvido é capaz de configurar o caminhar de forma bastante eficiente.*

1. Introdução

A área de robótica autônoma vem atraindo 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 [Medeiros 1998, Bekey 2005]. Os primeiros robôs móveis desenvolvidos se locomoviam através do uso de rodas [Dudek and Jenkin 2000], mas atualmente os robôs dotados de pernas (*legged robots*) vem ganhando bastante destaque, e várias pesquisas vem sendo realizadas utilizando robôs de 2, 4, 6 ou mais pernas [Bekey 2005].

Um dos maiores problemas no desenvolvimento de robôs dotados de pernas é a geração do caminhar, que é uma tarefa bastante complexa, pois exige a configuração de diversos parâmetros relativos ao caminhar [Kohl and Stone 2004]. A configuração manual destes parâmetros exige muitas horas de um especialista humano, e os resultados obtidos são sub-ótimos [Chernova and Veloso 2004]. Assim, a utilização de técnicas de Aprendizado de Máquina [Mitchell 1997] surge como uma alternativa bastante atrativa para a configuração automática destes parâmetros, facilitando bastante a tarefa em questão [Golubovic and Hu 2003]. Uma das técnicas de Aprendizado de Máquina mais adequadas para a configuração do caminhar são os Algoritmos Genéticos (AG) [Goldberg 1989], pois estes 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 [Mitchell 1996].

O objetivo deste artigo é descrever o sistema LegGen [Heinen and Osório 2006a, Heinen and Osório 2006b], que é um simulador capaz de realizar a configuração automática do caminhar em robôs simulados dotados de pernas. No sistema LegGen, os robôs são simulados dentro de um ambiente virtual bastante realista, no qual as leis da física foram implementadas através do uso da biblioteca *Open Dynamics Engine* (ODE), que uma

biblioteca de software baseada em C++ desenvolvida para a realização de simulações baseadas em física. As principais contribuições deste artigo são: (i) uso de uma biblioteca de simulação baseada em física (ODE) para permitir um maior realismo nas simulações; (ii) utilização de algoritmos genéticos com diferentes funções de *fitness*, e uma comparação dos resultados obtidos com cada uma delas; (iii) teste do sistema de controle em quatro diferentes modelos de robôs simulados, de quatro e seis pernas.

2. Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são métodos de busca estocástica baseados na Teoria da Evolução Natural das Espécies [Darwin 1859], criados por John Holland nos anos 60 [Holland 1975]. 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 se chegue a uma solução que atenda aos objetivos desejados.

Para a implementação dos Algoritmos Genéticos no sistema LegGen, foi selecionada a biblioteca de software GAlib[1], desenvolvida por Matthew Wall do Massachusetts Institute of Technology - 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 Sistema desenvolvido, foi utilizado o tipo de Algoritmo Genético descrito por Goldberg em seu livro [Goldberg 1989], 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, de forma que eles fiquem dentro do intervalo de valores máximos e mínimos possíveis.

3. Simulação de Robôs Móveis

Quando se deseja realizar experimentos em robótica autônoma, duas alternativas são possíveis: (i) realizar os experimentos diretamente em um robô real; ou (ii) realizar os experimentos utilizando um robô simulado através de um ambiente virtual. A utilização de um robô real possui a vantagem de tornar os resultados obtidos mais realísticos [Boeing et al. 2004], pois não são feitas simplificações do modelo em relação à realidade. Já o uso de simulação possui as seguintes vantagens:

- Na simulação não existe o risco de se danificar o robô;
- Tarefas como a 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 o tempo de aprendizado;
- Pode-se testar várias arquiteturas e modelos diferentes de robôs antes da construção física, e assim descobrir com qual modelo de robô é mais adequado.

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

Por estes motivos, optou-se por realizar os experimentos iniciais 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.

3.1. 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 do ambiente 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 várias bibliotecas de software disponíveis para a implementação de simulações baseadas em física. Após o estudo de diversas possibilidades, optou-se pela utilização de uma biblioteca de código aberto e gratuita chamada *Open Dynamics Engine* (ODE)[2], que permite a realização de simulações da dinâmica de corpos rígidos articulados com bastante realismo. Utilizando a ODE, é possível criar diversos corpos rígidos, e estes podem ser conectados através de juntas de vários tipos. Para a movimentação das juntas, é possível que sejam aplicadas forças diretamente aos corpos, ou podem ser utilizados os motores angulares que estão disponíveis no ambiente ODE. Um motor angular é um elemento que pode ser conectado a dois corpos articulados, e que possui uma velocidade e uma força máxima. Com o uso de juntas e motores angulares, é possível que sejam reproduzidas as diversas articulações presentes em um robô real, em seres humanos e nos animais com um alto nível de precisão.

4. Robôs Dotados de Pernas

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

4.1. Estabilidade

Um conceito muito importante relativo ao deslocamento de robôs dotados de pernas é o conceito de 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 [Dudek and Jenkin 2000]. 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 principal vantagem da estabilidade estática é que o risco do robô sofrer quedas é bem pequeno, pois as patas que estão em contato com o solo conseguem garantir a estabilidade mesmo no caso de uma falha ou se as baterias se descarregarem.

Se durante o caminhar o centro de gravidade do robô se deslocar periodicamente para fora do polígono de suporte, mas mesmo assim o robô conseguir se movimentar de forma controlada, é dito que este robô apresenta estabilidade dinâmica. A estabilidade dinâmica é bem mais difícil de ser atingida, pois exige um sofisticado modelo da dinâmica do robô e do uso da inércia [Dudek and Jenkin 2000].

[2]ODE – http://www.ode.org

4.2. Geração do Caminhar

Em um robô dotado de pernas, o caminhar pode ser gerado de diversas formas. Uma das alternativas possíveis consiste na utilização uma máquina de estados para o controle das juntas do robô, na qual é determinada para cada estado a duração deste e o valor de ativação (força/velocidade) para cada um dos motores das juntas. Pelo fato da maioria dos robôs existentes serem simétricos (e grande parte dos seres vivos também), apenas a metade dos estados do autômato precisa ser aprendida pelo Algoritmo Genético - a outra metade é obtida facilmente trocando-se os valores das juntas das pernas da direita com os valores das juntas das pernas da esquerda. A Tabela 1 ilustra um exemplo de tabela de estados para o controle de um robô.

Tabela 1. Exemplo de uma tabela de estados para o controle de um robô

	Estado 1	Estado 2	Estado 3	Estado 4	Estado 5	Estado 6
Duração	0,05	0,10	0,55	0,80	0,55	0,80
Junta A	1,25	2,45	3,00	0,95	3,00	0,55
Junta B	2,70	0,15	0,14	0,25	0,70	1,95
Junta C	1,15	1,65	2,20	0,15	0,30	2,70

A principal desvantagem de se realizar o controle das juntas do robô da forma descrita acima é que nenhum *feedback* do mundo externo é levado em conta no planejamento das ações. Mas em um robô real, pequenas diferenças no comportamento dos atuadores, no nível de carga da bateria, a fricção ou obstáculos externos podem alterar a velocidade efetiva e a resposta das juntas, fazendo com que o mesmo não se comporte da forma esperada[Boeing et al. 2004].

Uma abordagem alternativa consiste na utilização um autômato similar ao anterior, mas ao invés deste autômato determinar a duração e as velocidades das juntas para cada estado, ele determina o ângulo desejado ao final de cada um dos estados para cada uma das juntas do robô[Bongard and Pfeifer 2002]. 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 [Bekey 2005]. 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 próximo estado. Se alguma das juntas não tiver atingido o ângulo desejado após um limite de tempo, o estado é avançado independente desta. Em futuras versões do sistema, esta situação será tratada com mais cuidado, pois uma das pernas pode ter sido bloqueada, o que poderia danificar o robô.

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:

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

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.

5. Sistema LegGen

O Sistema LegGen[3] [Heinen and Osório 2006a, Heinen and Osório 2006b] é um sistema 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 sistema 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 2 mostra os parâmetros utilizados pelo sistema LegGen, com os valores utilizados nas simulações.

Tabela 2. Parâmetros do Sistema LegGen

Parâmetro	**Descrição**	**Valor**
Crossover	Taxa de cruzamentos	0,60
Mutation	Taxa de mutação	0,05
Population size	Tamanho da população	50
Number of generations	Número de gerações	100
Number of states	Número de estados do autômato	4
Time of walk	Tempo de caminhada	60
Velocity min	Velocidade relativa máxima	0,0
Velocity max	Velocidade relativa mínima	1,0

Os parâmetros *crossover*, *mutation*, *population size* e *number of generations* são usados diretamente pela biblioteca GAlib para definir o funcionamento do Algoritmo Genético. O parâmetro *number of states* define o número de estados do autômato finito, o parâmetro *time of walk* 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 *velocity min* e *velocity max* definem a faixa na qual as velocidades relativas (Vr) serão geradas pelo Algoritmo Genético.

O funcionamento do sistema 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 sistema são lidos, e o Algoritmo Genético é inicializado e executado até que seja atingido o número de gerações desejado. A avaliação de cada cromossomo é realizada 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 capturas e armazenadas informações sensoriais relativas ao caminhar;

[3]LegGen – http://www.inf.unisinos.br/~osorio/leggen

- O *fitness* é calculado e retornado para a GAlib.

Para o cálculo da função de *fitness*, as seguintes informações sensoriais precisam ser calculadas: (i) distância percorrida pelo robô (D); (ii) instabilidade (G); e (iii) número médio de patas em contato com o solo (B). A distância percorrida 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 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 [Dudek and Jenkin 2000]. A instabilidade G (Giroscópio) é calculada através da equação [Golubovic and Hu 2003]:

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

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:

$$\overline{x}_x = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \overline{x}_y = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \overline{x}_z = \frac{\sum_{i=1}^{n} z_i}{n}. \tag{4}$$

Para obter número médio de patas em contato com o solo, foram simulados sensores de toque que imitam o funcionamento de *bumpers* [Bekey 2005] colocados em baixo de cada uma das patas do robô. Durante a simulação, o número de patas em contado com o solo é coletado, e ao final de cada simulação física é calculado número médio de patas em contato com o solo B (Bumpers) através da fórmula:

$$B = \frac{\sum_{i=1}^{n} b_i}{n}, \tag{5}$$

onde b_i é o número de patas em contato com o solo no instante i. Após o processamento das informações sensoriais, o *fitness* F é calculado através da fórmula:

$$F = \frac{D}{1 + G + (B - L/2)^2}, \tag{6}$$

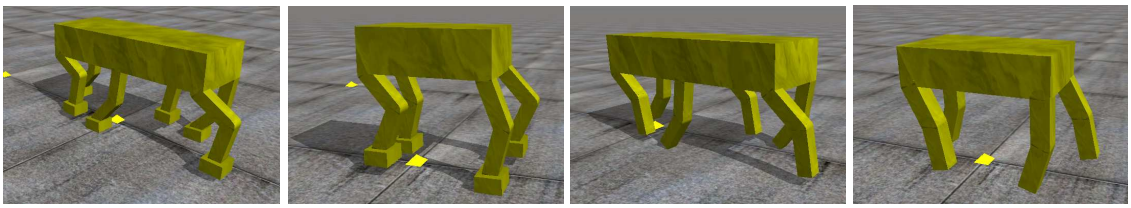
onde L é o número de pernas (Legs) do robô. Pela análise da função de *fitness*, se observa que B atinge o seu valor ideal quando o robô mantém em média metade de suas patas em contato com o solo, o que é desejado, pois se o robô mantiver todas as patas no chão, ele não irá se deslocar, e se manter menos da metade das patas em contato com o solo, ele fatalmente irá cair. Em relação aos outros parâmetros, o indivíduo mais bem qualificado é aquele que possui a melhor relação entre velocidade e estabilidade, de forma que as melhores soluções são aquelas mantém o compromisso entre estes dois critérios de avaliação [Golubovic and Hu 2003].

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.

5.1. 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 sistema será estendido para aceitar modelos de robôs com juntas mais complexas.

Inicialmente foram modelados e testados diversos tipos de robôs, até que se chegou aos quatro modelos principais, mostrados na Figura 1. O modelo da Figura 1(a),



(a) HexaL3J (b) TetraL3J (c) HexaL2J (d) TetraL2J

Figura 1. Modelos de robôs utilizados nas simulações

chamado de HexaL3J, 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 1(b), chamado de TetraL3J, é similar ao da Figura 1(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:

$$\alpha_p = -\sum_{i=1}^{n} \alpha_i, \tag{7}$$

onde α_i é o ângulo da junta i e n é o número de juntas da perna. Utilizando a Fórmula 7, as patas do robô ficam sempre paralelas ao solo. O modelo de robô da Figura 1(c), chamado de HexaL2J, é similar ao da Figura 1(a), mas possui apenas duas articulações por perna, e todos os ângulos das juntas são calculados pelo Algoritmo Genético, sem utilizar a Fórmula 7. O modelo da Figura 1(d), chamado de TetraL2J, possui quatro pernas e duas articulações por perna. A Tabela 3 mostra as dimensões dos robôs em centímetros.

Similar ao que acontece na maioria dos mamíferos de quatro pernas, as patas dos robôs das Figuras 1(a) e 1(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 Tabela 4 mostra os limites máximos e mínimos das juntas dos robôs da Figura 1 Os robôs do tipo hexapod (HexaL3J e HexaL2J) possuem todas as pernas idênticas, de forma que as juntas das pernas centrais são idênticas às demais.

Tabela 3. Dimensões dos robôs simulados

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

Tabela 4. Limites mínimos e máximos das juntas

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

6. Resultados

Para que fosse determinado qual o melhor modelo de robô a ser construído futuramente, diversos experimentos foram realizados. Desta forma, foram realizados experimentos para verificar se era mais vantajoso construir um robô utilizando quatro ou seis pernas, e também para que fosse determinada a melhor quantidade de partes em cada perna a ser utilizada. Devido a natureza estocástica dos Algoritmos Genéticos, cada um dos experimentos realizados foi repetido dez vezes com sementes aleatórias diferentes, e ao final foi calculada a média e o desvio padrão dos resultados obtidos. A Tabela 5 mostra os resultados obtidos nos experimentos realizados.

Tabela 5. Resultados obtidos nas simulações

	F		D		G		B	
Robô	μ	σ	μ	σ	μ	σ	μ	σ
HexaL3J	436,4	85,0	689,4	102,6	0,600	0,182	2,73	0,52
HexaL2J	445,3	55,5	706,7	95,2	0,591	0,144	2,91	0,19
TetraL3J	261,8	80,4	383,1	93,8	0,502	0,222	1,82	0,34
TetraL2J	183,0	72,5	330,1	78,9	0,912	0,372	1,86	0,98

A primeira coluna descreve os robôs utilizados nos experimentos, e as colunas seguintes mostram, respectivamente, a média (μ) e o desvio padrão (σ) do *fitness* (F), da distância percorrida pelo robô em centímetros (D), da instabilidade (G) e do número médio de patas em contato com o solo (B). Observando os resultados da Tabela 5 pode-se tirar as seguintes conclusões:

- Os robôs de seis pernas são mais velozes que os de quatro pernas;
- O HexaL2J é um pouco mais veloz que o HexaL3J;
- Os TetraL3J se deslocou mais que o TetraL2J, e de forma mais estável;

A primeira conclusão é justificada pelo fato de os robôs de seis pernas possuírem estabilidade estática com apenas a metade de suas patas em contato com o chão, o que faz

com que eles possam empregar maiores velocidades sem o risco de tombar. A segunda e a terceira conclusões mostram que um robô de seis pernas necessita de apenas duas partes móveis em cada perna, mas um robô de quatro pernas necessita de no mínimo três. Pelos resultados da Tabela 5, conclui-se que o robô que apresenta a melhor relação entre

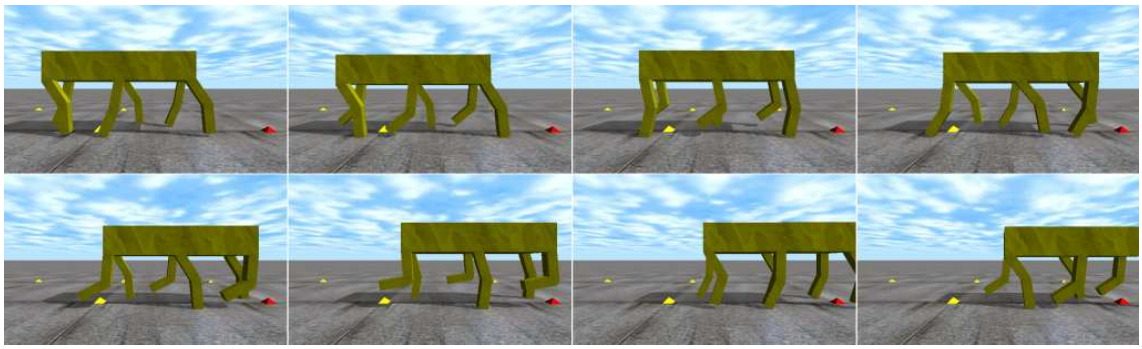


Figura 2. Exemplo de caminhada do robô HexaL2J

velocidade, estabilidade e custo é o HexaL2J, pois consegue se locomover da mesma forma que HexaL3J, porém necessitando menos componentes mecânicos e eletrônicos (juntas e motores). A Figura 2 mostra um exemplo de caminhada realizada pelo robô HexaL2J, e a Figura 3 mostra um exemplo de caminhada realizada pelo robô TetraL3J[4].

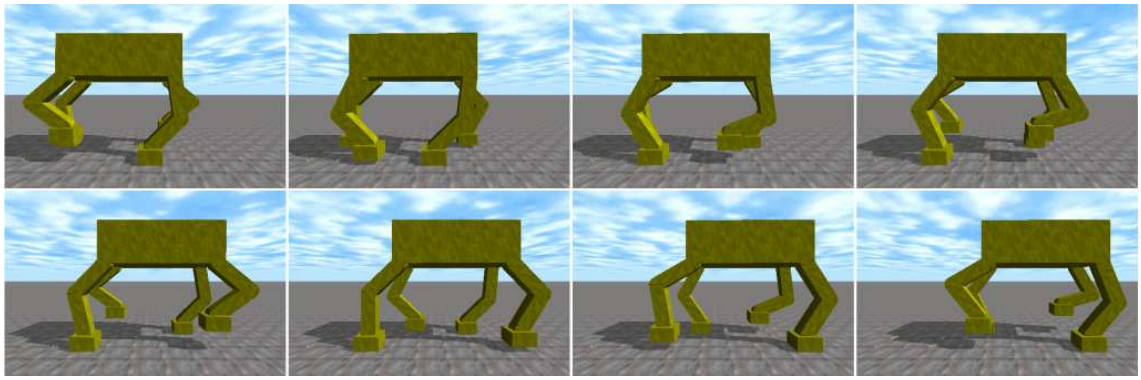


Figura 3. Exemplo de caminhada do robô TetraL3J

A execução das 40 simulações (10 simulações para cada robô) levou aproximadamente 16 horas em um computador típico (*clock* de 1,54GHz, 512MB de memória RAM e placa aceleradora gráfica de 128MB).

7. Conclusões e perspectivas

O objetivo deste artigo foi descrever o sistema LegGen, que é um sistema desenvolvido para realizar a configuração automática do caminhar de robôs móveis dotados de pernas. Neste sistema, 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. As perspectivas

[4]Vários vídeos estão disponíveis em http://www.inf.unisinos.br/~osorio/leggen

futuras incluem a construção de um robô físico utilizando as especificações dos robôs virtuais, bem como a extensão do sistema LegGen para ser utilizado em robôs bípedes e com juntas mais sofisticadas, para assim permitir o desenvolvimento de robôs humanóides.

Referências

Bekey, G. A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, Cambridge, MA.

Boeing, A., Hanham, S., and Bräunl, T. (2004). Evolving autonomous biped control from simulation to reality. In *Proc. 2nd Int. Conf. Autonomous Robots and Agents (ICARA)*, pages 440–445, Palmerston North, New Zealand.

Bongard, J. C. and Pfeifer, R. (2002). A method for isolating morphological effects on evolved behaviour. In *Proc. 7th Int. Conf. Simulation of Adaptive Behaviour (SAB)*, pages 305–311, Edinburgh, UK. MIT Press.

Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sendai, Japan.

Darwin, C. (1859). *Origin of Species*. John Murray, London, UK.

Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge Univ. Press, Cambridge, UK.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Golubovic, D. and Hu, H. (2003). Ga-based gait generation of sony quadruped robots. In *Proc. 3th IASTED Int. Conf. Artificial Intelligence and Applications (AIA)*, Benalmadena, Spain.

Heinen, M. R. and Osório, F. S. (2006a). Applying genetic algorithms to control gait of physically based simulated robots. In *Proc. IEEE Congr. Evolutionary Computation (CEC), to appear*, Vancouver, Canada.

Heinen, M. R. and Osório, F. S. (2006b). Uso de realidade virtual para a simulação do caminhar em robôs móveis. In *Proc. VIII Symposium on Virtual Reality*, Belém, PA, Brazil. Editora CESUPA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Univ. Michigan Press, Ann Arbor, MI.

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2619–2624, New Orleans, LA.

Medeiros, A. (1998). Introdução à robótica. In *Anais do XVII Encontro Nacional de Automática*, volume 1, pages 56–65, Natal, RN, Brazil.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Mitchell, T. (1997). *Machine Learning*. McGrall-Hill, New York.