In: 12º Congresso Regional de Informática e Telecomunicações (MT Digital), Cuiabá, MT

Utilizando Redes Neurais Artificiais no Controle Robusto de Navegação de Robôs Móveis

Gustavo Pessin[1], Fernando Osório[2], Soraia Musse[3]

[1]Faculdades UNICEN
Av. Guterres 241 – Primavera do Leste – MT – Brasil

[2]PIPCA – Universidade do Vale do Rio dos Sinos (UNISINOS)
Av. Unisinos 950 – São Leopoldo – RS – Brasil

[3]Faculdade de Informática – PUC-RS
Av. Ipiranga, 6681 – Porto Alegre – RS – Brasil

pessin@gmail.com, fosorio@gmail.com, soraia.musse@pucrs.br

Resumo. *O objetivo deste artigo é detalhar o modelo, a implementação e a avaliação da eficiência de uma Rede Neural Artificial (RNA) aplicada ao controle de navegação de robôs móveis fisicamente simulados[1]. A atuação dos robôs se dá em um ambiente virtual de simulação realística, que apresenta obstáculos estáticos e dinâmicos; desenvolvido em C++ com OSG, ODE e Demeter. As entradas da RNA são obtidas através de sensores presentes em cada um dos robôs (e.g. GPS, bússola, sonar). Este trabalho é parte de um projeto de desenvolvimento de um Sistema Multi-Agente (SMA) para monitoração e combate de incêndios em ambiente florestais [Pessin et. al. 2007]. Propomos neste artigo que a navegação dos robôs passe a ser completamente realizada pela RNA. Os resultados das simulações demonstram que a RNA é capaz de controlar de modo satisfatório os robôs móveis, e que o modelo de RNA proposto pode tanto ser usando no SMA de monitoração e combate de incêndios como em qualquer outro sistema que necessite de controle reativo de navegação.*

1. Introdução

Com a evolução das pesquisas em robótica, cada vez mais os robôs estão se tornando complexos em termos físicos. A grande variedade de estudos em morfologia robótica tem desenvolvido variações de robôs dotados de diversos meios de locomoção (*e.g.* pernas, rodas, esteiras). Em paralelo a este desenvolvimento temos a evolução constante de uma gama extremamente grande de sensores (*e.g.* sistemas de visão, posicionamento, detecção de obstáculos). O desenvolvimento de algoritmos e técnicas para coordenar estes conjuntos físicos em um ambiente dinâmico é um desafio extremamente complexo [Go, Browning e Veloso 2004]. Dotar robôs autônomos de capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos é uma área de pesquisa que tem atraído a atenção de um grande número de pesquisadores [Dudek e Jenkin 2000]. O uso de ambientes de realidade virtual torna a pesquisa em robótica muito mais ágil do que a pesquisa usando robôs reais, e, o uso de técnicas de RNA torna o desenvolvimento robótico de acordo com os Novos Princípios de Desenvolvimento

[1] Códigos-fonte e vídeos das simulações disponíveis em http://pessin.googlepages.com

Robótico (baseados em inspiração biológica) propostos em [Pfeifer e Scheier 1994, 1997; Pfeifer, Iida e Bongard 2005].

No trabalho [Pessin et. al. 2007] detalhamos o projeto e o desenvolvimento de um Sistema Multi-Agente (SMA) que opera em um ambiente virtual de simulação realística - completamente baseado em regras - onde uma equipe de agentes autônomos trabalha cooperativamente a fim de realizar com sucesso a identificação e o combate de incêndios florestais, sem intervenção humana. Experiências iniciais com RNAs aplicadas no controle foram apresentadas em um modelo bidimensional (2D) em [Pessin et. al. 2007a] e após em um modelo tridimensional com chão plano em [Pessin et. al. 2007b]. No presente artigo descrevemos a evolução da forma de controle de navegação, apresentando o modelo, a implementação e a avaliação da eficiência de uma Rede Neural Artificial (RNA) aplicada no controle da navegação dos robôs móveis com atuação em terreno irregular e com simulação de ruído nos sensores e atuadores.

Um problema fundamental da robótica é a navegação. O deslocamento de um lugar para outro depende de três aspectos fundamentais: localização, orientação e controle motor. Para conhecer tanto sua localização como sua orientação, um robô móvel deve possuir sensores próprios (*e.g.* GPS, bússola). Para o controle motor, deve possuir um número adequado de motores (um veículo autônomo, por exemplo, usualmente possui um motor angular, para giro das rodas e um motor linear, para tração). Normalmente sensores e atuadores são sujeitos a erros e interferências, assim o controle das ações de um robô deve sempre levar em conta a imprecisão dos sensores e motores envolvidos. Um sistema robusto deve permitir que, mesmo com sensores e atuadores imprecisos, o agente cumpra o seu objetivo. Uma técnica de Aprendizado de Máquina indicada para este controle é a de Redes Neurais Artificiais, dada sua capacidade de aprendizado a partir de exemplos e a respectiva generalização e adaptação das saídas. É uma técnica muito utilizada no controle de navegação de sistemas reativos [Marchi 2001; Osório et. al. 2006].

Um grande sonho de nossa sociedade é a aplicação de sistemas robóticos substituindo atividades que coloquem em risco a vida humana, em atividades onde a atuação de humanos é deficitária ou onde a atuação humana de certa forma é ineficiente. A tarefa de monitoração e combate de incêndios em áreas florestais é um caso onde a aplicação de um sistema multi-robótico poderia diminuir consideravelmente os prejuízos humanos, materiais e ambientais. Os incêndios florestais causam diversos tipos de danos humanos, materiais e ambientais. A extensão territorial do Brasil e a diversidade de sua cobertura vegetal, bem como o número expressivo de ocorrências de incêndios florestais verificados no país, são fatores que enfatizam a necessidade de um sistema cada vez mais aprimorado e consistente de detecção e combate de incêndios florestais [Batista 2004]. Diversas iniciativas a fim de incrementar a capacidade de reação de órgãos públicos e civis no sentido de evitar desastres tem sido uma das preocupações junto a órgãos como a Secretaria Nacional de Defesa Civil, levando a criação de novos CEPEDs (Centro de Estudos para a Prevenção de Emergências e Desastres). Importantes iniciativas, como a RBV – Rede Brasileira de Visualização, financiada pela FINEP, também tem sido incentivadas, onde a competência de Segurança e Defesa (Civil e Militar) da Rede vem sendo foco de desenvolvimentos junto a nossa Universidade e na qual este projeto se integra. O controle automático de navegação dos veículos é parte fundamental deste projeto.

Neste artigo apresentamos na Seção 2 características importantes de robótica móvel que tenham relação com as necessidades deste trabalho. Na Seção 3 descrevemos brevemente conceitos de simulação e modelagem, as ferramentas pesquisadas e as bibliotecas de simulação selecionadas para o desenvolvimento do trabalho. Na Seção 4 apresentamos conceitos de Aprendizado de Máquina e de Redes Neurais Artificiais. Na Seção 5 descrevemos técnicas e operações reais de identificação e combate a incêndios florestais. Na Seção 6 descrevemos o ambiente desenvolvido, a operação multi-agente de identificação e combate, o desenvolvimento, o treinamento e a avaliação da Rede Neural no controle dos robôs móveis. Finalizamos apresentando a conclusão do trabalho realizado.

2. Robótica Móvel

Um robô móvel é um dispositivo mecânico montado sobre uma base não fixa que age sob o controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir com o ambiente [Marchi 2001, Bekey 2005]. A interação com o ambiente se dá através de ciclos de percepção-ação que consistem em três passos fundamentais: (*i*) Obtenção de informação através de sensores; (*ii*) Processamento das informações para seleção de ação; e, (*iii*) Execução da ação através do acionamento dos atuadores.

Esse conjunto de operações, em uma análise superficial, pode parecer simples, porém o controle robusto de sistemas robóticos tem complicações físicas e mecânicas (como cinemática e dinâmica), eletrônicas (como falta de precisão de sensores) e computacionais que tornam a criação de um conjunto de regras uma tarefa árdua e sujeita a erros. Sensores são os mecanismos de percepção de um robô e realizam medições físicas (*e.g.* contato, distância, orientação, temperatura), provêm sinais ou dados crus que precisam ser interpretados pelo "cérebro" do robô. A interpretação destes sinais deve ser a única maneira de um robô autônomo entender o ambiente que o cerca para poder realizar as mudanças de ação necessárias [JPL/NASA 2007]. Atuadores são os mecanismos de ação de um robô (*e.g.* motores, pistões, braços manipuladores), controlados por circuitos eletrônicos que recebem valores de ação, valores estes que devem ser calculados pelo "cérebro" do robô e devem estar de acordo com especificações de fabricantes [JPL/NASA 2007].

3. Simulação e Modelagem

Experimentos em robótica móvel podem ser realizados de duas formas: diretos em um robô real ou em um robô simulado em um ambiente virtual realista [Pfeifer e Scheier 1999]. Usualmente, experimentos em robótica móvel utilizando um robô real exigem enorme despendimento de tempo e de recursos financeiros. Para que seja possível a implementação física real, o sistema que propomos deve ser projetado, desenvolvido e testado anteriormente em ambientes de simulação realísticos. A simulação de sistemas robóticos é especialmente necessária para robôs caros, grandes, ou frágeis [Go, Browning e Veloso 2004]. É uma ferramenta extremamente poderosa para agilizar o ciclo de desenvolvimento de sistemas de controle robóticos eliminando desperdício de recursos, tanto financeiros como computacionais. Para que uma simulação seja útil, entretanto, ele deve capturar características importantes do mundo físico, onde o termo *importantes* tem relação ao problema em questão [Go, Browning e Veloso 2004]. No caso deste trabalho, é fundamental que existam restrições físicas no modelo e que exista

a possibilidade de trabalho em um terreno irregular, provido de obstáculos. Para o desenvolvimento deste trabalho foram pesquisadas algumas ferramentas de simulação, porém, nenhuma mostrou possuir o conjunto completo das características requisitadas. Uma pequena descrição de cada uma das ferramentas pesquisadas é fornecida a seguir.

O *Microsoft Robotics Studio* (msdn.microsoft.com/robotics) tem como objetivo prover um ambiente para desenvolvimento de simulação robótica. Permite a simulação em terrenos irregulares e é livre apenas para uso não comercial. O desenvolvimento dos sensores, dos controles inteligentes e da comunicação entre os robôs depende de programação (*e.g. C#, VB.NET*). O *Webots* (www.cyberbotics.com) é um simulador de robôs móveis que tem como base a biblioteca de simulação física ODE, inclui modelos prontos de alguns robôs comerciais e modela sensores com a capacidade de detecção de obstáculos, visão, e manipuladores simples. O usuário pode programar cada robô utilizando *C/C++*. É um produto comercial. Outros ambientes de simulação estudados foram o *Khepera Simulator* (diwww.epfl.ch/lami/team/michel/khep-sim), o *Mission Simulation Facility* (ase.arc.nasa.gov/msf), o *JUICE* (www.natew.com/juice) e o *Simulator BOB* (www.tu-harburg.de/ti6/mitarbeiter/pst/Sim/Simulator.html).

Um dos requisitos básicos para nosso ambiente de simulação é que todas as bibliotecas de programação sejam software livre e em linguagem C/C++. As bibliotecas utilizadas no desenvolvimento do sistema são: (i) *Open Dynamics Engine* (ODE): (www.ode.org) é uma biblioteca desenvolvida para a simulação física de corpos rígidos articulados [Smith 2006]. É utilizada para o desenvolvimento dos robôs e do mundo colisivo. Uma estrutura articulada é criada quando corpos rígidos são conectados por algum tipo de articulação, como, por exemplo, um veículo terrestre que tem a conexão de rodas em um chassi. A ODE não tem como objetivos realizar simulação de outras dinâmicas além da dinâmica de corpos rígidos (*e.g.* ondas, fluídos, corpos flexíveis, fraturas). A utilização da ODE no nosso ambiente é fundamental por fornecer restrições físicas, principalmente na definição da morfologia dos robôs; (*ii*) *Open Scene Graph (OSG)*: (www.openscenegraph.org) é uma biblioteca para desenvolvimento de aplicações gráficas 3D de alta performance. Baseada no conceito de grafos de cena, provê ao desenvolvedor um ambiente orientado a objeto sobre a *OpenGL* (www.opengl.org), liberando este da necessidade de implementação e otimização de chamadas gráficas de baixo nível. Um grafo de cena permite a representação de objetos em uma cena com uma estrutura que permite a criação de grupos que podem compartilhar propriedades, assim, podemos definir uma propriedade comum em um nível hierárquico mais alto e todos os objetos inferiores receberão esta propriedade [Barros, Gonzaga e Raposo 2007]; (*iii*) *DrawStuff*: como a ODE é completamente independente de visualizador, iniciar a criação dos corpos e das simulações pode ser uma tarefa bastante árdua caso não tenhamos uma forma simples e fácil de visualizar os objetos. Por este motivo, a biblioteca *DrawStuff* é disponibilizada em conjunto com a ODE. Basicamente, o *DrawStuff* é um ambiente de visualização de objetos 3D que tem o propósito de permitir a demonstração visual da ODE sendo uma biblioteca bastante simples e rápida para utilização; (*iv*) *Demeter*: (www.tbgsoftware.com) é uma biblioteca desenvolvida para renderizar terrenos 3D, desenvolvida para ter rápida performance e boa qualidade visual, pode renderizar grandes terrenos em tempo-real sem necessidade de hardware especial, depende da *Simple DirectMedia Layer* (www.libsdl.org) para realizar o tratamento das texturas do terreno e da *Geospatial Data Abstraction Library* (www.gdal.org) para carregar arquivos de elevação.

4. Aprendizado de Máquina

Aprendizado de Máquina (*Machine Learning*) é uma área da Inteligência Artificial que tem como objetivo desenvolver técnicas computacionais de aprendizado e de aquisição de conhecimentos [Rezende 2003]. Essas técnicas devem exibir um comportamento inteligente e realizar tarefas complexas com um nível de competência equivalente ou superior ao de um especialista humano [Nikolopoulos 1997]. Para a construção do sistema proposto neste artigo utilizamos Redes Neurais Artificiais, assim, descrevemos sucintamente suas características a seguir.

4.1. Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são sistemas paralelos distribuídos, compostos por unidades de processamento simples que calculam determinadas funções matemáticas, normalmente não-lineares [Braga, Carvalho e Ludermir 2000]. Seus atributos básicos podem ser divididos em arquitetura e neurodinâmica. A arquitetura determina a estrutura da rede, ou seja, o número de neurônios e sua interconectividade; a neurodinâmica, por sua vez, define as propriedades funcionais da rede, ou seja, como ela aprende, recupera, associa e compara novas informações com o conhecimento já armazenado [Kartalopoulos 1996]. Matematicamente, RNAs são aproximadores universais, que realizam mapeamentos em espaços de funções multi-variáveis [Hornik, Stinchcombe e White 1989].

O processamento da informação em uma RNA é feito por meio de estruturas neurais artificiais [Rezende 2003], sendo que esta estrutura, bem como o próprio neurônio artificial são uma analogia biológica ao funcionamento do cérebro. O neurônio artificial proposto por [McCulloch e Pitts 1943] pode ser descrito como um modelo com n terminais de entrada $x_1,x_2,x...,x_n$ que, para cada entrada, possui um peso w_i correspondente. A soma das entradas x_i ponderadas pelos pesos correspondentes w_i produzem a chamada *saída linear u*. A saída *y* do neurônio é obtida pela aplicação de uma função *f(.)* à saída linear *u*. A função *f(.)* é chamada *função de ativação* e pode assumir diferentes formas (*e.g.* linear, semi-linear ou *sigmoid*). A função de ativação *sigmoid* faz com que a saída de um neurônio seja normalizada entre 0 e 1. Devido a sua curvatura ela é usada no processo de aprendizado para gerar certa estabilidade no sistema, evitando que os pesos variem muito depois que o valor da saída já se encontre próximo a um dos extremos [Rumelhart, Hinton e Williams 1986]. O algoritmo *Backpropagation* [Rumelhart e Mcclelland 1986] é um algoritmo supervisionado de aprendizagem de RNAs. A base de treinamento é um conjunto de dados que deve apresentar, para cada entrada, a saída prevista do sistema. Este tipo de aprendizado ocorre em várias épocas; cada época representa a apresentação do conjunto inteiro de dados a rede neural para o ajuste dos pesos. O treinamento no *Backpropagation* ocorre em duas fases [Braga, Carvalho e Ludermir 2000]: a fase *forward* é utilizada para definir a saída da rede para um dado padrão de entrada; a fase *backward* utiliza a saída desejada e a saída fornecida pela rede para atualizar o peso das conexões. O *Backpropagation* é baseado na Regra Delta Generalizada [Widrow e Hoff 1960] e os ajustes dos pesos são realizados pelo método da Descida do Gradiente [Braga, Carvalho e Ludermir 2000]. Por utilizar o método da Descida do Gradiente, não há garantia de que o aprendizado não tenha ficado preso em mínimos locais (Figura 1(a)), assim, o treinamento de uma rede neural deve envolver várias rodadas de simulação, inicializando os pesos de forma aleatória. Outra questão importante é o grau de

generalização. Quando uma rede neural é treinada, os pesos sinápticos vão se ajustando para que sejam dadas respostas satisfatórias à base de treinamento apresentada. Se o aprendizado for feito por um número muito grande de épocas pode ocorrer o chamado *overfitting*, onde a RNA começa a decorar a saída, perdendo sua capacidade de generalização. A solução deste problema usualmente envolve o uso de uma base de validação usada em paralelo a base de treinamento (busca do Ponto Ótimo de Generalização) como mostra a Figura 1(b).

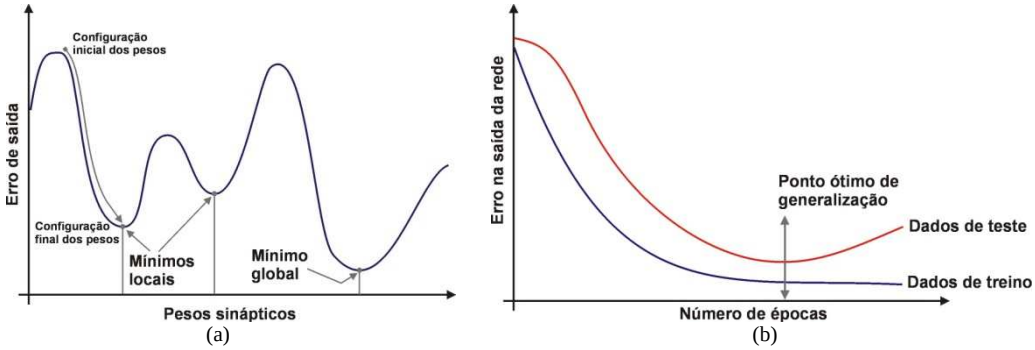


Figura 1. (a) Descida do gradiente de uma superfície de erro; (b) Curvas de erro em aprendizado e validação.

O algoritmo *Backpropagation* padrão é muito lento e seu desempenho piora sensivelmente para problemas maiores e mais complexos; mesmo para problemas relativamente simples, geralmente requer que todos os padrões de treinamento sejam apresentados centenas ou milhares de vezes, o que limita sua utilização prática [Braga, Carvalho e Ludermir 2000]. Os algoritmos atualmente empregados no treinamento são variações do *Backpropagation*, que tem em geral o foco de aumentar a velocidade do treinamento ou melhorar a classificação de padrões. As variações mais conhecidas são: *Backpropagation com Momentum* [Rumelhart e McClelland 1986]; *Quickprop* [Fahlman 1988]; e *Resilent Backpropagation* (RProp) [Riedmiller e Braun 1994].

O *Stuttgart Neural Network Simulator (SNNS* - http://www-ra.informatik.uni-tuebingen.de/SNNS/) é um simulador de Redes Neurais Artificiais criado no *Institute for Parallel and Distributed High Performance Systems* (IPVR) da Universidade de Stuttgart. Proporciona um ambiente eficiente e flexível para auxiliar a criação, o treinamento e a manutenção de Redes Neurais Artificiais. O SNNS possui um grande número de algoritmos de aprendizado, como *Backpropagation*, *Quickprop*, *RProp*, *Backpercolation*, *Counterpropagation*, *Generalized radial basis functions*, entre outros. O *kernel* do sistema é desenvolvido em C e sua utilização pode ser completamente feita através de linha de comando, porém também possui uma interface desenvolvida em JAVA (JavaNNS). Esta interface gráfica possibilita fácil criação de diversas topologias de Redes Neurais Artificiais, além de permitir acompanhar a evolução de taxas de erro e aprendizado da RNA através de gráficos. Um aplicativo do pacote SNNS, o SNNS2C, permite a conversão de uma RNA em código C, que pode então ser facilmente inserido em outra aplicação.

5. Técnicas Reais de Identificação e Combate de Incêndios Florestais

Para os SMA proposto em [Pessin et. al. 2007] foi realizada uma extensa pesquisa a fim de melhor entender como proceder no combate a incêndios florestais, e assim planejar as estratégias a serem implementadas nos agentes autônomos, foi realizado um estudo

sobre as técnicas reais de operação. Este estudo teve como base os trabalhos de [Antunes 2000, Batista 2004, CPTEC/INPE 2006, LIF 2006, Remel and Pereira 2001]. A operação de combate ou supressão de um incêndio envolve seis etapas distintas [LIF 2006]: (i) Detecção: tempo decorrido entre o início do fogo e o momento que ele é visto por alguém, alguns métodos são: torres de vigilância, patrulhamento (e.g. terrestre, avião) ou imagens de satélites; (ii) Comunicação: tempo compreendido entre a detecção do fogo e o recebimento da informação por um responsável; (iii) Mobilização: tempo gasto entre o recebimento da informação da existência do fogo e a saída do pessoal para combate; (iv) Deslocamento: tempo compreendido entre a saída do pessoal de combate e a chegada da primeira turma ao local do incêndio; (v) Planejamento: tempo gasto pelo responsável pelo combate para avaliar o comportamento do fogo e planejar a estratégia de combate; e (vi) Combate: tempo consumido na operação de combate do incêndio. Existem quatro métodos de combate ao fogo nos incêndios florestais [LIF 2006]: (i) Método direto: usado quando a intensidade do fogo permite uma aproximação suficiente da brigada à linha de fogo; utilizadas as seguintes técnicas e materiais: água (e.g. bombas costais, baldes, moto-bombas); terra (pás); ou batidas (abafadores); (ii) Método paralelo ou intermediário: usado quando não é possível o método direto e a intensidade do fogo não é muito grande, consiste em limpar, com ferramentas manuais, uma estreita faixa, próxima ao fogo, para deter o seu avanço e possibilitar o ataque direto; (iii) Método indireto: usado em incêndios de intensidade muito grande, consiste em abrir aceiros com equipamento pesado (e.g. trator, motoniveladeira), utilizando ainda um contra-fogo, para ampliar a faixa limpa e deter o fogo, antes que chegue ao aceiro; e (iv) Método aéreo: usado nos incêndios de copa, de grande intensidade e área e em locais de difícil acesso às brigadas de incêndio, são usados aviões e helicópteros, especialmente construídos ou adaptados para o combate ao incêndio.

A rapidez e a eficiência na detecção e monitoramento dos incêndios florestais são fundamentais para a viabilização do controle do fogo, redução dos custos nas operações de combate e atenuação dos danos. Para países de grande extensão territorial, como o Brasil, o monitoramento dos incêndios florestais através de imagens de satélites é o meio mais eficiente e de baixo custo quando comparado com os demais meios de detecção [Batista 2004]. O lançamento em 1972 do primeiro satélite *Landsat* possibilitou detectar alterações nas áreas florestais do espaço. Desde então, as imagens termais e de infravermelho têm sido usadas na detecção de incêndios e estudos de mapeamento, permitindo que áreas queimadas e não queimadas sejam detectadas através do contraste entre os gradientes térmicos [Remmel e Perera 2001].

6. Implementação do Protótipo

A operação multi-agente depende essencialmente de duas etapas: **planejamento** e **ação**, assim, foram implementados protótipos específicos para cada etapa. Ambos foram desenvolvidos em C++. O protótipo onde é realizada a ação dos robôs bombeiros, controlados por uma Rede Neural Artificial pode ser visto na Figura 2. Este protótipo tridimensional usa a biblioteca OSG, responsável pela saída gráfica do protótipo, a biblioteca *Demeter*, responsável pelo terreno irregular e a biblioteca ODE, responsável pelo realismo físico, tanto da morfologia robótica como da colisão entre os objetos presentes no ambiente (*e.g.* robôs, árvores, inclinação de terreno). O uso da biblioteca ODE permite que os robôs simulados fisicamente respeitem questões como gravidade, inércia e atrito. Por exemplo, mantendo uma força f constante nos motores lineares

(torque) um veículo terá velocidade *v* em regiões planas, em regiões de declive terá *v* maior e em aclives terá *v* menor. O protótipo onde é realizado o planejamento usa Algoritmos Genéticos e foi descrito em [Pessin et. al. 2007c]. A integração entre os protótipos se dá através de um arquivo texto, após realizar a evolução para obtenção de posições, estas são passadas para o protótipo de ação no combate.

Figura 2. Ambiente de simulação desenvolvido com OSG, ODE e Demeter apresentando veículos em deslocamento.

No ambiente desenvolvido (Figura 2), cada árvore existe como um modelo OSG e como um cilindro ODE. Os sensores identificam os cilindros que são posicionados junto às árvores. O ambiente permite que, através de parâmetros, possamos habilitar ou não a exibição dos objetos físicos ao invés de apenas a sua representação gráfica.

6.1. Operação de Identificação e Combate

Usamos o ambiente para simular a seguinte operação: um agente monitor (satélite) monitora o terreno da área florestal, ao identificar uma área com foco de incêndio, ativa o módulo de evolução de estratégias [Pessin et. al. 2007c]. Após obter as coordenadas de atuação através do GA, o agente monitor envia mensagens aos agentes de combate informando suas posições de atuação (nesta implementação simulamos esta operação através de um arquivo texto). O comportamento dos agentes de combate é reativo, deslocando-se em direção a posição de seu objetivo específico desviando de obstáculos. O método de combate de incêndio simulado é o método indireto. Os agentes de combate simulados são motoniveladoras que tem como finalidade cercar o foco de incêndio e criar um aceiro (área livre de vegetação onde o fogo se extingue pela falta de combustível). Quanto ao controle de posicionamento, um sensor do tipo GPS é simulado em cada robô. Em experimentos de campo realizados com um GPS *Garmin Etrex* (www.garmin.com) obtivemos um erro médio de 18,6 metros. Considerando que cada agente possui seu próprio GPS, o tratamento deste erro é crucial na criação dos aceiros. Tratamos esta informação da seguinte maneira: o erro médio deste sensor durante o deslocamento é usado somado a distância de criação do aceiro e também somado ao final da área de criação.

6.2. Morfologia dos Robôs Móveis

Os robôs móveis foram desenvolvidos com a biblioteca de simulação de corpos rígidos articulados ODE. A Figura 3 apresenta imagens do veículo desenvolvido. Dada a existência de restrições físicas, a única maneira de controlar este veículo é com a aplicação de forças em seus dois motores simulados, que são: um motor angular (para o giro do volante) e um motor linear (para o torque). Além do GPS, responsável pela

obtenção da localização, cada robô possui também uma bússola, necessária para a obtenção da orientação do veículo.

Figura 3. Imagens de simulações utilizando os veículos desenvolvidos.

O azimute (ângulo para o alvo) é obtido a partir da posição atual (GPS) e da posição do alvo (recebida por mensagem). Os sensores de distância são sonares simulados, apresentando as características de capacidade de medir distâncias entre 15cm e 11m, como o *Polaroid* 6500 (www.senscomp.com).

6.3. Controle de Navegação

No início da simulação, cada robô recebe uma mensagem do tipo "*desloque-se autonomamente até (x,y)*". Isso faz com que cada robô inicie a navegação em direção a posição solicitada, sendo controlado apenas pela RNA. O controle inteligente de navegação foi desenvolvido com uma RNA do tipo *Multi Layer Perceptron* (MLP) treinada com o algoritmo de aprendizagem *Resilent Backpropagation* (RProp). Esta RNA foi desenvolvida e treinada no SNNS. O controle inteligente realizado pela RNA permite navegação e desvio de obstáculos em um ambiente dinâmico, usando apenas informações disponíveis localmente, obtidas pelos sensores dos veículos.

A RNA tem como entradas as seguintes informações: (*i*) Orientação (ângulo) do veículo, em relação ao plano (x,y), obtido através de uma bússola simulada; (*ii*) Azimute (ângulo para o alvo) do veículo, obtido através da bússola, do GPS e da mensagem contendo a coordenada do alvo; (*iii*) Cinco valores de sensores de distância (sonares). As saídas da RNA são: (*i*) Força a aplicar no motor angular (giro da barra de direção, de -1.5 a 1.5); (*ii*) Força a aplicar no motor linear (torque, de 0.0 a 6.0).

Experiências iniciais mostraram que é imprescindível que o veículo diminua a velocidade em caso de curvas, principalmente quando bruscas. Usamos uma única RNA que controla tanto o giro como o torque, sendo capaz de orientar o veículo e realizar a navegação sem a necessidade de um controle humano ou de uma codificação prévia de um autômato que informe quando desviar de um obstáculo ou como navegar. A base de dados de treinamento foi obtida a partir de um sistema simples baseado em regras. Foi realizado um total de 32 simulações para obter dados; 16 simulações a fim de se obter apenas dados de orientação; outras 8 simulações foram realizadas para obter dados de desvio, usando chão plano e um ambiente com 10% de ocupação por árvores; ainda, outras 8 simulações foram realizadas para obter dados de desvio usando chão irregular e 10% de ocupação por árvores. A base de dados final obtida apresentou 4.985 registros, sendo dividida em 70% para treino e 30% para validação.

As regras criadas para as simulações e que foram utilizadas na extração de dados para o treino da RNA foram as seguintes: (*i*) Para a velocidade: velocidade padrão igual

a 6 unidades de velocidade; quando qualquer sensor de distância percebe um obstáculo a menos de 9 unidades de distância, a velocidade cai a 3 unidades de velocidade; e, quando qualquer sensor de distância percebe um obstáculo a menos de 3 unidades de distância, a velocidade cai a 1,5 unidades de velocidade; (*ii*) Para a orientação: caso a diferença entre orientação do veículo e o azimute for maior que 25º, então giro da barra de direção recebe 0.5; caso a diferença entre orientação do veículo e o azimute for maior que 5º e menor que 25º, então giro da barra de direção recebe 0.2; caso a diferença entre orientação do veículo e o azimute for maior que 0,1º e menor que 5º, então giro da barra de direção recebe 0.1; caso a diferença entre orientação do veículo e o azimute for menor que 0,1º, então giro da barra de direção recebe 0.0; (*iii*) Para o desvio: foram implementadas 24 regras que tem relação com proximidade de obstáculos detectada pelos sensores e respectivos comandos para giro da barra de direção; o sistema prioriza o desvio antes da regra de orientação e prioriza os sensores frontais antes dos laterais, aumentando a importância de acordo com a proximidade percebida. As regras completas podem ser observadas no código fonte do protótipo.

6.3.1. Aprendizagem da RNA

Uma vez obtidos os dados para treino e validação, partimos para definição da topologia da RNA. Testamos 6 diferentes topologias de RNA, com 4, 9, 18, 24, 30 e 36 neurônios na camada oculta. A análise e escolha da melhor RNA foi feita através do Erro Médio Absoluto (E_{MAE}) e do Erro Médio Quadrático (E_{MSE}). O E_{MSE} é fornecido pelo próprio SNNS e apresenta o erro geral da rede, o E_{MAE} é calculado por uma rotina desenvolvida a parte e fornece valores específicos para cada neurônio de saída da RNA.

Inicialmente realizamos 3 treinos de cada topologia de RNA para análise do E_{MAE} sendo cada treino realizado com uma semente aleatória diferente. O E_{MAE} usando nesta validação foi obtido a partir da análise do resultado dos testes das RNA com 5.000, 10.000, 20.000, 40.000, 60.000. 80.000 e 100.000 ciclos. Os primeiros treinos, com as RNAs com 4 e 9 neurônios na camada oculta, mostrou que estas RNAs não apresentam capacidade de aprender. A Tabela 1 apresenta o E_{MAE} destas Redes; podemos ver que o erro permanece constante para todos os ciclos apresentados, bem como para as diferentes sementes aleatórias.

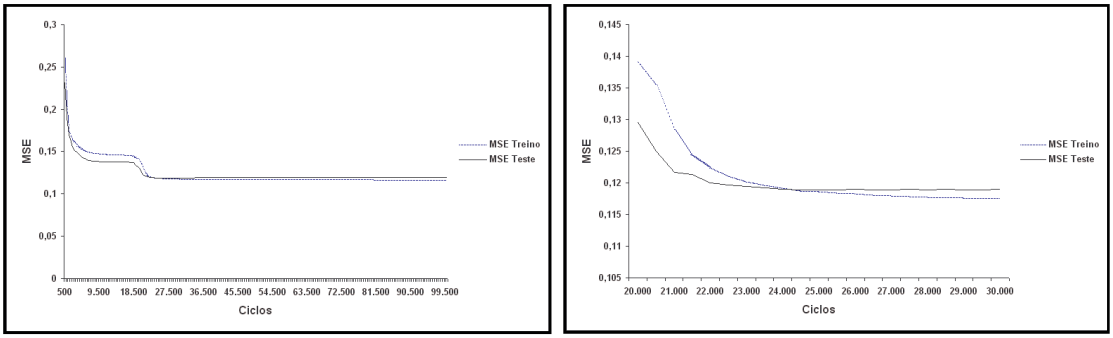
Tabela 1. E_{MAE} medido na base de teste usando as RNAs com 4 e 9 neurônios na camada oculta.

		50.000 ciclos		20.000 ciclos		80.000 ciclos		100.000 ciclos	
RNA	*Seed*	Giro	Veloc.	Giro	Veloc.	Giro	Veloc.	Giro	Veloc.
7x4x2	*a*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000
7x4x2	*b*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000
7x4x2	*c*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000
7x9x2	*a*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000
7x9x2	*b*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000
7x9x2	*c*	0,306	2,000	0,306	2,000	0,306	2,000	0,306	2,000

O treino nas RNAs com 18, 24, 30 e 36 neurônios na camada oculta mostrou que todas estas RNAs apresentam capacidade de aprendizado. A Tabela 2 apresenta o E_{MAE} destas Redes; podemos ver que a RNA com 24 neurônios na camada oculta apresenta menor erro no valor de giro de volante que as outras. Consideramos o valor do giro mais importante que o valor da velocidade. Assim, esta RNA foi escolhida para análise do Erro Médio Quadrático (E_{MSE}), apresentado na Figura 4.

Tabela 2. E_{MAE} medido na base de teste usando as RNAs com 18, 24, 30 e 36 neurônios na camada oculta.

		50.000 ciclos		20.000 ciclos		80.000 ciclos		100.000 ciclos	
RNA	*Seed*	Giro	Veloc.	Giro	Veloc.	Giro	Veloc.	Giro	Veloc.
7x18x2	*a*	0,241	0,185	0,239	0,176	0,233	0,175	0,231	0,159
7x18x2	*b*	0,228	0,113	0,228	0,110	0,228	0,110	0,228	0,111
7x18x2	*c*	0,236	0,153	0,218	0,108	0,218	0,109	0,218	0,109
7x24x2	*a*	0,132	0,111	0,129	0,111	0,129	0,111	0,129	0,111
7x24x2	*b*	0,149	0,132	0,147	0,132	0,147	0,132	0,147	0,132
7x24x2	*c*	0,138	0,121	0,129	0,108	**0,118**	**0,106**	0,118	0,107
7x30x2	*a*	0,190	0,128	0,180	0,123	0,180	0,123	0,180	0,124
7x30x2	*b*	0,183	0,107	0,182	0,103	0,173	0,103	0,173	0,103
7x30x2	*c*	0,168	0,112	0,161	0,108	0,163	0,109	0,164	0,109
7x36x2	*a*	0,152	0,097	0,148	0,097	0,148	0,097	0,148	0,097
7x36x2	*b*	0,159	0,141	0,159	0,119	0,154	0,110	0,154	0,110
7x36x2	*c*	0,146	0,104	0,139	0,106	0,140	0,107	0,140	0,107



(a) (b)

Figura 4. (a) Visão completa (de 0 até 100.000 cliclos); (b) Visão ampliada (de 20.000 até 30.000 cliclos) da inversão das curvas (Ponto Ótimo de Generalização).

Como a Figura 4 mostra que, aproximadamente a partir de 24.000 ciclos existe a inversão das curvas de treino e teste (Ponto Ótimo de Generalização, descrito na Seção 4.1), foi feita uma nova análise do E_{MAE}, com ciclos de teste próximos ao Ponto Ótimo (de 500 em 500 ciclos), a fim de observar exatamente quando o erro no giro começa a subir. A Tabela 3 apresenta os resultados da análise do E_{MAE} próxima ao Ponto Ótimo de Generalização.

Tabela 3. Erro no giro aumenta a partir de 32.500 ciclos.

Ciclos	Giro	Veloc.
23.000	0,12051	0,10561
25.000	0,11898	0,10590
27.000	0,11842	0,10611
29.000	0,11814	0,10620
30.000	0,11805	0,10624
30.500	0,11797	0,10631
31.000	0,11789	0,10631
31.500	0,11780	0,10623
32.000	0,11769	0,10627
32.500	**0,11767**	0,10628
33.000	0,11771	0,10625
34.000	0,11771	0,10626

Podemos ver na Tabela 3 que o ponto de inversão do E_{MAE} no giro ocorre entre as RNAs com 32.500 e 33.000 ciclos. Assim, escolhemos a RNA treinada com 32.500 ciclos para ser convertida em um programa em C e aplicada no controle inteligente dos robôs simulados.

6.3.2. Avaliação da Rede Neural Artificial

Após implementar a RNA selecionada no controle dos robôs móveis, inserindo no sistema de controle o código da RNA, buscamos avaliar se o controle proporcionado pela RNA é eficiente para realizar a navegação entre os pontos iniciais e finais solicitados pelo agente monitor. Assim, realizamos inúmeros experimentos com diferentes quantidades de árvores no ambiente e com diferentes topografias do terreno. Também buscamos avaliar a quantidade de erro que a RNA suporta, mantendo a navegação correta.

Dados os pontos iniciais e finais para um grupo de agentes, a navegação no ambiente desenvolvido pode ser vista na Figura 5. O ambiente foi parametrizado com 4 níveis de ocupação por árvores, calculamos a área de ocupação do total de árvores em relação à área do terreno. As ocupações são aproximadamente 10%; 5%; 2,5% e 0,625%. Resultados serão considerados satisfatórios da seguinte forma: cada veículo deve ser capaz de atravessar uma região simulada representando uma área de cerca de 1*km* utilizando as diferentes taxas de ocupação por árvores. Os veículos não podem colidir com árvores nem com outros veículos.

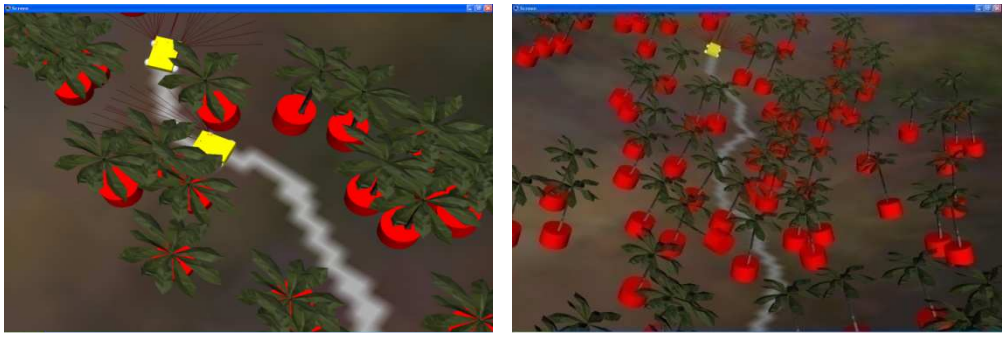


Figura 5. Imagens de simulação com navegação e desvio satisfatórios, para observação dos caminhos realizados pelos robôs móveis.

Os resultados das primeiras simulações, com diferentes tipos de ocupação do ambiente por árvores, podem ser vistos na Tabela 4. Podemos ver nesta Tabela que, para ambientes com ocupação de 5%, ou menos, a RNA foi capaz de realizar a navegação com desvios satisfatórios. Os erros que ocorreram advém da entrada dos veículos em regiões de afunilamento; uma forma inicial para tratar este erro seria a aplicação de capacidade de dar ré, que não foi implementada em nosso modelo.

Tabela 4. Resultado das simulações usando a RNA com diferentes ocupações no ambiente.

Quantidade de simulações de navegação	Área ocupada com árvores	Resultados satisfatórios com a RNA
50	10,00%	42 (84%)
50	5,00%	49 (98%)
50	2,5%	49 (98%)
50	0,625%	50 (100%)

A partir das simulações apresentadas na Tabela 4, escolhemos o ambiente com 5% de ocupação com árvores para realizar novas simulações aplicando ruído simulado nos sensores e nos atuadores.

A Tabela 5 apresenta os resultados de simulações aplicando ruído simulado na leitura de todos os 5 sonares. Essa simulação de ruído foi feita com a aplicação de um

percentual aleatório entre *-Ruído* e *+Ruído* sobre o valor original do sensor. Podemos ver que os resultados são satisfatórios e confiáveis com aplicação de até 10% de ruído. A aplicação de 20% de ruído apresentou falhas em desvios simples.

Tabela 5. Resultado das simulações aplicando ruído na leitura dos 5 sonares.

Quantidade de simulações de navegação	Ruído aplicado nos 5 sonares	Resultados satisfatórios com a RNA
20	Até 5,00%	20 (100%)
20	Até 10,00%	20 (100%)
20	Até 20,00%	18 (90%)

A Tabela 6 apresenta os resultados de simulações aplicando ruído simulado nos valores de comando dos dois atuadores. Essa simulação de ruído foi feita da mesma forma descrita acima. Podemos ver que os resultados são satisfatórios e confiáveis com aplicação de até 10% de ruído. A aplicação de 20% de ruído também apresentou falhas em desvios simples.

Tabela 6. Resultado das simulações aplicando ruído nos 2 atuadores.

Quantidade de simulações de navegação	Ruído aplicado nos 2 atuadores	Resultados satisfatórios com a RNA
20	Até 5,00%	20 (100%)
20	Até 10,00%	20 (100%)
20	Até 20,00%	18 (90%)

A Tabela 7 apresenta os resultados de simulações aplicando ruído simulado de até 100% na leitura de um sensor de distância (escolhido aleatoriamente a cada ciclo). Podemos ver que os resultados são satisfatórios e confiáveis com aplicação de até 100% de ruído em apenas um sensor; quando aplicado em dois sensores, o controle apresenta falha ao percorrer o trajeto.

Tabela 7. Simulações aplicando ruído de até 100% em um sensor aleatório.

Quantidade de simulações de navegação	Quantidade de sonares com até 100% de ruído	Resultados satisfatórios com a RNA
20	1	20 (100%)
20	2	26 (80%)

A Figura 6 apresenta imagens de simulações com aplicação de ruído de até 100% na leitura de um sensor aleatório, como as apresentadas na Tabela 7. Esta Figura apresenta um veículo com sensores puros e um veículo com aplicação de ruído nos sensores; podemos ver a trajetória perfeitamente reta do veículo sem erro nos sensores e a trajetória irregular do veículo com erro simulado.

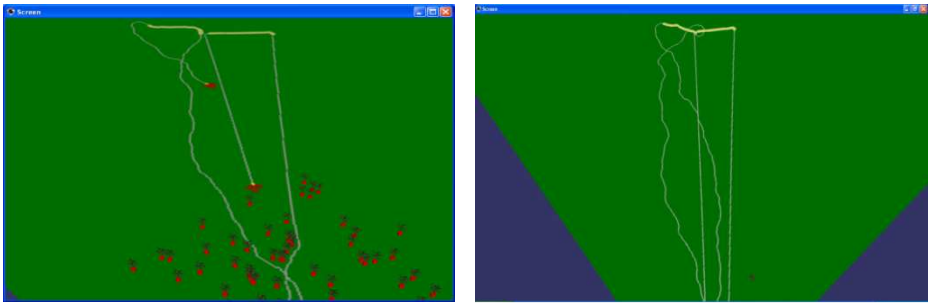


Figura 6. Navegação de dois robôs. Ambas as imagens apresentam trajetórias de ida e volta; um robô com ruído simulado e outro não.

A trajetória (Figura 6) torna-se irregular devido ao fato de o veículo interpretar o ruído como um possível aviso de colisão; o valor do comando aplicado no giro das rodas é maior no desvio do que para a reorientação para o alvo.

7. Conclusão

O objetivo deste artigo foi detalhar o modelo, a implementação e a avaliação da eficiência de uma Rede Neural Artificial (RNA) aplicada ao controle de navegação de robôs móveis fisicamente simulados. A atuação dos robôs se dá em um ambiente virtual de simulação realística, que apresenta obstáculos estáticos e dinâmicos; desenvolvido em C++ com OSG, ODE e Demeter. As entradas da RNA são obtidas através de sensores presentes em cada um dos robôs (e.g. GPS, bússola, sonar). O modelo proposto de RNA mostrou ser capaz de controlar os atuadores do robô móvel desenvolvido usando apenas informações disponíveis localmente, obtidas pelos sensores dos veículos em um ambiente dinâmico, com terreno irregular e com as propriedades físicas disponíveis pela ODE. O controle robusto foi validado com a demonstração de resultados aplicando ruídos nos sensores e nos atuadores dos robôs móveis. Os resultados das simulações demonstram que a RNA é capaz de controlar de modo satisfatório os robôs móveis, e que o modelo de RNA proposto pode tanto ser usando no SMA de monitoração e combate de incêndios como em qualquer outro sistema que necessite de controle reativo de navegação.

Referências

Antunes, M. A. H. (2000) “Uso de satélites para detecção de queimadas e para avaliação do risco de fogo”, Ação Ambiental, 12:24-27.

Batista, A. C. (2004) “Detecção de incêndios florestais por satélite”, Revista Floresta, Curitiba/PR, v. 34, p. 237-241.

Barros, L. M., Gonzaga, L., Raposo, A. B. (2007) “Open Scene Graph: conceitos básicos e aplicações em realidade virtual”, Tutorial on IX SVR, LNCC.

Bekey, G. A. (2005) “Autonomous Robots: From Biological Inspiration to Implementation and Control”. Cambridge, USA: The MIT Press, 577 p.

Braga, A. P.; Carvalho, A. P. L. F. de; Ludermir, T. B. “Redes Neurais Artificiais: Teoria e aplicações” Rio de Janeiro, RJ, Brasil: LTC, 2000. 262 p.

CPTEC/INPE. (2006) “Centro de previsão do tempo e estudos climáticos - Instituto nacional de pesquisas espaciais”. Disponível em <www.cptec.inpe.br/queimadas>.

Dudek, G., Jenkin, M. (2000) “Computational Principles of Mobile Robotics” Cambridge, London, UK: The MIT Press, 280 p.

Fahlman, S. E. (1988) “Computer Science Technical Report: An empirical study of learning speed in back-propagation networks” Carnegie-Mellon University, USA.

Go, J., Browning, B., Veloso, M. (2004) “Accurate and flexible simulation for dynamic, vision-centric robots”. In: International Joint Conference on Autonomous Agents.

Hornik, K.; Stinchcombe, M.; White, H. (1989) “Multilayer feedforward networks are universal approximators”. Neural Networks (ISSN:0893-6080), v. 2, p. 359-366.

JPL/NASA (2007) “Jet Propulsion Laboratory/NASA”, wwwrobotics.jpl.nasa.gov.

Kartalopoulos, S. V. (1996) “Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications” New York, USA: IEEE Press, 232 p.

LIF - Laboratório de Incêndios Florestais (2006) “Pesquisas e projetos em prevenção e combate de incêndios florestais”, UFPR, www.floresta.ufpr.br/~firelab, Setembro.

Marchi, J. (2001) “Navegação de robôs móveis autônomos: estudo e implementação de abordagens”, Dissertação de Mestrado, Universidade Federal de Santa Catarina.

Mcculloch, W.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, v. 5, p. 115-133, 1943.

Nikolopoulos, C. (1997) “Expert Systems - Introduction to First and Second Generation and Hybrid Knowledge Based Systems” New York, USA: Marcel Dekker Inc. Press.

Osório, F. S., et. al. “Seva3d: Using artificial neural networks to autonomous vehicle parking control” In: Proceedings of IEEE World Congress On Computational Intelligence, International Joint Conference On Neural Networks (IJCNN’06). Vancouver, Canada: IEEE Press, 2006. p. 4704–4711.

Pessin, G., et. al. (2007) “Simulação Virtual de Agentes Autônomos para a Identificação e Controle de Incêndios em Reservas Naturais”, In: IX Symposium on Virtual and Augmented Reality (SVR’07), LNCC, v.1, p. 236-245.

Pessin, G., et al. (2007a) “Utilizando agentes autônomos com aprendizado para a identificação e combate de incêndios em Áreas florestais” In: Anais do VII Simpósio de Informática do Planalto Médio (SIPM’07), Universidade de Passo Fundo (UPF).

Pessin, G., et al. (2007b) “Utilizando redes neurais artificiais no controle de robôs móveis aplicados ao combate de incêndios florestais” In: Anais do XVI Seminário de Computação (SEMINCO), Universidade Regional de Blumenau (FURB), p. 19-30.

Pessin, G., et al. (2007c) “Evoluindo estratégias de posicionamento em um sistema multi-robótico aplicado ao combate de incêndios florestais” Revista Hífen, Uruguaiana, RS, Brasil, v. 31, n. 59/60, p. 78-84.

Pfeifer, R.; Iida, F.; Bongard, J. (2005) “New robotics: Design principles for intelligent systems”. Artificial Life, v. 11, p. 99-120.

Pfeifer, R., Scheier, C. (1994) From perception to action: The right direction? In: Proceedings of IEEE International Conference: From Perception to Action. Lausanne, Switzerland: IEEE Computer Society Press, p. 1-11.

Pfeifer, R., Scheier, C. (1997) Sensory-motor coordination: The metaphor and beyond. Robotics and Autonomous Systems, v. 20, n. 2, p. 157-178.

Pfeifer, R., Scheier, C. (1999) “Understanding Intelligence”. The MIT Press.

Remmel, T. K., Perera, A. H. (2001) “Fire mapping in a northern boreal forest: assessing AVHRR/NDVI methods of change detection”, Forest Ecology and Management 152:119-129.

Rezende, S.O. (2003) “Sistemas inteligentes: fundamentos e aplicações”, Manole, SP.

Riedmiller, M.; Braun, H. (1994) “Technical Report: RPROP, Description and Implementation Details” Universitat Karlsruhe, Deutschland.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. “Learning Internal Representations by Error Propagation” Cambridge, MA, USA: The MIT Press, 1986.

Rumelhart, D., McClelland, J., (1986) “Parallel Distributed Processing”, MIT Press.

Smith, R. (2006) “Open Dynamics Engine v0.5 User Guide”.

Widrow, B.; Hoff, M. E. (1960) “Adaptative switching circuits” In: Institute of Radio Engineers (IRE) Western Electronic Show and Convention Record (WESCON). New York: Institute of Radio Engineers, p. 96–104.