Controle da Tarefa de Estacionamento de um Veículo Autônomo através do Aprendizado de um Autômato Finito usando uma Rede Neural J-CC

F. Osório, F. Heinen, L. Fortes
UNISINOS – Universidade do Vale do Rio dos Sinos – Centro de Ciências Exatas e Tecnológicas
Mestrado em Computação Aplicada - Av. Unisinos, 950 – São Leopoldo, RS – 93022-000
{osorio,farlei}@exatas.unisinos.Br – Luciane.fortes@gm.com

Abstract

This paper presents the SEVA system (Autonomous Vehicle Parking Simulator). This tool was used to implement a robust control system for autonomous vehicle parking based on a FSA (Finite State Automata) and also based on an ANN (Artificial Neural Networks). Initially a FSA was developed in order to control the vehicle performing a parallel parking manoeuvre. Therefore, a special model of ANN, called J-CC (Jordan-Cascade Correlation), was designed to learn how to control the vehicle based onto the behavior of the FSA. The Neural Controller (SEVA-N) uses the infrared sensorial information and the present state of the automata to control the speed (car velocity), the angle of the wheels (car orientation) and to determine the next state of the automata. At present, the simulation shows that our system can safely park the vehicle, using both methods: FSA or J-CC Nets. We conclude that the proposed ANN model reach our needs and also can be applied in more complex tasks.

1. Introdução

Este trabalho visa dotar um veículo autônomo de um sistema de controle robusto, baseado em um autômato finito (FSA – *Finite State Automata*), a fim de realizar o estacionamento deste veículo em uma vaga paralela. O sistema de controle será obtido pelo aprendizado de uma rede neural J-CC inspirada nos modelos de Jordan[2, 20] e de Fahlman[3].

Os veículos autônomos (e.g. carros dirigidos de forma autônoma[15]) tem atraído a atenção de um grande número de pesquisadores da área de Inteligência Artificial, devido ao desafio que este novo domínio de pesquisas nos propõe: dotar sistemas de uma capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos. Os veículos autônomos devem poder "sentir" o ambiente em que estão inseridos através da leitura de seus sensores (e.g. sensores infra-vermelho, lasers, bumpers, câmeras de vídeo, etc), e através desta percepção sensorial eles podem planejar melhor as suas ações [8, 5].

Podemos citar alguns exemplos de veículos autônomos que se tornaram bastante conhecidos na atualidade: o sistema desenvolvido pelo NavLab da CMU [1, 15] que é capaz de conduzir uma caminhonete pelas estradas americanas; o robô do tipo rover enviado para Marte pela NASA [18]; o robô Dante que explora vulcões [7]; o sistema de controle de um veículo Ligier elétrico desenvolvido pelos pesquisadores do INRIA na França [14, 17]. Todos estes sistemas possuem em comum a capacidade de receber leituras de sensores que lhes dão informações sobre o ambiente em que estão inseridos e de modo semi ou completamente autônomo, geram os comandos que fazem com que eles se desloquem em um ambiente de modo seguro: sem se chocar contra obstáculos ou colocar em risco sua integridade ou a dos diferentes elementos presentes no ambiente.

A partir de trabalhos de pesquisa que vem sendo desenvolvidos pelo Grupo de Inteligência Artificial do PIPCA[19] relacionados aos veículos autônomos, foram criadas aplicações (simuladores) na área de robótica autônoma móvel. Destaca-se particularmente o desenvolvimento dos projetos HMLT (Hybrid Machine Learning Tools) e COHBRA / HyCAR (Controle Híbrido Inteligente de Robôs Autônomos), que contam com a participação dos membros do Mestrado em Computação Aplicada da Unisinos.

Para realizar este trabalho, foi implementado um simulador de um veículo ("tipo carro") equipado com sensores de proximidade. A seguir, foi desenvolvido um sistema de controle reativo (sensorial-motor) usando uma rede neural, capaz de receber os sinais dos sensores e gerar os comandos de acionamento do veículo (motor e direção). O sistema de controle aprendeu a controlar o veículo a partir de uma série de exemplos obtidos pelo uso de um controlador baseado em um autômato finito (FSA)[10]. Nosso principal desafio foi o de encontrar um modelo de rede neural que fosse capaz de aprender a realizar corretamente a tarefa de estacionamento do carro. O uso de uma rede neural deverá permitir a obtenção de um sistema de controle mais robusto e menos sensível a ruídos externos dos sensores.

Nas seções seguintes vamos descrever o modelo de simulação adotado, e após descreveremos o controlador baseado em regras, seguido da descrição do funcionamento do controlador baseado na Rede Neural. Por fim, apresentaremos as perspectivas de melhorias que estão sendo estudadas a fim de serem implementadas no sistema.

2. Simulador SEVA

Neste trabalho, optamos pela implementação de um sistema capaz de controlar um veículo autônomo similar a um carro (robô não-holonômico – forma retangular) equipado com um conjunto de seis sensores de proximidade infra-vermelhos de baixo custo. Baseando-se nos modelos da cinemática de um veículo real [4] e dos sensores [9] (modelo este usado em simuladores amplamente adotados pela comunidade científica internacional, como o Khepera-SIM), implementamos o nosso simulador procurando criar um modelo que fosse o mais próximo possível de um sistema real. O simulador foi denominado de SEVA – Simulador de Estacionamento de Veículos Autônomos, foi desenvolvido em Visual C++ de forma modular, onde podem ser facilmente adaptados modelos da cinemática do veículo, dos sensores e sistemas de controle dos atuadores (aceleração e rotação da direção). Sendo assim, este simulador permite que sejam definidas diferentes configurações de carros, bem como de módulos de controle aplicados em diferentes tarefas.

Nosso trabalho se focalizou especificamente na simulação de um carro com um sistema de controle autônomo desenvolvido a fim de permitir que este carro localize uma vaga de estacionamento paralelo, realizando a seguir o estacionamento de modo automático (sem a intervenção de um condutor humano).

O simulador SEVA oferece três formas distintas de controle do veículo:

(i) Baseado em um autômato finito, associando um conjunto de regras de ação para cada estado do autômato – denominado SEVA-A (autômato);

(ii) Baseado em comandos do usuário digitados pelo teclado ou via *joystick* – denominado SEVA-H (humano);

(iii) Baseado em uma Rede Neural Artificial com aprendizado supervisionado usando uma rede J-CC inspirada nos modelos de Jordan [2, 20] e no modelo Cascade-Correlation [3] – denominado SEVA-N (neural).

Os sistemas de controle SEVA-A e SEVA-N usam as informações provenientes dos sensores para definir o comportamento dos atuadores que irão controlar a direção do deslocamento, a velocidade do veículo e a rotação da direção.

O simulador permite a visualização, em uma janela gráfica, da trajetória e do comportamento do veículo durante o processo de estacionamento. Uma análise visual da evolução do comportamento do veículo controlado pelo sistema permite assim verificar se as tarefas de estacionamento foram realizadas com sucesso.

2.1. Componentes do SEVA

O simulador SEVA (Simulador de Estacionamento de Veículos Autônomos) possui os seguintes componentes principais:

- Modelo de simulação dos sensores;
- Modelo de simulação da cinemática do veículo (deslocamento do carro);
- Comandos do atuador relacionado ao deslocamento (avançar / recuar e velocidade);
- Comandos do atuador relacionado ao giro do veículo (rotação da direção).

A figura 1 apresenta a interface gráfica do simulador SEVA, onde na parte superior encontramos uma zona de status que indica: o estados dos sensores, o estado da velocidade de deslocamento do veículo e o estado da rotação do veículo. Na janela principal também podemos visualizar uma representação simplificada do cenário hipotético no qual o veículo está inserido. Um desenho da trajetória do veículo foi sobreposto sobre a imagem da interface gráfica, permitindo visualizar o comportamento típico do veículo, obtido em modo de controle autônomo usando o autômato (SEVA-A), sem interferência humana.

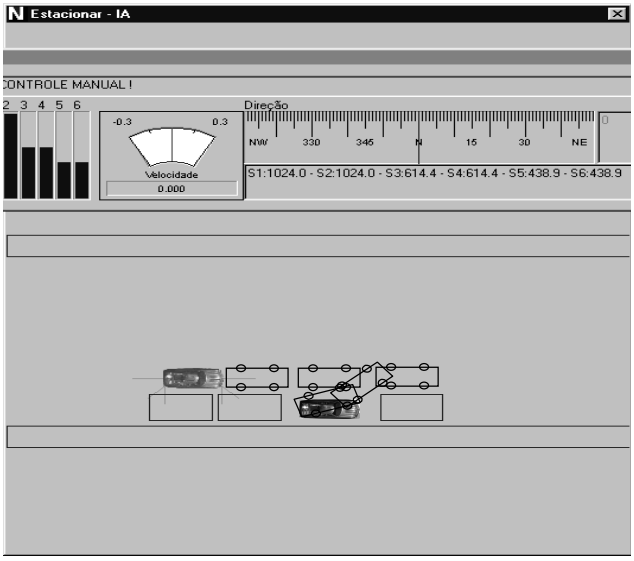

Figura 1. Simulador SEVA.

2.2. Modelo dos Sensores

Os sensores de distância simulam sensores infra-vermelhos, sendo capazes de determinar a distância entre o carro e os obstáculos presentes no ambiente: outros

carros e a calçada. Os seis sensores utilizados estão distribuídos em pontos estratégicos do carro (vide figura 2), estando localizados na frente, na traseira e na lateral direita do carro. Foram implementados apenas os sensores da lateral direita do veículo, pois nossos estudos atuais se restringiram ao estacionamento em vagas paralelas localizadas no lado direito do carro, caso típico em pistas de duas vias.

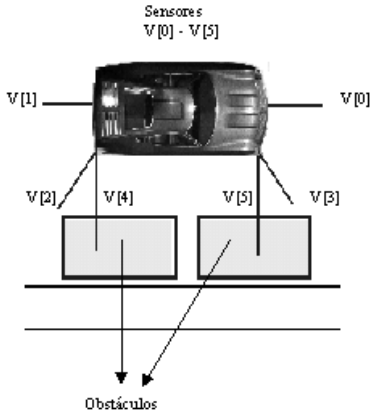

Figura 2. Sensores do Veículo.

Os sensores retornam um valor numérico entre 0 e 1024, onde o valor 0 indica que houve um contato com o objeto (colisão) e 1024 indica que não há obstáculos "visíveis" no campo de percepção do sensor (caminho livre). O simulador apresenta este campo de percepção representado graficamente por segmentos de reta, onde quando este segmento intercepta um objeto isto é convertido para um valor entre 0 e 1024 representando a distância relativa entre o sensor e o objeto interceptado. Os valores numéricos dos sensores também são apresentados na janela gráfica a fim de permitir um melhor acompanhamento das informações referentes a simulação do estacionamento.

2.3. Modelo da Cinemática do Veículo

A movimentação do veículo respeita o modelo da cinemática de um carro, conforme modelo também adotado em [4]. Neste modelo é simulado um veículo representado por um volume retangular suportado por quatro rodas, onde as rodas traseiras possuem um eixo fixo e as rodas dianteiras podem ser direcionadas, através do giro da barra da direção. As coordenadas do veículo são definidas por Vp = (X, Y, θ), onde 'X' e 'Y' definem o ponto médio do eixo traseiro do veículo e 'θ' indica a sua orientação (ângulo em relação a direção de referência). A velocidade do veículo é representada por 'V' e o ângulo de giro (rotação) da direção é denotado por 'Φ'. O valor de 'L' indica o tamanho do eixo das rodas. O deslocamento do veículo é descrito pelas seguintes equações: (Eq. 1-3)

$X = V * \cos(\Phi) * \cos(\theta)$	Eq. 1
$Y = V * \cos(\Phi) * \cos(\theta)$	Eq. 2
$\theta = V / L * \sin(\Phi)$	Eq. 3

A figura 3 apresenta um esquema do modelo da cinemática do veículo que foi adotado em nossas simulações.

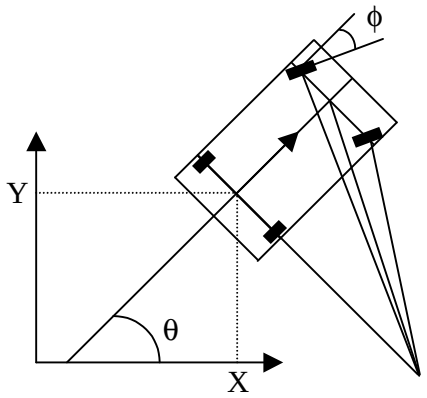

Figura 3. Cinemática do veículo com eixo de rotação dianteiro.

2.4. Atuador de Deslocamento

O deslocamento do veículo é obtido na simulação através do controle de sua velocidade 'V'. Para que fosse possível controlar o veículo na tarefa de estacionamento, definiu-se cinco estados referentes a sua velocidade de deslocamento: Avanço_Rápido (AR), Avanço_Lento (AL), Parado (P), Ré_Lenta (RL), Ré_Rápida (RR). Cada um destes estados é associado a um valor numérico correspondente a velocidade real adotada pelo veículo.

2.5. Atuador de Rotação

O giro da direção, assim como a velocidade do veículo é definido através de um conjunto de quatro valores adotados como ângulos de rotação: Giro_Esquerda_Max (GEM), Giro_ Esquerda_Pequeno (GEP) , Direção_Reta (DR) , Giro_Direita_Pequeno (GDP), Giro_ Direita_Max (GDM). Cada um destes estados é associado a um valor numérico que corresponde ao giro (ângulo) da direção adotado pelo veículo.

3. Controle baseado em um Autômato Finito: Simulador SEVA-A

O controle do estacionamento do carro foi implementado através da criação de um autômato, composto por um conjunto de 7 estados que buscam descrever a evolução do comportamento de um motorista

ao realizar esta tarefa. Um condutor experiente foi o responsável pela criação do autômato e do conjunto de ações associadas a cada estado deste autômato. Este autômato foi testado e refinado até que fosse capaz de realizar a tarefa corretamente.

O esquema final do autômato que foi criado está representado na figura 4, sendo composto dos seguintes estados: procurando_vaga, posicionando, entrando_vaga, posicionando_vaga, otimizando_vaga, alinhando e parado.

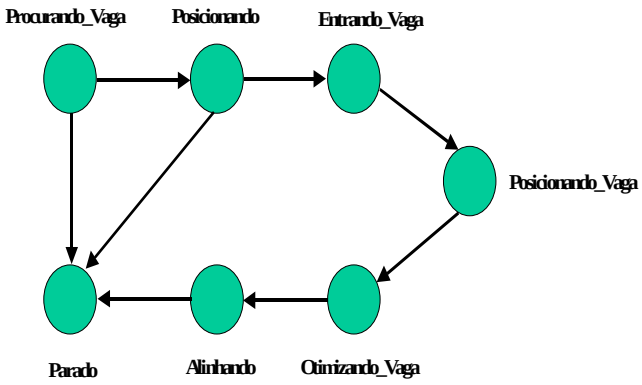

Figura 4. Autômato com os estados usados no processo de estacionamento do carro.

O uso de autômato permitiu que fosse simplificado o tratamento do problema, visto que um especialista reage de modo diferente aos estímulos sensoriais do ambiente, segundo o estado em que se encontra. Por exemplo, ao procurar uma vaga paralela estamos usualmente andando em linha reta e buscando localizar uma "abertura" entre dois carros, e quando vamos colocar o carro na vaga estamos usualmente dando uma ré, controlando os veículos ao nosso lado de modo a girar corretamente a direção e entrar na vaga sem bater neles. É importante salientar que o estado sensorial ao se procurar uma vaga (carros na lateral) e entrar na vaga (carros na lateral) são praticamente iguais, sendo que a diferença está no fato de que ao entrar na vaga, esta já foi localizada anteriormente.

Constatamos também que, para que fosse possível realizar a tarefa de estacionamento, era necessário adicionar um odômetro (deslocamento relativo), o que nos permitiu controlar o quanto o carro se deslocou. O odômetro é usado apenas em casos onde, após acharmos uma vaga, é necessário primeiro avançar uma certa distância antes de entrar de ré na vaga (no resto da simulação ele permanece zerado). O comportamento final do veículo controlado pelo autômato é apresentado na figura 1.

O autômato e as ações associadas a cada estado podem ser codificados sob a forma de um conjunto de regras. As regras consultam o estado atual do autômato, dos sensores e do odômetro, e geram como efeito o controle dos atuadores de deslocamento (velocidade – controlada pela variável *Speed*) e de rotação (giro da direção – controlado pela variável *Turn*). O sistema SEVA_A completo possui um total de 30 regras, onde listamos a seguir alguns exemplos destas regras empregadas no simulador:

```
Se Estado_Atual(Procurando_Vaga) e
   Próximo_ao_Obstáculo(V[4]) e
   Próximo_ao_Obstáculo(V[5])
Então Speed = Avanço_Rápido e
      Turn =  Direção_Reta;
Se Estado_Atual(Procurando_Vaga) e
   Longe__do_Obstáculo(V[2]) e
   Longe__do_Obstáculo(V[3]) e
   Longe__do_Obstáculo(V[4]) e
   Longe__do_Obstáculo(V[5])
Então Troca_Estado(Posicionando) e
      Inicializa(Odômetro);
Se Estado_Atual(Posicionando)
Então Speed = Avanço_Rápido e
      Turn = Direção_Reta;
Se Estado_Atual(Posicionando) e
   Longe_do_Obstáculo(V[4]) e
   Deslocamento_Suficiente(Odômetro)
Então Estado_atual(Entrando_Vaga) e
      Inicializa(Odômetro);
Se Estado_Atual(Entrando_Vaga)
Então Speed = Ré_Rápida e
      Turn = Giro_Esquerda_Max;
```

Quadro 1 – Exemplo das regras usadas no sistema de controle do veículo.

4. Controle baseado em uma Rede Neural Artificial J-CC

Os resultados obtidos com o controle baseado no autômato nos levaram ao estudo de uma alternativa para a aquisição de conhecimentos, visto que o processo de codificação de regras é muito trabalhoso, além de não garantir uma grande robustez do sistema na presença de ruídos (e.g. valores inexatos dos sensores e situações não previstas). Buscou-se então a implementação de uma solução baseada no "aprendizado prático" a partir de uma base de exemplos de condução do veículo autônomo. A solução adotada foi a utilização de Redes Neurais Artificiais (RNA) [2, 11, 16], com aprendizado supervisionado, capazes de aprender bases de exemplos que demonstram como se deve estacionar o carro.

O primeiro passo foi a construção das bases de exemplos empregadas no aprendizado da Rede Neural. Adaptou-se o simulador de modo a gerar um "arquivo de log", contendo o registro do estado dos sensores, estado do autômato e comandos enviados aos atuadores (velocidade e rotação), gerados pelo SEVA-A. O arquivo gerado permite que seja treinada uma rede neural, que inicialmente, deve ser capaz de reproduzir o compor-

tamento do autômato. Posteriormente poderemos usar o SEVA-H para gerar novos exemplos da tarefa de estacionamento, com o veículo sendo controlado por um ser humano. O uso de uma rede neural incremental [13], como a adotada no SEVA-N, possibilita que seja feito um refinamento contínuo dos conhecimentos adquiridos.

O modelo de RNA adotado foi o J-CC (Jordan-Cascor) um novo modelo neural derivado do modelo de Jordan [2, 20] e do modelo Cascade-Correlation [3, 13], ambos baseados em redes do tipo MLP (Multi-Layer Perceptron) [2, 11]. O princípio de funcionamento da rede J-CC é o uso de um conjunto de entradas que servem para indicar o estado atual da rede (compatível com o autômato) e um outro conjunto de saídas da rede que servem para indicar o estado que a rede deve assumir (novo estado do autômato) em função de suas entradas. Deste modo a rede neural é capaz de receber como entrada o seu estado atual e o estado dos sensores (vide figura 5) e determinar em sua saída se deve manter o estado atual ou passar para um novo estado, baseada nas informações que recebeu. O próximo estado (atual ou novo), obtido na saída da rede é então re-injetado na entrada da rede, obtendo assim um modelo similar ao proposto por Jordan [20] onde a saída da rede é re-injetada na entrada formando uma recorrência e obtendo assim as chamadas unidades de contexto. É através das unidades de contexto que temos uma informação sobre o estado atual do autômato/rede.

A rede J-CC emprega o algoritmo de aprendizado e ajuste dos pesos da rede do tipo Cascade-Correlation no lugar do tradicional algoritmo de Back-Propagation [16] devido ao fato deste outro algoritmo simplificar a definição da rede (não precisamos indicar o número de neurônios da camada oculta) e por ser um algoritmo comprovadamente mais eficiente que o Back-Propagation [6, 3]. O algoritmo Cascade-Correlation é um algoritmo que implementa uma rede neural incremental, tornando o J-CC uma rede com propriedades bastante interessantes para o aprendizado de autômatos.

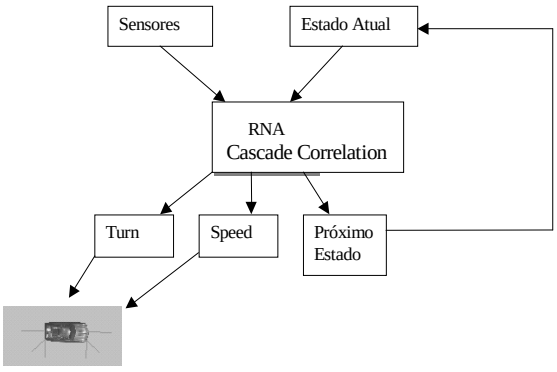

Figura 5. Descrição da Rede Neural Artificial utilizada para controlar o veículo.

As variáveis (atributos) de entrada da rede que empregamos foram: o estado dos seis sensores de proximidade, o odômetro e uma indicação do estado atual do processo de estacionamento Na saída da rede iremos obter o estado dos atuadores (velocidade e rotação), assim como uma indicação do próximo estado do processo de estacionamento. Deste modo, a rede irá controlar os atuadores, mas também irá decidir quando devemos passar de um estado a outro no autômato de controle. A figura 5 apresenta o esquema das entradas e saídas da RNA adotada. Note que o estado previsto pela rede pode ser re-inserido nela como sendo o estado atual, sendo então usado no instante de tempo seguinte.

Salientamos a importância de se conhecer o estado atual, para interpretar corretamente os sensores, pois:

(i) Quando estamos procurando uma vaga, temos os sensores laterais indicando a proximidade de um carro, onde isto indica que devemos continuar a avançar até encontrar uma vaga, quando então os sensores laterais irão indicar que o obstáculo não está mais tão próximo;

(ii) Quando estamos entrando na vaga, temos os sensores laterais também indicando a proximidade de um carro (como na situação anterior), mas isto indica que devemos engatar uma ré para colocar o carro na vaga previamente localizada.

Apesar de precisarmos de uma indicação do estado do processo de estacionamento em que nos encontramos, a rede neural é capaz de aprender a detectar a passagem de um estado a outro (gerando o estado próximo estado), e uma vez terminada a fase de aprendizado, ela será capaz de realizar o estacionamento de forma autônoma sem a intervenção humana.

A base de aprendizado utilizada para treinar a rede neural usou uma codificação binária do tipo "1 entre N" para representar a maioria das entradas e saídas da rede:

- Entrada: o estado atual (7 estados = 7 entradas binárias, com 1 bit representando cada estado); o estado dos sensores (6 sensores = 6 entradas numéricas com valores entre 0 e 1024); odômetro (numérico)
- Saída: a velocidade (5 velocidades = 5 saídas binárias: AR, AL, P, RL, RR), o giro da direção (4 rotações possíveis = 4 saídas binárias: GEM, GEP, DR, GDP, GDM) e o próximo estado (que também codifica 7 estados possíveis em 7 saídas binárias).

Os resultados das simulações do aprendizado da RNA, para um conjunto de 10 simulações que realizamos, foram os seguintes: obtivemos uma taxa de aprendizado com um acerto médio de 98.64% nas respostas da rede; foram usadas (inseridas) apenas 2 unidades na camada intermediária e o número médio de épocas de aprendizado foi de 572 épocas. Como foi usada uma base de 392 exemplos, notamos que apenas cerca de 6 exemplos não foram corretamente aprendidos pela rede. Estes exemplos não comprometem a capacidade da rede em trocar de estado (momento crítico no controle do veículo).

5. Conclusões e Perspectivas

Este trabalho teve como objetivo o desenvolvimento de um simulador para o controle de veículos autônomos em uma tarefa de estacionamento em vagas paralelas. Os resultados obtidos nas simulações realizadas com a ferramenta que desenvolvemos, o SEVA, demonstraram que o sistema de controle, seja ele implementado através de um autômato ou através de uma Rede Neural Artificial, possui a capacidade de controlar corretamente o veículo, cumprindo seu objetivo principal: estacionar o veículo na vaga, sem bater nos demais obstáculos que estão ao seu redor. Uma verificação tanto numérica quanto visual permitiu constatar que a tarefa foi corretamente executada pelos dois tipos de sistemas de controle.

Futuramente pretendemos melhorar o aprendizado da rede neural através da inclusão de mais exemplos, com a inserção proposital de ruído e também com o uso de exemplos obtidos através do controle pelo SEVA-H (controle humano). A rede J-CC se mostrou tão robusta que está sendo estudada a possibilidade de se alterar o algoritmo de funcionamento desta rede, de modo a permitir a inserção de novos estados no autômato (novas entradas e saídas) mesmo durante o aprendizado.

Agradecimentos: *Agradecemos a Fapergs, ao CNPq e a Unisinos pelo financiamento de nossas pesquisas e dos bolsistas envolvidos no desenvolvimento destes trabalhos*.

6. References

[1] Batavia, Parag; Pomerleau, Dean & Thorpe, Charles. Applying Advanced Learning Algorithms to ALVINN. CMU Technical Report CMU-RI-TR-96-31. Carnegie Mellon University. Pittsburgh. 1996.

[2] Braga, Antônio de Pádua; Ludermir, Tereza Bernarda; Carvalho, André Carlos Ponce de Leon Ferreira. Redes Neurais Artificiais, Teoria e Aplicações; LTC Editora, Rio de Janeiro. 2000.

[3] Fahlman, Scott E. & Lebiere, Christian. The Cascade-Correlation Learning Architecture. Carnegie-Mellon University – CMU, Computer Science Techincal Report. CMU-CS-90-100. 1990.

[4] Garnier, Philippe; Fraichard, Thierry; Laugier, Christian; Paromtchik, I. and Scheuer. Motion Autonomy Through Sensor-Guided Manoeuvres. IEEE-RSJ International Conference on Intelligent Robots and Systems, Procedings of the Intelligent Cars and Automated Highway Systems Workshop. Sept. 1999. Grenoble, France.

[5] Heinen, Farlei José. Robótica Autônoma: Integraçào entre Planificação e Comportamento Reativo; UNISINOS Editora, São Leopoldo, Novembro, 1999.

[6] Joost M, W.Schiffmann, R. Werner. Comparison of Optimized Backpropagtion Algoritms. Presented at ESANN 93, Brüssel.

[7] Lemonick, Michel. Dante Tours the Inferno. Time Magazine – Time Domestic/Science. Vol. 144, No. 7. August 15, 1994.

[8] Medeiros, Adelardo. Introdução a Robótica. ENA-98 Encontro Nacional de Automática (50º Congresso da SBPC). Natal, RN. pp.56-65. 1998.

[9] Michel, Olivier. Khepera Simularor Version 2.0 – User Manual. Université de Nice – Sophia Antipolis. Laboratoire I3S - CNRS, France / EPFL – Lausanne, Swiss. March 1996.

[10] Menezes, Paulo Blauth. Linguagens Formais e Autômatos. Instituto de Informática – Série: Livros didáticos. 3ª. edição. Editora Sagra Luzzatto. 2000.

[11] Osório, Fernando S. Redes Neurais Artificiais: Do aprendizado Natural ao Aprendizado Artificial. Tutorial – I Fórum de Inteligência Artificial / Ulbra; Canoas, Agosto, 1999.

[12] Osório, Fernando S. & Vieira, Renata. Sistemas Híbridos Inteligentes. Tutorial apresentado no ENIA'99 – Encontro Nacional de Inteligência Artificial, Congresso da SBC, Rio de Janeiro 1999.

[13] Osório, Fernando S & Amy, Bernard. INSS: A hybrid system for constructive machine learning. Neurocomputing, Elsevier Press. Netherlands. 1999.

[14] Paromtchik, I. E. & Laugier, C. Autonomous Parallel Parking of a Nonholonomic Vehicle. Procedings of the IEEE International Symposium on Intelligent Vehicles., pp. 13-18. September, 1996.

[15] Pomerleau, D. Neural network based autonomous navigation. In: Thorpe, C.(Ed). Vision and Navigation: The CMU Navlab. Kluwer Academic Publihers, 1990.

[16] Rumelhart, D.; Hinton, G; Williams, R. Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing. V1. MIT Press, Cambridge. 1986.

[17] Sheuer, A. & Laugier, C. Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots. IEEE-RSJ International Conference on Intelligent Robots and Systems. Victoria, British-Columbia, Canada. Oct. 1998.

[18] Stone, H. W. Mars Pathfinder Microver: A low-cost, low-power Spacecraft. Proceeding of the 1996 AIAA. Forum on advanced developments in Space Robotics. Madison, WI. August 1996.

[19] Grupo de Inteligência Artificial da Unisinos. URL: http://inf.unisinos.br /~osorio/gia.html (Jul. 2001).

[20] Ludik, J.; Prins, W.; Meert, K.; Catfolis, T. A comparative study of fully and partially recurrent networks. International Conference on Neural Networks, 1997. IEEE Press. Volume: 1 , pp. 292 –297. 1997.