Acta Scientiarum. Technology
ISSN: 1806-2563
eduem@uem.br
Universidade Estadual de Maringá
Brasil

Biazus, Claudio Jose; Roisenberg, Mauro
Desenvolvimento de uma arquitetura embarcada reativa para agentes autônomos inteligentes, aplicada ao futebol de robôs
Acta Scientiarum. Technology, vol. 31, núm. 2, 2009, pp. 123-132
Universidade Estadual de Maringá
Maringá, Brasil

DOI: 10.4025/actascitechnol.v31i2.371

Desenvolvimento de uma arquitetura embarcada reativa para agentes autônomos inteligentes, aplicada ao futebol de robôs

Claudio Jose Biazus* **e Mauro Roisenberg**

*Universidade Federal de Santa Catarina, Cx. Postal 476, 88040-900, Florianópolis, Santa Catarina, Brasil. *Autor para correspondência. E-mail: cbiazus@inf.ufsc.br*

RESUMO. Dentre as principais dificuldades encontradas para a construção de sistemas multiagentes em que existe a disponibilidade de um sistema de visão local, como é o caso de algumas categorias de futebol de robôs, destacam-se: necessidade de resposta em tempo real para identificação dos objetos em cena, conhecimento do ambiente, distribuição das competências de controle entre os comportamentos reativos a cargo de cada agente. O presente artigo descreve a implementação de uma arquitetura embarcada reativa para controle de Sistemas Multiagentes equipados com sistema de visão local e dotados de sensores, e sua aplicação em ambientes de futebol de robôs. Neste trabalho, foram descritas as técnicas de processamento digital de imagens e a arquitetura proposta para satisfazer às restrições relacionadas ao problema.

Palavras-chave: arquitetura de controle embarcada, sistemas multiagentes, futebol de robôs.

ABSTRACT. Development of an onboard reactive architecture for intelligent autonomous agents, applied to robot soccer. There are a lot of difficulties that must be faced during the development of multi-agent systems equipped with local vision, as is the case of some robotic soccer leagues, such as: the real-time constraints for recognition of scene objects, acquisition of environment knowledge, and the distribution and allocation of control competencies among reactive behaviors under the responsibility of each agent. This article describes the implementation of an onboard reactive control architecture for a multi-agent system, equipped with local vision camera and local sensors. This multiple agent system is applied in robot soccer environments. The work describes the digital image processing techniques as well as the proposed control architecture that satisfy the constraints related with this kind of application.

Key words: onboard control architecture, multi-agent systems, robot soccer.

Introdução

Durante a década de 1990, surgiram duas organizações científicas, a *RoboCup (Robot Soccer World Cup)* e a *FIRA (Federation of International Robosoccer Association)*, cujo objetivo é promover e desenvolver internacionalmente pesquisas na área de Inteligência Artificial aplicada à robótica móvel inteligente. Para atingir seus objetivos, tanto a *RoboCup* quanto a *FIRA* escolheram o jogo de futebol como a aplicação primária. Estas entidades passaram a organizar campeonatos mundiais de futebol de robôs, uma vez que para formar uma equipe de robôs é necessário o desenvolvimento de diversas tecnologias, tais como: princípios de projeto de agentes autônomos, controle inteligente, processamento de imagens, colaboração multiagentes, raciocínio e mobilidade em tempo real, aquisição e realização de estratégias de jogo.

Muitas dessas tecnologias têm sido desenvolvidas de maneira fortemente acoplada com o aparato sensório (GUPTA et al., 2005). No caso de haver um dispositivo de visão global, tal como uma câmera posicionada sobre o campo de jogo, as estratégias de implementação utilizadas remetem ao desenvolvimento de robôs com baixo grau de autonomia e que têm no "técnico" um sistema de controle centralizado e de caráter deliberativo, localizado junto ao dispositivo de visão global, a sua principal fonte de estratégia e controle. Este "técnico" gera comandos que dirigem os robôs em suas tarefas. Em configurações que não contam com o sistema de visão global, todo o aparato sensório está, nos próprios robôs, que contam então com câmeras de visão local, sensores de proximidade e sensores toque. Neste caso, as equipes normalmente não têm um "técnico", os robôs tendem a ser totalmente autônomos. Os comportamentos e estratégias têm, em geral, caráter fortemente reativo e são implementados diretamente nos robôs, e a cada

momento um comportamento pode emergir sem que haja interferência externa (GUPTA et al., 2005).

Obviamente, quando consideradas de maneira isolada, cada uma dessas estratégias apresenta vantagens e desvantagens no atendimento das restrições de tempo de resposta, capacidade de processamento dos sinais sensórios e de geração de estratégias (SHIM; VADAKKEPAT, 2000).

Em um modelo de arquitetura local ou embarcada, todo o controle, comportamento e ações do agente encontram-se no sistema que está concentrado no próprio robô e, portanto, ele é bastante eficiente para desvio de obstáculos e condução da bola, porém sofre com as restrições de capacidade de memória, processamento e visão incompleta do ambiente. Neste caso, a única forma de obter informações de um meio colaborativo é pela cooperação entre robôs (SHIM; VADAKKEPAT, 2000).

Uma tendência tem sido o desenvolvimento de arquiteturas embarcadas para realizar tarefas de caráter reativo e que envolvam a necessidade de um tempo de resposta imediato, tais como o desvio de obstáculos, a condução da bola e a decisão do momento do chute sejam feitas localmente e de maneira autônoma para cada robô da equipe (MURPHY, 2000).

Uma grande dificuldade no desenvolvimento desse tipo de arquitetura é a definição precisa de quais competências colocar no sistema embarcado dos agentes locais de forma a: minimizar o tamanho e a quantidade de mensagens de comunicação trocadas entre os agentes; permitir que os agentes possam operar no ambiente de maneira satisfatória; e poder atender às restrições presentes em aplicações robóticas reais, tais como a necessidade de as informações serem tratadas em tempo real, sempre considerando a influência de ruído que está presente em cada informação, bem como informações incompletas, pouco precisas, extraídas do ambiente dinâmico e com iluminação não-uniforme (NORBERT, 2006; MURPHY, 1998).

Assim, o objetivo deste trabalho é descrever a metodologia empregada no desenvolvimento da arquitetura embarcada reativa para agentes autônomos inteligentes, aplicada ao futebol de robôs. Esta arquitetura atenderá aos requisitos necessários para o processo de reconhecimento do ambiente, além do poder reativo imediato e tomada de decisões estratégicas.

Este artigo está estruturado da seguinte maneira: após esta introdução, na seção 2, são descritas as entidades que compõem o cenário do futebol de robôs. Na seção 3, é descrita a arquitetura embarcada para sistemas multiagentes. Na seção 4, são apresentados material e métodos e, na seção 5, os resultados e discussão. Finalmente, na seção 6, apresentam-se a conclusão e trabalhos futuros.

Entidades que compõem o cenário do futebol de robôs

Características do cenário

O cenário utilizado para testes, na liga *Small-Size*-(F-180) da *RoboCup*, é composto de um campo plano, de cor verde, com dimensões de 5 m de comprimento x 3,5 m de largura. Para capturar as imagens do ambiente, é utilizado um sistema de visão local acoplado a cada robô *EyeBot*. A bola utilizada é de golfe, de cor laranja. A Figura 1 apresenta o campo e as dimensões.

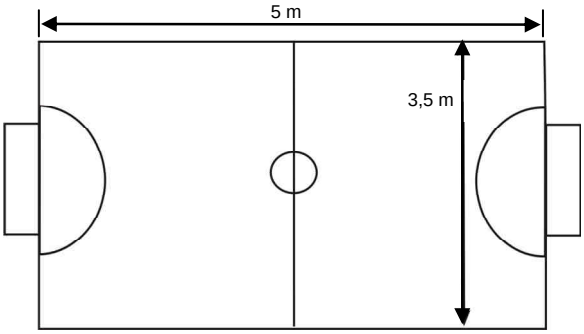

Figura 1. Campo e dimensões.

Características do *EyeBot*

O modelo de robô *SoccerBot Plus (EyeBot)*, conforme pode ser observado pela Figura 2, apresenta as seguintes características: microcontrolador MC68332 Motorola de 32 *bits* com 25 *MHz*, 1 *MB* de memória *RAM*, 512 *Kb* de memória *Flash-ROM,* display de *LCD* com resolução de 128 x 64 pixels para apresentação de gráficos com baixa resolução, portas paralelas e seriais, entradas e saídas digitais e analógicas.

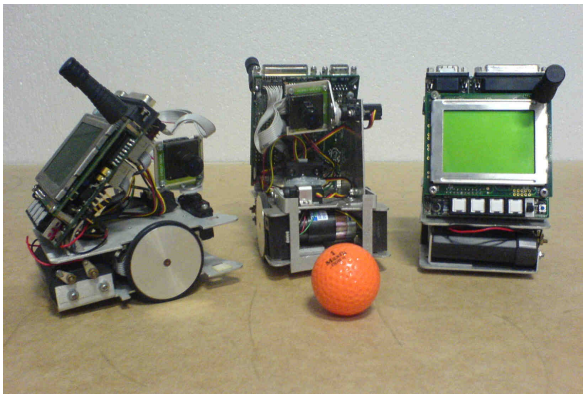

Figura 2. Robôs *EyeBot.*

Este modelo de robôs apresenta também dois motores de passo, dois servos motores, dois *encoders* acoplados a cada uma das rodas, três sensores de aproximação infravermelhos, bateria recarregável com indicador de nível, câmera digital colorida de 24 *bits* com resolução de 80 x 60 pixels, comunicação via rádio com velocidade 9.600 *bps*, atuando em frequência de 433 *MHz*, protocolo de tolerância a falha e configuração automática de rede, mecanismo de chute posicionado na parte da frente.

Arquitetura embarcada para sistemas multiagentes

De acordo com os modelos de arquiteturas que foram propostas por diversos pesquisadores (SHIM; VADAKKEPAT, 2000; WANG et al., 2003; HU; BRADY, 1996; VADAKKEPAT et al., 2007), propõe-se um modelo de arquitetura embarcada. Neste novo modelo, pretende-se integrar o que cada modelo de arquitetura apresenta de melhor; isto significa dizer que se propõe um sistema embarcado no qual, todas as informações, tais como reconhecer o ambiente, os agentes, a bola, controlar a bola, vagar pelo ambiente, desviar de obstáculos, decidir o chute, são realizadas por um sistema de processamento embarcado.

O sistema embarcado recebe informações por meio do conjunto de sensores e da câmera, processa as informações capturadas no próprio sistema embarcado do robô e, imediatamente, envia comandos para seus atuadores. A vantagem em construir um modelo de arquitetura embarcada consiste no fato de que os recursos disponíveis são mais bem aproveitados. Diante de tudo isso, o novo modelo procura atender às reais necessidades de nosso objetivo, que é construir um modelo de arquitetura embarcada para sistemas multiagentes que possa ser aplicado ao futebol de robôs.

A Figura 3 apresenta o modelo da arquitetura embarcada para sistemas multiagentes proposto.

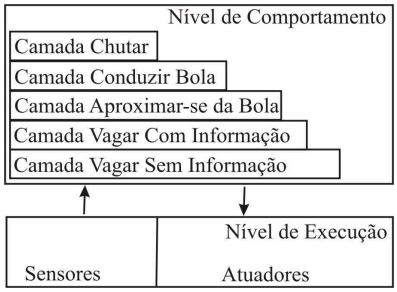

Figura 3. Modelo da Arquitetura embarcada para sistemas multiagentes.

Como podemos observar, a arquitetura proposta está dividida em duas partes. Uma parte é composta por um módulo que corresponde ao comportamento, e a outra é composta por outro módulo que corresponde à execução dos robôs *EyeBot*. Ao definir a arquitetura para os agentes locais, optamos pela implementação de uma arquitetura puramente reativa, baseada na Arquitetura de *Subsumption*, proposta por Brooks na década de 1980 (BROOKS, 1986; MURPHY, 2000). Este modelo reativo apresenta muitas propriedades que são consideradas adequadas, dentre as quais podemos citar: capacidade de execução de forma rápida e, ao mesmo tempo, oferece a eliminação da necessidade de planejamento.

Os comportamentos foram montados conforme a Arquitetura de *Subsumption*, e sua posição está de acordo com a importância na arquitetura e, concomitantemente, com a necessidade de fornecer informações para as camadas superiores. A Arquitetura proposta contém cinco camadas que são: vagar sem informação, vagar com informação, aproximar-se da bola, conduzir bola e chutar, conforme consta na Figura 3. Os módulos que compõem a arquitetura do *EyeBot* representam uma estrutura hierárquica, e a camada que se encontra em mais alto nível assume o papel designado pelas camadas inferiores, quando uma determinada configuração dos sensores indicar uma situação que seja considerada propícia para a atuação. Desse modo, as camadas de mais alto nível suprimem as saídas das camadas dos níveis inferiores.

Material e métodos

Implementação do módulo do *EyeBot*

A implementação do módulo de inicialização do *EyeBot* consiste no processo de checagem dos componentes de hardware do robô e da integração de todas as camadas que fazem parte do software embarcado. A checagem dos componentes de hardware do robô é responsável por indicar se existe algum problema com relação ao rádio, à câmera, aos motores e ao mecanismo de chute. É este módulo que informa se um desses componentes não está conectado ou não funciona. Isto significa dizer que o processo de integração das camadas une todas as camadas que fazem parte do sistema embarcado em um único sistema e, ao mesmo tempo, este sistema é responsável pelo processo de liberação de memória utilizada após a execução de cada camada.

Implementação do sistema de visão

A implementação do sistema de visão para o robô *SoccerBot (EyeBot)* foi desenvolvida com o objetivo de que o robô reconheça apenas três cores: laranja, azul e amarelo. A cor laranja corresponde à cor da bola, e as cores azul e amarelo representam as cores das

metas. A forma utilizada na implementação consiste em procurar pelo tom da cor correspondente e armazenar este valor em uma variável. Após a análise de todo o quadro da imagem, o algoritmo traça um histograma para o eixo horizontal e outro para o eixo vertical, e intercala os histogramas. Dessa forma, é possível obter um valor aproximado do centro da imagem que corresponde à posição aproximada do objetivo.

Para que seja possível construir um histograma do eixo horizontal e outro histograma do eixo vertical da imagem que está sendo analisada, é necessário armazenar a informação referente à coordenada inicial em que o tom da cor correspondente foi encontrado, bem como o ponto que marca a coordenada final. A partir do instante em que se conhece este intervalo, que correlaciona à coordenada em seu ponto inicial e final, e dividindo este intervalo ao meio, é possível obter o ponto da coordenada que corresponde ao centro da imagem da bola no eixo em que está sendo analisado.

O valor obtido da variável que armazena o ponto da coordenada correspondente ao centro da imagem da bola se ajusta ao valor da distância em que a bola se encontra do robô, tanto para o eixo horizontal quanto para o eixo vertical.

A informação referente ao valor da distância em que a bola se encontra do robô é obtida e atualizada toda vez que um quadro é capturado e analisado. Ao mesmo tempo, essa informação é disponibilizada para as camadas: vagar com informação, aproximar-se da bola, conduzir bola e chutar.

Implementação do sistema de sensores

O modelo de robô *SoccerBot Plus (EyeBot)* possui três sensores *PSD (Position Sensitive Detection)* que são responsáveis por indicar a distância em que um obstáculo se encontra do robô. A forma utilizada na implementação consiste, primeiramente, em definir um valor de distância que seja aceitável para detectar todos os obstáculos. Este valor de distância depende da posição em que o sensor está posicionado no *EyeBot*. Por exemplo: para o sensor que está posicionado na parte da frente, é necessário que o sensor indique a presença de um obstáculo com um valor de distância maior. Dessa forma, é possível evitar a colisão. Já para os sensores posicionados à direita e à esquerda, o valor de distância deve ser menor, visto que estes sensores auxiliam apenas em desviar de obstáculos nos movimentos rotacionais do robô.

O processo que envolve a identificação dos obstáculos consiste na pré-definição dos valores que indicam a presença ou a ausência de um obstáculo. Assim, definem-se dois valores para cada sensor: um valor menor, indicando a presença de obstáculo, e outro valor maior, indicando a ausência de obstáculo.

A partir da definição dos dois valores pré-definidos para cada sensor, o algoritmo desenvolvido verifica a leitura do sensor e testa o valor obtido com os valores pré-definidos. Caso o valor lido esteja abaixo do menor valor pré-estabelecido para o sensor, o algoritmo desenvolvido indica a presença de um obstáculo. Por outro lado, se o valor lido está acima do maior valor pré-estabelecido para o sensor, o algoritmo desenvolvido indica a ausência de obstáculo. Já o valor que corresponde ao intervalo que vai do menor valor pré-estabelecido até o maior valor pré-estabelecido para cada sensor faz-se necessário para que a máquina de estados possa ser atualizada. Portanto, um obstáculo pode estar a uma distância considerada: perto, média ou longe.

A informação que corresponde ao valor da distância em que um obstáculo se encontra do robô é atualizada em tempo real. Ao mesmo tempo, esta informação é disponibilizada para o sistema de atuadores.

Implementação do sistema dos atuadores

O sistema implementado para os atuadores é responsável pelo controle dos robôs no ambiente. Este sistema controla os dois motores DC: um é responsável pelo controle da velocidade linear e o outro, pelo controle da velocidade angular. Dessa forma, quando se deseja que o robô se desloque pelo ambiente, é preciso passar em forma de comando um vetor de velocidade que é composto pela velocidade linear e velocidade angular.

A forma utilizada na implementação consiste, primeiramente, em definir os valores, tanto para a velocidade linear quanto para a velocidade angular. A escolha desses valores depende do movimento desejado, isto é, movimentos tais como: seguir em frente, retornar, girar à direita, girar à esquerda etc. No entanto, esses valores nem sempre são fixos, isto significa dizer que, em situações onde o robô necessita ir em direção à bola, ou desviar de obstáculos, esses valores são variáveis.

O algoritmo desenvolvido define a velocidade linear e a velocidade angular para um estado desejado. A partir dessa definição, os atuadores executam este comando, que é executado até que um novo comando seja atribuído.

Implementação da camada vagar sem informação

A camada vagar sem informação representa o comportamento do robô de mais baixo nível na arquitetura proposta. Este comportamento

corresponde a vagar pelo ambiente desviando dos obstáculos que possam surgir; são obstáculos as laterais do campo e os robôs. A Figura 4 apresenta o autômato da camada "vagar sem informação".

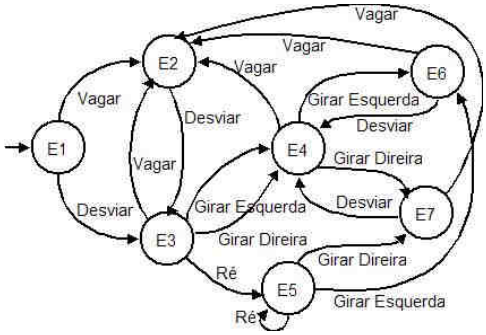

Figura 4. Autômato da camada "vagar sem informação".

O comportamento "vagar sem informação" emerge quando o robô não está de posse da informação que corresponde à posição em que a bola se encontra. Este comportamento é iniciado no estado Parado, que corresponde ao E1. A troca de estados ocorre pelas ações executadas pelo robô.

O sistema implementado para a camada "vagar sem informação" foi desenvolvido com o objetivo de que o robô possa vagar pelo ambiente a partir das informações adquiridas do seu conjunto de sensores. Esta camada é a de mais baixo nível na arquitetura embarcada proposta. Sua aplicação se faz necessária para que o robô possa vagar pelo ambiente, independentemente de possuir a informação da posição em que a bola se encontra e, ao mesmo tempo, de forma reativa.

O algoritmo implementado consiste nos estados possíveis em que o robô pode se encontrar durante uma partida de futebol e em quais ações devem ser executadas para cada estado possível. Estes estados e suas ações são executados até o momento em que o sistema de visão detectar a presença da bola. A partir da informação da posição da bola, o robô passa a ter um objetivo e, dessa forma, é ativada a camada "vagar com informação".

Implementação da camada vagar com informação

A camada "vagar com informação" inicia sua atuação a partir do momento em que o robô ou o sistema do robô está de posse da informação da posição em que a bola se encontra, isto é, esta camada parte do princípio de que o robô está vendo a bola. A Figura 5 apresenta o autômato da camada "vagar com informação".

O sistema implementado para a camada "vagar com informação" foi desenvolvido com o objetivo de que esta camada somente seja acionada a partir do momento em que o robô possui a informação da posição em que a bola se encontra. A informação da posição em que a bola se encontra é fornecida pelo sistema de visão embarcado.

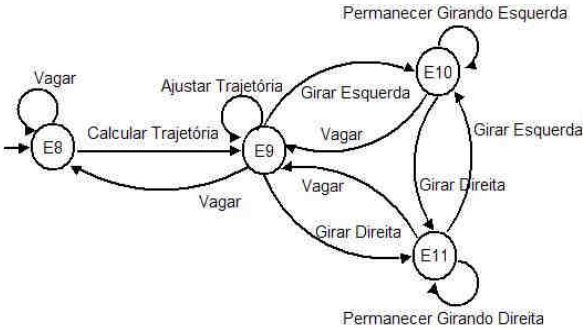

Figura 5. Autômato da camada "vagar com informação".

O funcionamento da camada "vagar com informação" é iniciado no instante em que o sistema de visão embarcado indicar a posição da bola. Dessa forma, o algoritmo desenvolvido executa uma ação que é responsável por calcular a trajetória de direção para o robô. O cálculo da trajetória de direção toma como referência o centro da imagem que representa a bola. Por outro lado, a partir do momento em que o sistema de visão não indicar mais a presença da posição em que a bola se encontra, o sistema desenvolvido retorna sua atividade de processamento para a camada "vagar sem informação", que passa a ter o controle de planejamento da trajetória para o robô.

Implementação da camada aproximar-se da bola

A camada "aproximar-se da bola" é executada a partir do momento em que o robô está aproximando-se da bola. Esta camada é a responsável pelo momento de aproximação do robô com relação à posição em que a bola se encontra. A camada "aproximar-se da bola" se faz necessária em função da importância que envolve o instante de transição do estado em que o robô não está de posse da bola e o instante em que o robô está de posse da bola; isto significa dizer que é necessário que esta transição de estado seja a mais suave possível, pois a chegada do robô à bola em uma velocidade constante pode resultar em um toque na bola de forma que ela seja conduzida para uma posição distante do robô.

O processo de redução de velocidade somente é aplicado para o robô que apresenta o comportamento de atacante. Sendo assim, este comportamento emerge somente a partir do instante em que o sistema de visão informa ao robô que ele está ocupando uma posição no ambiente a partir da linha que divide o campo; o robô passa a exercer o

comportamento de atacante, e se faz necessário aplicar o comportamento de redução de velocidade para que o robô possa permanecer com a posse da bola. Por outro lado, o comportamento de redução de velocidade não é aplicado quando o robô encontra-se em uma posição no ambiente antes da linha que divide o campo, pois esta posição no ambiente é considerada como área de defesa; dessa forma, é necessário que o robô apenas retire a bola da área.

O autômato da camada "aproximar-se da bola" possui três estados que são: Estado Longe da Bola, Estado Intermediário à Bola e Estado Próximo da Bola, conforme pode ser observado na Figura 6, que apresenta o autômato da camada "aproximar-se da bola".

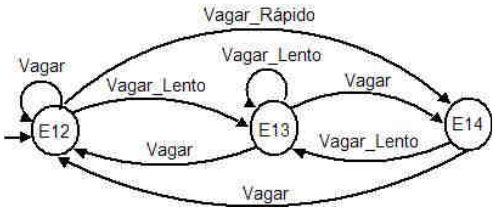

Figura 6. Autômato da camada "aproximar-se da bola".

O sistema implementado para a camada "aproximar-se da bola" foi projetado com o objetivo de atender à deficiência que o robô tem no momento em que passa do estado em que não está de posse da bola para o estado em que está de posse da bola. A abordagem do robô em relação à bola deve ser a mais suave possível, pois uma abordagem em alta velocidade pode resultar em colisão entre o robô e a bola, fazendo com que esta seja lançada para uma posição mais distante daquele. Portanto, o comportamento do algoritmo consiste em tornar o movimento do robô um pouco mais lento, até o momento que a abordagem aconteça. A partir do momento em que a abordagem aconteceu, a velocidade do robô retorna às condições normais de navegação. Quando o robô está de posse da bola, esta informação é enviada para a camada "conduzir bola", que passa a ter o controle do robô.

Implementação da camada conduzir bola

A camada "conduzir bola" somente é executada a partir do momento em que o robô está de posse da bola. No entanto, esta camada é responsável pelo controle da bola, ou seja, estando o robô de posse da bola e necessitando conduzi-la, esta camada é acionada. É esta camada a responsável pelo controle mais fino do robô, isto é, a camada "conduzir bola" informa quando a bola está em uma posição mais à direita ou mais à esquerda do mecanismo de chute do robô. O autômato da camada "conduzir bola" possui três estados que são: Estado Seguir em Frente, Estado Seguir à Esquerda e Estado Seguir à Direita, conforme pode ser observado na Figura 7, que apresenta o autômato da camada "conduzir bola".

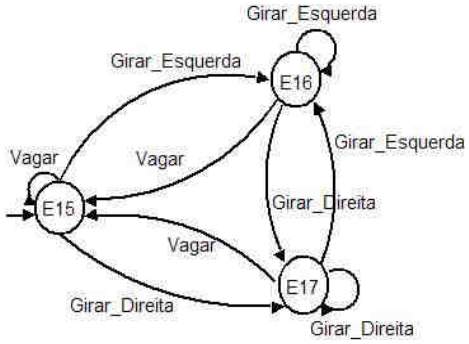

Figura 7. Autômato da camada "conduzir bola".

O sistema implementado para a camada "conduzir bola" é responsável pelo controle que o robô exerce na condução da bola. Esta camada controla o robô nos momentos em que a bola está sob o seu domínio. O controle exercido por este algoritmo é aplicado nos momentos em que a bola está escapando do domínio do robô, ou seja, quando a bola está em uma das extremidades do mecanismo de chute, que é externo ao robô. Por isso, é preciso controlar a bola em direção ao seu objetivo, que é a meta da equipe adversária.

O algoritmo que implementa o comportamento "conduzir bola" consiste apenas em verificar se a posição da bola se encontra no centro do mecanismo de chute; caso não esteja, ajusta a trajetória do robô para que a bola permaneça no centro do mecanismo de chute.

Implementação da camada chutar

A camada "chutar" somente é executada a partir do momento em que o robô está de posse da bola. No entanto, esta camada é responsável pelo controle do mecanismo de chute, isto significa dizer que, estando o robô de posse da bola e em condições de chutar, esta camada é acionada. O autômato da camada "chutar" possui três estados: Estado Abaixado, Estado Passe e Estado Levantado, conforme pode ser observado na Figura 8, que apresenta o autômato da camada "chutar".

O sistema implementado para a camada "chutar" é responsável pelo movimento do mecanismo de chute. O mecanismo de chute somente é acionado em duas condições: quando o robô está em condições de chutar a bola na meta da equipe adversária ou quando pretende passar a bola para um robô da mesma equipe. Para o caso em que o robô chuta a bola em direção à

meta da equipe adversária, o robô deve estar em condições de chute. Estas condições são: visão da meta da equipe adversária e posição do robô considerada como sendo próxima da meta da equipe adversária. Já para o caso em que o robô passa a bola para outro robô da mesma equipe, esta condição depende apenas da dinâmica do jogo.

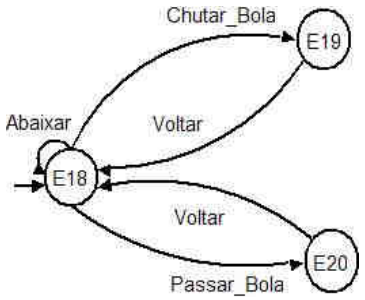

Figura 8. Autômato da camada "chutar".

Resultados e discussão

Com relação aos resultados obtidos com o desenvolvimento do sistema de visão do *EyeBot*, constatamos que este sistema é consistente em relação à localização da bola e das metas. No entanto, o tempo necessário para processamento de um quadro capturado pela câmera do *EyeBot* e a identificação da cor desejada pertence ao intervalo [0,22 s, 0,25 s]. Este tempo é atribuído aos recursos de hardware oferecidos pela câmera do *EyeBot*, pois somente o processo que envolve a captura do quadro pela câmera do *EyeBot* necessita de um tempo aproximado de 0,2 s, ou seja, por sua característica, a câmera utilizada captura até cinco quadros por segundo. Dessa forma, torna-se difícil melhorar o sistema de visão desenvolvido, visto que a margem de melhoria possível é de até 0,03 s. A Figura 9 apresenta no display de LCD do *EyeBot* a imagem da bola capturada após o processamento.

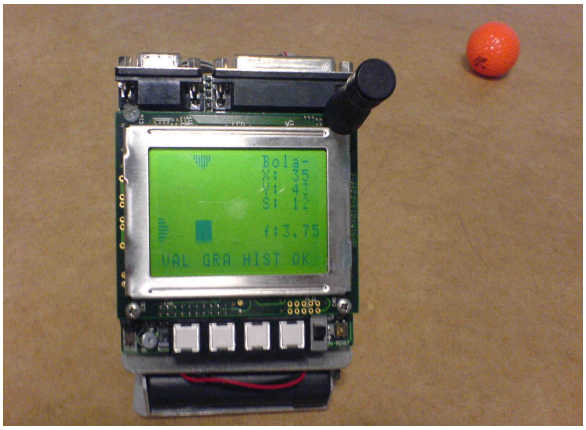

Figura 9. Imagem da bola no display.

O sistema desenvolvido para os sensores, por sua simplicidade, que consiste apenas em indicar a presença ou a ausência de um obstáculo e sua distância, comportou-se de forma eficiente. No entanto, dada a característica do conjunto dos sensores *EyeBot* que se está utilizando, percebeu-se que em situações nas quais o robô choca-se contra a parede, e sendo este choque fruto de uma colisão imprevista causada por outro robô ou qualquer outro problema que possa surgir, o sensor do *EyeBot* não consegue indicar que o robô está em colisão, ou seja, para uma distância menor que 5 cm, o sensor do *EyeBot* indica ausência de obstáculo.

O resultado obtido com o desenvolvimento do sistema dos atuadores constatou que, para uma velocidade linear maior do que 0,7, os robôs *EyeBot* não se comportam de forma adequada. Constatou-se também que uma velocidade linear que pertence ao intervalo [0,1; -0,1] para movimentos angulares, na qual o robô necessita girar à direita ou girar à esquerda, nem sempre é adequada. Em situações em que o robô está parado, a velocidade linear que pertence ao intervalo [0,1; -0,1] nem sempre consegue superar o atrito exercido sobre o ponto de apoio que o robô possui. A Tabela 1 apresenta o melhor conjunto de valores para solucionar o problema de velocidade linear e angular para o *EyeBot*.

Os resultados obtidos com relação à camada "vagar sem informação" permitiram constatar que realmente o robô vaga pelo ambiente conforme foi proposto. Com a implementação dos estados projetados, constatou-se que os estados e suas ações são considerados suficientes para que o robô vague pelo ambiente sem colidir com os obstáculos; referiu-se como sendo obstáculos os robôs e as paredes. Constatou-se também que, em situações nas quais o valor pré-definido no algoritmo do sistema de sensores é um valor muito alto, o robô não chega a aproximar-se das paredes; quando este valor é muito baixo, o robô não consegue detectar o obstáculo a tempo de evitar uma possível colisão. Dessa forma, foi necessário um ajuste desses valores para se obter o melhor desempenho por parte do robô.

Tabela 1. O melhor conjunto de valores para a velocidade linear e angular do *EyeBot*.

Ação	Velocidade linear	Velocidade angular
Navegação para frente	0,4	0,0
Navegação para frente rápido	0,6	0,0
Navegação para frente lento	0,2	0,0
Navegação de ré	-0,4	0,0
Girar à direita	-0,4	0,0
Girar à esquerda	0,4	0,0

Os resultados obtidos com relação à camada "vagar com informação" permitiram constatar que o algoritmo proposto para calcular a nova trajetória, que é em direção à posição da bola, realmente vai ao encontro do que havia sido proposto, ou seja, ao encontro da bola. No entanto, constatou-se que nem sempre o robô atinge este alvo. Esta deficiência é atribuída à máquina de estados do *EyeBot*, pois ela executa uma ação por um tempo aproximado de 1 s. Este tempo é atribuído em função de que cada servo motor é atualizado periodicamente a cada 20 ms, e o *EyeBot* utiliza: dois motores (motor A, motor B), um servo motor para a câmera e um servo motor para o mecanismo de chute. Também necessita atualizar a máquina de estados, então cada ação será executada por aproximadamente 1 s. Neste tempo de execução da máquina de estados, ficou comprovado que, dependendo do movimento exercido pela bola, ou seja, seu deslocamento no ambiente, o resultado obtido do deslocamento do robô em direção a uma posição em que a bola não esteja mais pode ocorrer. Esse tipo de problema não possui solução para esta aplicação, visto que para solucionar o problema é necessário desenvolver um novo sistema operacional para o conjunto de robôs *EyeBot*, com características diferenciadas para o sistema de deslocamento do robô. A Tabela 2 apresenta os resultados que foram obtidos por um conjunto de ensaios realizados com relação à camada "vagar com informação".

Tabela 2. Resultados obtidos da camada "vagar com informação".

Ensaios	Coordenada X	Coordenada Y	Tamanho da bola	Velocidade linear	Velocidade angular
1	9	37	7	0,2	0,23
2	11	56	4	0,2	0,19
3	15	47	8	0,2	0,31
4	44	49	17	0,2	0,73
5	67	56	7	0,2	0,85
6	24	46	10	0,2	0,46
7	21	32	10	0,4	0,0
8	20	26	8	0,4	0,0
9	3	1	2	0,2	- 0,39
10	11	3	5	0,2	-0,65
11	30	9	15	0,2	-0,68
12	55	8	12	0,2	-0,75
13	31	16	15	0,2	-0,54

Esta camada apresenta um conjunto de 13 ensaios. Para cada ensaio, foram coletadas as coordenadas (x,y) em que a bola se encontra e também o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. A partir da posição em que a bola se encontra nas coordenadas (x,y), o sistema desenvolvido para a camada "vagar com informação" define um vetor de velocidade que corresponde à velocidade linear e à velocidade angular, que será executado pelos atuadores. Estes resultados representam as possíveis posições em que a bola pode encontrar-se no ambiente. Estes dados foram extraídos do display do *EyeBot* após o processamento desta camada.

Com relação aos resultados obtidos com o desenvolvimento da camada "aproximar-se da bola", percebeu-se que a quantidade de variáveis possíveis que podem influenciar o momento da chegada do robô à bola é muito grande. Observou-se que o robô encontra dificuldade com relação à forma como essa abordagem acontece. Em situações em que a bola encontra-se parada, ou segue a mesma trajetória de direção do robô, a abordagem acontece de forma perfeita. No entanto, em situações em que a bola encontra-se em movimento e na direção contrária à trajetória de deslocamento do robô e, ao mesmo tempo, dependendo da velocidade da bola, o momento da abordagem do robô em relação à bola nem sempre acontece de forma suave. Esta deficiência acontece em função de a máquina de estados estar executando uma determinada ação e o tempo envolvendo a colisão ser muito pequeno. Dessa forma, a máquina de estados ainda não foi atualizada para o novo estado e, assim, pode ocorrer uma colisão envolvendo o momento em que o robô chega à bola. No entanto, mesmo que ocorra uma colisão, e dependendo da forma como ela aconteça, o robô permanece com a posse da bola. A Tabela 3 apresenta os resultados obtidos da camada "aproximar-se da bola".

Tabela 3. Resultados obtidos da camada "aproximar-se da bola".

Ensaios	Coordenada X	Coordenada Y	Tamanho da bola	Velocidade linear	Velocidade angular
14	66	26	66	0,4	0,0
15	65	41	21	0,1	0,0
16	64	29	18	0,1	0,0
17	49	32	15	0,1	0,0
18	40	32	12	0,1	0,0
19	39	32	11	0,4	0,0

Esta camada apresenta um conjunto de seis ensaios. Para cada ensaio foram coletadas as coordenadas (x,y) em que a bola se encontra e também o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. Nesta camada, o sistema utiliza apenas a coordenada (x) para ativar o comportamento de redução de velocidade. Desse modo, o sistema desenvolvido verifica a coordenada (x) e, a partir de um valor inferir a 66, que corresponde à distância em que a bola se encontra do robô, aplica o comportamento de redução de velocidade; a partir do instante em que o valor da distância é inferior a 40, retorna à velocidade normal de navegação. Esses resultados representam as possíveis posições em que

a bola pode encontrar-se no ambiente. Esses dados também foram extraídos do display do *EyeBot* após o processamento da camada "aproximar-se da bola".

Com relação aos resultados obtidos com o desenvolvimento da camada "conduzir bola", constatou-se que o algoritmo proposto para ajustar a nova trajetória do robô, que é responsável pela condução da bola, realmente vai ao encontro do que havia sido proposto, ou seja, a camada "conduzir bola" realiza o processo de correção da trajetória do robô nos momentos em que a bola está saindo do controle do mecanismo de chute do próprio robô. Também se verificou que o comportamento desta camada é semelhante ao comportamento da camada "vagar com informação". No entanto, a diferença que existe é que a bola está próxima do robô, isto é, a função pela qual esta camada foi projetada é ajustar a curva do robô no momento em que o robô está de posse da bola, mas a bola está deslizando por uma das laterais do mecanismo de chute. Dessa forma, faz-se necessário redirecionar o robô para que ele possa permanecer com a posse da bola. A Tabela 4 apresenta os resultados obtidos da camada "conduzir bola".

Tabela 4. Resultados obtidos da camada "vagar com informação".

Ensaios	Coordenada X	Coordenada Y	Tamanho da bola	Velocidade linear	Velocidade angular
20	72	53	11	0,2	0,94
21	73	48	7	0,2	0,90
22	71	20	19	0,4	0,0
23	72	16	12	0,4	0,0
24	71	6	13	0,2	-0,74
25	70	3	11	0,2	-0,76

A camada "conduzir bola" apresenta um conjunto de seis ensaios. Para cada ensaio, foram coletadas as coordenadas (x,y) em que a bola se encontra e também o tamanho da imagem gerada pela integração dos histogramas nos eixos horizontal e vertical. Nesta camada, o sistema utiliza apenas a coordenada (y) para ativar o comportamento responsável por planejar a nova trajetória do robô. Desse modo, o sistema desenvolvido verifica a coordenada (y) e, ao perceber que a bola não se encontra em um intervalo aceito como sendo o centro do mecanismo de chute, calcula uma nova trajetória, que é responsável por redirecionar o robô para que a bola possa retornar ao centro do mecanismo de chute. A partir de um valor inferir a 15, que corresponde a um ponto à direita do mecanismo de chute, o algoritmo calcula a nova trajetória para o robô. A nova trajetória deve direcionar o robô à direita, dessa forma, a velocidade angular será negativa. Por outro lado, a partir de um valor superior a 45, que corresponde a um ponto à esquerda do mecanismo de chute, o algoritmo calcula a nova trajetória para o robô. A nova trajetória deve direcionar o robô à esquerda, assim, a velocidade angular será positiva. Esses resultados representam as possíveis posições em que a bola pode encontrar-se no ambiente. Esses dados também foram extraídos do display do *EyeBot* após o processamento da camada "conduzir bola".

Finalmente, com os resultados obtidos com o desenvolvimento da camada "chutar bola", percebeu-se que o algoritmo desenvolvido para controlar o mecanismo de chute, que é responsável pelo toque na bola em direção à meta da equipe adversária, ou mesmo para passar a bola para um robô da mesma equipe, apresenta uma característica de movimento lento. Ou seja, apesar do servo motor que controla o mecanismo de chute estar em condições de movimentar-se com total velocidade – e em nenhum momento o algoritmo desenvolvido apresenta controle de velocidade de movimento, apenas as condições de abaixar, passar e levantar o mecanismo de chute – percebe-se que a bola não é lançada com muita força. Dessa forma, faz-se necessário aproximar o robô o máximo possível da meta da equipe adversária para que possa ser ativado o comportamento chutar a gol.

Também se constatou que, apesar do algoritmo desenvolvido apresentar a condição de passar a bola para o jogador da mesma equipe, este comportamento nem sempre acontece, ou acontece em momentos considerados inadequados. Para poder explorar melhor esse comportamento, faz-se necessário implementar condições relacionadas à cooperação multiagentes. Esta cooperação indica quais são os momentos em que o robô da mesma equipe se encontra em melhor condição para receber a bola e continuar com a condução da bola em direção à meta da equipe adversária.

Conclusão

Neste artigo, descreveu-se e analisou-se o desenvolvimento de uma arquitetura embarcada para sistemas multiagentes e sua aplicação no futebol de robôs. Esta arquitetura é adaptada com um sistema de visão local e dotada de sensores para captura das informações. O processamento das imagens é realizado localmente, no próprio sistema de visão do robô.

A arquitetura embarcada para processamento das informações permitiu tratar as informações em tempo real e possibilitou ganho com relação ao

desempenho do sistema. Assim, o sistema de visão local foi beneficiado, pois esta arquitetura permitiu aos níveis de comportamento e execução se beneficiarem com relação ao domínio do ambiente.

A arquitetura proposta neste trabalho foi testada na V Competição Brasileira de Robótica. Diante dos resultados obtidos, mostrou-se uma solução adequada para controle embarcado para sistemas que operam em tempo real e sistemas multiagentes capazes de jogar futebol.

Referências

BROOKS, R. A robust layered control system for a mobile robot. **IEEE Journal of Robotics and Automation**, v. 2, n. 1, p. 14-23, 1986.

GUPTA, G. S.; MESSOM, C. H.; DEMIDENKO, S. Real-time identification and predictive control of fast mobile robots using global vision sensing. **IEEE Transaction on Instrumentation and Measurement**, v. 54, n. 1, p. 1-25, 2005.

HU, H.; BRADY, M. A parallel processing architecture for sensor-based control of intelligent mobile robots. **Robotics and Autonomous Systems**, v. 17, p. 235-257, 1996.

MURPHY, R. R. Dempster-shafer theory for sensor fusion in autonomous mobile robots. **IEEE Transactions on Robotics and Automation**, v. 14, n. 2, p. 197-206, 1998.

MURPHY, R. R. **Introduction to AI robotics**. London: The MIT Press, 2000.

NORBERT, J. Decision making and image processing in robot soccer: the challenge of the FIRA MiroSOT league. **Ecole des Mines**, v. 12, n. 1, [s/p], 2006.

SHIM, H. S.; VADAKKEPAT, P. A hybrid control structure for vision based Soccer Robot System. **International Journal of Intelligent Automation and Soft Computing**, v. 6, n. 1, p. 89-101, 2000.

VADAKKEPAT, P.; PENG, X.; QUEK, B. K.; LEE, T. H. Evolution of fuzzy behaviors for multi-robotic system. **Robotics and Autonomous Systems**, v. 55, n. 2, p. 146-161, 2007.

WANG, Z. D.; TAKAHASHI, T.; NITSUMA, T.; NINJOUJI, T.; NAKANO, E. Logue: an architecture for task and behavior object transmission among multiple autonomous robots. **Robotics and Autonomous Systems**, v. 44, n. 1, p. 261-271, 2003.

Received on November 08, 2007.
Accepted on July 29, 2008.