RECONHECIMENTO EM TEMPO REAL DE AGENTES AUTÔNOMOS EM FUTEBOL DE ROBÔS

William Robson Schwartz*
william@inf.ufpr.br

Oliver Matias van Kaick*
oliver@inf.ufpr.br

Murilo Vicente Gonçalves da Silva*
murilo@inf.ufpr.br

Hélio Pedrini*
helio@inf.ufpr.br

*Departamento de Informática
Universidade Federal do Paraná
81531-990 Curitiba-PR, Brasil

ABSTRACT

Robot soccer has been adopted as a standard problem to promote the development of new techniques in the field of Artificial Intelligence. The domain of robot soccer is highly dynamic and complex, consisting of multi-agents that are capable of performing individual and cooperative actions to solve a task. This paper describes the computer vision module implemented in UFPR robot-soccer team. The software architecture used for the identification and tracking of the objects is presented and discussed. A flexible and efficient algorithm is proposed for real time identification and tracking of objects in the scene. Several experiments have been performed in order to demonstrate the performance of the algorithms.

KEYWORDS: Object tracking, color classification, robot soccer.

RESUMO

Futebol de Robôs tem sido adotado internacionalmente como uma atividade científica voltada à educação e pesquisa, visando estimular o desenvolvimento de novas técnicas na área de Inteligência Artificial. O domínio de Futebol de Robôs é bastante dinâmico e complexo, consistindo em múltiplos agentes capazes de realizar ações individuais e colaborativas para a execução de uma tarefa. Este trabalho descreve o sistema de visão computacional implementado na equipe de Futebol de Robôs da UFPR. A arquitetura básica do projeto e o sistema de visão computacional são apresentados e discutidos. Um algoritmo simples e eficiente para a identificação e rastreamento em tempo real de objetos durante o jogo é proposto, incluindo um sistema eficiente de classificação de cores. Diversos experimentos são realizados para demonstrar o desempenho dos algoritmos.

PALAVRAS-CHAVE: Rastreamento de objetos, classificação de cores, futebol de robôs.

1 INTRODUÇÃO

Futebol de Robôs foi proposto por diversos pesquisadores (Kitano et al., 1997; Hamlin and Sanderson, 1997) como uma atividade científica que integra uma grande variedade de tópicos relacionados à educação e pesquisa na área de Inteligência Artificial, tais como visão computacional (Bandlow et al., 1999), princípios de agentes autônomos, raciocínio em tempo real e robótica (Simon et al., 2001; Nakamura and Ogasawara, 1999).

Com a internacionalização da idéia do Futebol de Robôs, surgiu a necessidade de definição de regras de modo a garantir compatibilidade entre as equipes. Como resultado, surgiram duas associações: a *Federation of International Robot-soccer Association* (FIRA, 2003), e a *Robot World Cup Initiative* (ROBOCUP, 2003), que organizam torneios e promovem o desenvolvimento de sistemas inteligentes e robôs autônomos, contribuindo para o estado da arte em áreas especializadas. Para compatibilidade das equipes, cada associação define suas categorias ou ligas, as quais diferenciam entre si com relação ao número, tamanho e forma dos robôs utilizados pelas equipes.

No projeto de Futebol de Robôs desenvolvido por nossa equipe, decidiu-se adotar as regras da *FIRA* na categoria *Small League Mirosot* onde cada equipe é composta de três robôs com dimensões 7.5cm $\times$ 7.5cm $\times$ 7.5cm. Marcas coloridas com dimensões 3.5cm $\times$ 3.5cm são colocadas sobre os robôs para identificá-los individualmente. O campo tem dimensões de 150cm $\times$ 130cm com superfície de cor preta. Uma câmera instalada no centro do campo com altura de 2m transmite imagens dos objetos para o computador. O computador intrepreta essas imagens e envia comandos para movimentação dos robôs baseados em decisões da estratégia de jogo. Um esquema básico do sistema é apresentado na figura 1, adaptado de (FIRA, 2003).

O sistema utilizado em nosso projeto está dividido em quatro

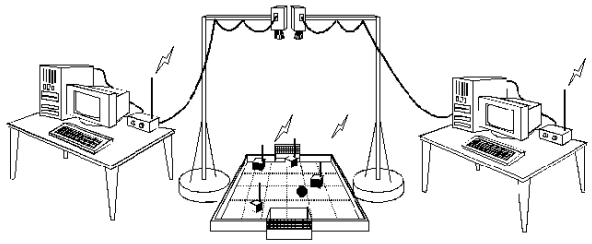


Figura 1: Esquema básico de um sistema de Futebol de Robôs.

módulos principais: (figura 2):

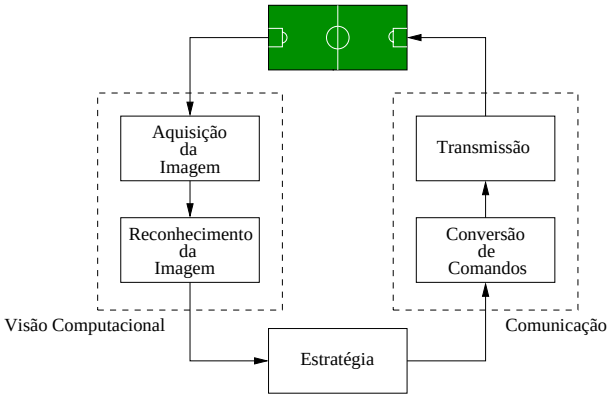


Figura 2: Diagrama da arquitetura proposta.

- *Visão Computacional*: a partir da imagem capturada pela câmera digital, este módulo é responsável pelo reconhecimento dos robôs e da bola no campo de jogo. A determinação da posição e orientação dos robôs é realizada através de marcas coloridas colocadas sobre cada robô;
- *Estratégia*: através das informações fornecidas pelo módulo de visão computacional, este módulo é responsável pelas decisões de jogadas a serem executadas pelos robôs, incluindo táticas de jogo.
- *Tática*: responsável pela tradução dos comandos enviados pelo módulo de estratégia, definindo trajetórias dos jogadores representadas por comandos mais simples que serão interpretados pelo módulo de comunicação.
- *Comunicação*: converte as instruções resultantes da tática em comandos de movimentação para os robôs. Tendo em vista a proposta de se ter robôs autônomos, não é permitido qualquer tipo de ligação aos robôs com fios.

Este artigo descreve o módulo de visão computacional implementado em nossa equipe de Futebol de Robôs, desenvolvido de acordo com as regras da Robocup.

As seções 2.1 e 2.2 apresentam os algoritmos propostos para permitir a identificação e localização dos objetos de interesse (robôs e bola) durante o jogo. Uma otimização para melhorar o desempenho do módulo de visão computacional é descrita na seção 2.4. A seção 3 analisa o desempenho do sistema desenvolvido e apresenta alguns resultados experimentais. Conclusões e sugestões para trabalhos futuros são apresentadas na seção 4.

2 SISTEMA DE VISÃO COMPUTACIONAL

O principal objetivo do módulo de visão computacional é criar um modelo a partir das informações sobre cada objeto encontrado na imagem, mais especificamente identificando a posição da bola e a posição e orientação dos agentes autônomos.

As principais estapas do módulo de visão computacional implementados em nossa equipe de Futebol de Robôs consistem em: construção de uma base de conhecimento, durante a fase de calibração do sistema, que será utilizada para processar cada imagem capturada e criar a representação de cada situação; aquisição da imagem; reconhecimento das cores dos objetos (robôs e bola) e determinação da posição e orientação dos robôs de nossa equipe no campo de jogo. Essas etapas são descritas nas seções a seguir.

2.1 Construção da Base de Conhecimento

Antes do início do jogo, uma base de conhecimento é construída para que o algoritmo possa reconhecer quantos objetos existirão na cena e quais são as cores e características desses objetos. Essa base de conhecimento é construída por meio de uma interface gráfica controlada por um operador humano durante o período de calibração do sistema.

A base de conhecimento contém as seguintes informações: número de robôs de nossa equipe (n); diâmetro da bola (*diam_b*); diâmetro da marca dos robôs (*diam_m*); diâmetro da camiseta dos robôs (*diam_c*) e distância em pixels entre a marca e a camiseta dos robôs (*dist_c_m*). A marca e a camiseta, descritas na seção 2.3, são identificações posicionadas sobre os robôs para permitir o reconhecimento.

2.2 Classificação das cores

O sistema utilizado para atribuir um valor a um pixel é o espaço cromático RGB, o qual foi escolhido por ser o modo fornecido pela nossa câmera, o qual possui mais informações (24 bits) se comparado aos outros modos. O algoritmo também pode ser utilizado com outros modelos de cores, por exemplo, HSI ou YUV.

No sistema RGB, a cor de cada pixel é definida por três componentes (R, G, B). Cada componente define a quantidade de uma cor primária presente na cor do pixel, sendo *R* a quantidade de vermelho, *B* a quantidade de azul e *G* a quantidade de verde. No RGB fornecido por nossa câmera, cada componente é definida por 8 bits.

Quando as imagens são convertidas para a forma digital, os pixels que definem os objetos acabam sendo de vários tons de uma mesma cor. De forma a verificar se um pixel pertence a uma classe de cor particular, o reconhecimento deve utilizar um espaço de tolerância de uma determinada cor ao invés de um valor absoluto para comparação.

Para determinar se o pixel pertence ao espaço de tolerância, cada componente do pixel deve pertencer a um determinado intervalo. Este intervalo é calculado usando-se um valor fornecido pelo usuário (por meio de uma interface gráfica apresentada durante a fase de calibração) e somando-se e subtraindo-se um valor de tolerância T deste valor. Esse valor T é determinado de acordo

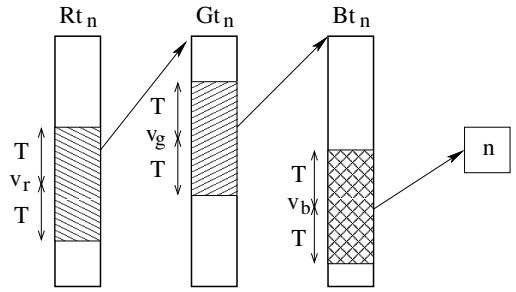


Figura 3: Criação de três tabelas *hash* utilizadas para indexação das componentes R, G e B de uma determinada cor C_x para classificá-la como a cor C_n.

com as condições de iluminação e contraste das imagens.

Durante a etapa de calibração do módulo de visão computacional, o usuário informa o valor de N_c cores (para criar N_c classes) e é criada uma árvore de tabelas *hash* (Cormen et al., 1996) que será utilizada para classificação de cores.

Essa árvore é composta de várias tabelas *hash* de 256 posições (são necessárias 256 posições porque cada componente RGB da cor contém 8 bits de informação) que serão indexadas pelo valor das componentes RGB de uma cor, com o objetivo de classificá-la como uma das cores definidas pelo usuário na fase de calibração.

Para cada cor C_n calibrada pelo usuário, n variando entre 1 e N_c, que tem valor v_r na componente R, v_g na componente G e v_b na componente B, são criadas as tabelas *hash* Rt_n, Gt_n e Bt_n. As posições entre $v_b - T$ e $v_b + T$ da tabela Bt_n apontam para o número n, as posições entre $v_g - T$ e $v_g + T$ da tabela Gt_n apontam para a tabela Bt_n e as posições entre $v_r - T$ e $v_r + T$ da tabela Rt_n apontam para a tabela Gt_n, como mostra a figura 3. Essa estrutura de apontadores permite classificar uma cor C_x qualquer como C_n apenas com a indexação dos valores R, G e B de C_x nas tabelas Rt_n, Gt_n e Bt_n, respectivamente. Se o resultado dessa indexação apontar para n, C_x será classificada como C_n.

No modelo descrito acima, para classificar cada cor deve-se indexar o conjunto de tabelas Rt_n, Gt_n e Bt_n para cada cor calibrada. Visando melhor desempenho, as tabelas Rt de todas as cores são agrupadas em uma matriz, M_R, com N_c colunas por 256 linhas. A primeira coluna de M_R é preenchida com os valores de Rt_1. O valor de cada linha l das tabelas R_t das demais cores é colocado na primeira coluna sem nenhum apontador da linha l da matriz M_R, como mostra a figura 4(a).

Com a utilização da matriz será necessário uma nova coluna de M_R para uma tabela R_t apenas nos valores da componente R da cor n que têm interseção com os valores da componente R de outra cor.

A indexação dessas tabelas equivale à indexação de endereços de memória. Para não permitir a utilização de endereços de memória inválidos, são criadas as tabelas Nt_g e Nt_b. Todas as posições de Nt_g apontam para Nt_b e todas as posições de Nt_b apontam para $NULL$. Todas as posições sem apontadores de M_R de todas tabelas G_t e de todas as tabelas B_t, passam a apontar para Nt_g, Nt_b e $NULL$, respectivamente, conforme figura 4(b).

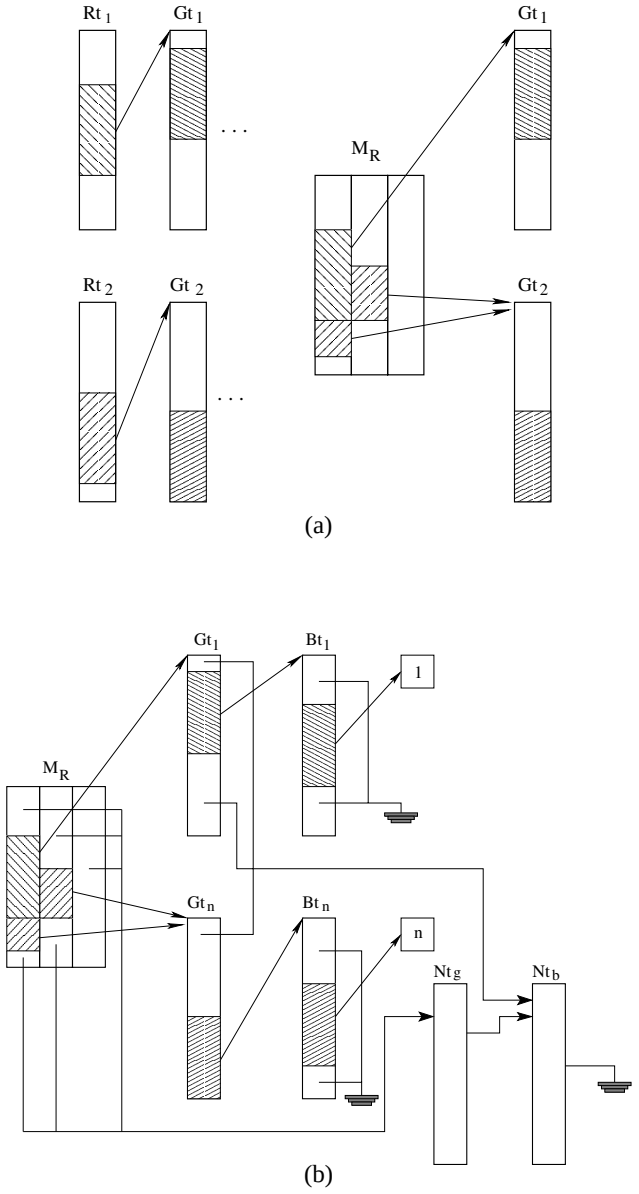


Figura 4: (a) Agrupamento em uma matriz de todas as tabelas *hash* Rt; (b) acréscimo das tabelas *hash* Nt_g e Nt_b para não permitir a utilização de endereços inválidos de memória.

	Classificação de uma cor	
	melhor caso	pior caso
Método proposto	1	w_l
Método (Bruce et al., 2000)	1	N_c
Método (van Kaick et al., 2001)	6	$6N_c$

Tabela 1: Número de comparações para classificação de cores.

	Descarte de uma cor	
	melhor caso	pior caso
Método proposto	1	w_l
Método (Bruce et al., 2000)	N_c	N_c
Método (van Kaick et al., 2001)	$6N_c$	$6N_c$

Tabela 2: Número de comparações para descarte de cores.

Para o sistema de visão computacional, apenas os robôs e a bola são objetos relevantes sob o ponto de vista de identificação. Em uma imagem com resolução de 640 $\times$ 480 pixels, apenas 1323 pixels são relevantes, ou seja, 0,43% da imagem. Os outros pixels devem ser descartados rapidamente. Como no método proposto os objetos irrelevantes não são inseridos em M_R, o descarte de uma cor tende a ser executado em tempo constante, no melhor caso.

Define-se w_l como a largura utilizada em cada linha l de M_R, $w_l \leq N_c$. As tabelas 1 e 2 mostram o número de comparações para classificação e descarte de uma cor, respectivamente, entre o nosso método e os métodos descritos em (Bruce et al., 2000) e (van Kaick et al., 2001).

2.3 Reconhecimento e Rastreamento dos Objetos

Os objetos a serem identificados na imagem são os robôs e a bola. De modo a permitir a identificação, a localização e a orientação dos robôs, algumas marcas são colocadas sobre cada robô. A figura 5 mostra como estão dispostas as marcas sobre os robôs de nossa equipe.

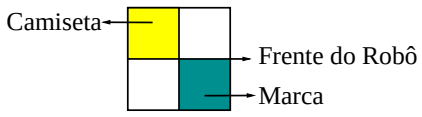


Figura 5: Disposição das marcas sobre os robôs.

Uma cor é utilizada para a identificação de nossa equipe, ou seja, todos os robôs possuem uma marca (chamada de *camiseta*) desta cor. Adicionalmente, a uma distância $dist\_c\_m$ do centro da camiseta, cada robô recebe uma identificação (chamada de *marca*) de uma cor única. É essa marca que identifica cada robô individualmente.

O método para reconhecer os robôs e a bola na imagem consiste em duas etapas. A primeira etapa utiliza um algoritmo de agrupamento (*clustering*), que cria conjuntos de pixels de elementos (marcas, camisetas e bola) da imagem. A segunda etapa consiste em associar conjuntos de pixels de camisetas a conjuntos de pixels de marcas para identificar individualmente cada robô.

2.3.1 Algoritmo de Agrupamento

O algoritmo de agrupamento lê a imagem e cria vários conjuntos, onde cada elemento destes conjuntos é um pixel da imagem. As métricas utilizadas para criar os conjuntos são as cores dos pixels e a distância entre eles.

As distâncias $diam\_b$ para os pixels da cor da bola, $diam\_c$ para os pixels da cor da camisetas e $diam\_m$ para os pixels da cor das marcas são obtidas na etapa de calibração do sistema. Se mais conjuntos do que o esperado são criados, apenas os maiores são considerados (ex.: uma bola, três camisetas e três marcas). O número de conjuntos de cada elemento é obtido na etapa de calibração.

2.3.2 Associação de Marcas a Camisetas

A posição de cada robô é dada pelo ponto médio do segmento de reta que liga o centróide da marca com o centróide da camiseta de um robô.

O algoritmo que cria os pares marca-camiseta deve evitar que essas associações sejam feitas de maneira incorreta. Para tal, o algoritmo inicialmente resolve os casos triviais, isto é, os casos em que há apenas uma marca próxima a uma camiseta. Essa proximidade é definida pelo limiar *dist_c_m* obtida na etapa de calibração. O algoritmo iterativamente adia a decisão de associação para as camisetas que possuem mais de uma marca próxima à ela, até que estes casos não ocorram mais. Esses casos não triviais são eliminados quando a marca extra próxima a uma camiseta é associada a uma outra camiseta em um caso trivial.

O cálculo do ângulo de um robô é dado por $\alpha + 45°$, onde α é o ângulo formado entre o segmento orientado de reta que liga o centróide da camiseta com o centróide da marca do robô e o eixo x. O valor $45°$ é conseqüência da disposição diagonal das marcas sobre o robô.

A seguir é apresentado o pseudocódigo do algoritmo utilizando algumas definições da subseção 2.1. Considere $dist(p, q)$ a distância Euclidiana do ponto p até o ponto q, $cent(elem)$ o centróide do elemento $elem$ (exemplo, camiseta), *Clustering()* uma função que cria os conjuntos de pixels, *n_marcas* um vetor que guarda em cada posição *n_marcas[i]* o número de marcas associadas à camiseta i, e *assoc* um vetor que guarda em cada posição *assoc[i]* uma marca associada à camiseta i.

```
Clustering();
Inicializa vetores n_marcas e assoc;
i ⇐ 0;
ok ⇐ FALSO;
enquanto (i ≤ n e não(ok))
        ok ⇐ TRUE;
        i ⇐ i + 1;
        para cada camiseta j
                para cada marca k
                        se (dist(cent(j), cent(k)) ≤ dist_c_m)
                                assoc[j] ⇐ k;
                                n[j] ⇐ n[j] + 1;
        para cada camiseta j
                se (n_marca[j] = 1)
                        associa camiseta j com a marca assoc[j];
                senão
                        ok ⇐ FALSO;
                        i ⇐ i − 1;
```

2.4 Otimização: Varredura Parcial da Imagem

Com intuito de aumentar a velocidade de processamento, é proposto um método que utiliza a base de conhecimento para evitar que todos os pixels da imagem sejam percorridos. Esse método consiste na alteração da varredura da imagem, de forma a percorrer parcialmente a área da imagem, o que é possível já que os objetos cobrem uma área que contém um determinado número de pixels. Para detectar o local onde se encontra um objeto é necessário identificar somente a coordenada de um pixel.

Para percorrer parcialmente a imagem, uma grade com espaçamento entre cada ponto é criada de tal forma que os objetos sempre estejam no mínimo sobre um ponto da grade. A distância entre dois pontos p_1 e p_2 adjacentes da grade deve ser menor que o lado l do menor objeto a ser detectado; para a bola considera-se a dimensão do lado do quadrado inscrito em sua circunferência.

Define-se a seguinte fórmula para calcular a distância entre cada ponto da grade, $dist(p_1, p_2) < l$, onde $dist(p_1, p_2)$ retorna a distância entre os pontos p_1 e p_2. Supondo uma imagem com altura de h pixels e largura de w pixels, utilizando-se o método atual o número de pixels percorridos é $w * h$. Utilizando-se o método de varredura parcial, seriam percorridos $(w * h)/(l * l)$ pixels.

3 RESULTADOS EXPERIMENTAIS

Alguns resultados obtidos através da utilização de nosso algoritmo são apresentados na figura 6. A imagem (a) mostra uma situação facilmente processada, mostrando três robôs (numerados de 0 a 2) e a bola (rotulada com a letra B) corretamente reconhecidos. A imagem (b) mostra, além dos robôs e da bola, a marca de um adversário (rotulada com a letra A) reconhecida. As imagens (c) e (d) mostram duas situações que podem gerar erros durante o processamento. Entretanto, em ambos casos, os robôs foram mapeados corretamente, devido à utilização do algoritmo de agrupamento (seção 2.3.1).

Os algoritmos foram implementados em um microcomputador PC Pentium III 450 MHZ com 128 Mbytes de memória RAM utilizando linguagem de programação C com sistema operacional Linux.

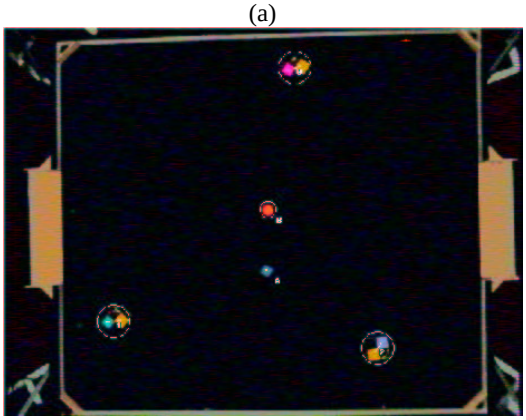
As imagens são capturadas por uma câmera de vídeo colorida com saída em VHS e digitalizadas por uma placa PCI da PixelView PV-Bt-878. A imagem digitalizada é adquirida através da interface *video4linux*. A imagem utiliza resolução de 640×480 pixels e 24 bits de quantização em RGB.

4 CONCLUSÕES

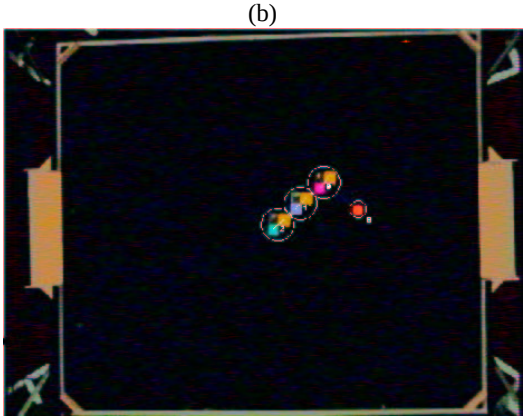
Nosso sistema de visão computacional mostrou-se robusto às condições de luminosidade, sendo capaz de identificar as cores dos objetos com alta precisão. Ele permite a execução em tempo real, tal que objetos podem ser localizados e rastreados a uma taxa de 30 quadros por segundo, com o ângulo variando no máximo 5° de quadro para quadro. Além disso, a técnica de classificação de cores proposta é genérica, podendo ser aplicada a outros domínios de aplicação envolvendo segmentação de cores.

Como trabalho futuro, será implementada a otimização proposta

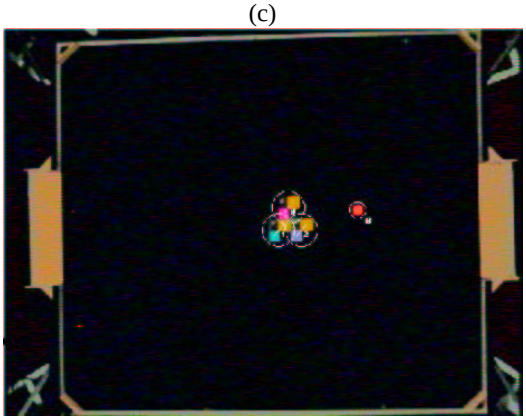
(a)



(b)



(c)



(d)

Figura 6: (a) e (b) mostram situações simples de reconhecimento; (c) e (d) mostram situações complexas de reconhecimento.

anteriormente. O uso de um método de varredura parcial da imagem deve melhorar o desempenho do atual sistema de visão computacional, utilizando menos processamento e, assim, fornecendo mais recursos a outros módulos do sistema.

AGRADECIMENTOS

Os autores gostariam de agradecer aos integrantes do Programa Especial de Treinamento (PET) do Departamento de Informática da UFPR.

REFERÊNCIAS

Bandlow, T., Klupsch, M., Hanek, R. and Schmitt, T. (1999). Fast image segmentation, object recognition and localization in a RoboCup scenario, *RoboCup Workshop, International Joint Conference on Artificial Intelligence, RoboCup Workshop, Estocolmo, Suécia*, pp. 174–185.

Bruce, J., Balch, T. and Veloso, M. (2000). Fast and cheap color image segmentation for interactive robots, *Workshop on Interactive Robotics and Entertainment, Carnegie Mellon University, Pittsburgh, PA, Estados Unidos*.

Cormen, T., Leiserson, C. and Rivest, R. (1996). Introduction to algorithms.

FIRA (2003). *Federation of International Robot-soccer Association*. `http://www.fira.net`.

Hamlin, G. J. and Sanderson, A. C. (1997). Tetrobot: A modular approach to parallel robotics, *IEEE Robotics and Automation Magazine* **4**(1): 42–50.

Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H. and Osawa, H. (1997). Robocup: A challenge problem for AI, *AI Magazine* **18**(1): 73–85.

Nakamura, T. and Ogasawara, T. (1999). On-line visual learning method for color image segmentation and object tracking, *Proceedings of IROS'99*, pp. 222–228.

ROBOCUP (2003). *Robot World Cup Initiative*. `http://www.robocup.org`.

Simon, M., Behnke, S. and Rojas, R. (2001). Robust real time color tracking, *Lecture Notes in Computer Science* **2019**: 239–248.

van Kaick, O. M., Schwartz, W. R., da Silva, M. V. G. and Pedrini, H. (2001). Identificação e rastreamento em tempo real de múltiplos agentes autônomos, *X Seminário de Computação, FURB, Blumenau, SC, Brasil*, pp. 59–70.