colloquium exactarum

IDENTIFICAÇÃO DE OBJETOS DO FUTEBOL DE ROBÔS UTILIZANDO ALGORITMO DE DESCRIÇÃO DE PONTOS CHAVE

IDENTIFYING OBJECTS USING ROBOT SOCCER ALGORITHM DESCRIPTION OF KEY POINTS

Ricardo da Silva Barros[1], Francisco Assis da Silva[1], Mário Augusto Pazoti[1], Danillo Roberto Pereira[1], Leandro Luiz de Almeida[1], Almir Olivette Artero[2]

[1]Faculdade de Informática – FIPP, Universidade do Oeste Paulista – UNOESTE
E-mail: ricardo.sbarros@hotmail.com, {chico, mario, danilopereria, llalmeida}@unoeste.br
[2]Faculdade de Ciências e Tecnologia – FCT, Universidade Estadual Paulista – Unesp
e-mail: almir@fct.unesp.br

RESUMO - Este trabalho propõe uma metodologia aplicada na parte visão computacional necessária para a construção de um futebol de robôs, que fornece a posição e orientação da bola e dos robôs, utilizando um algoritmo de descrição de pontos chave. Foi escolhido o algoritmo SIFT por apresentar melhores resultados que os outros algoritmos similares. Para alcançar um tempo de processamento adequado ao futebol de robôs, foi utilizado o processador da placa de vídeo (GPU) com o algoritmo, tornando a obtenção das informações de localização e orientação mais próximas do tempo real. A visão é apenas uma das partes que compõem a construção de um futebol de robôs. Classes de objetos foram definidas e modeladas na linguagem de programação C++ para as tarefas mínimas necessárias dessa parte da visão, a fim de facilitar e possibilitar o desenvolvimento das outras partes (construção mecânica, controle dos robôs e estratégia de jogo) para se ter um futebol de robô completo.
Palavras-chave: Descritores de Pontos Chave; Futebol de Robôs; Visão Computacional; SIFT; GPU.

ABSTRACT - This work proposes a method implemented in computer vision part required to build a robot soccer, which provides the position and orientation of the ball and robots, using an algorithm description of keypoint. SIFT algorithm was chosen because it provided better results than other similar algorithms. To achieve adequate time for processing to robot soccer, we used the video card processor (GPU) with the algorithm, making obtaining location information and orientation closer to the real-time. The view is just one of the parts that make up the construction of a robot soccer. Object classes were defined and shaped in the C ++ programming language to the minimum necessary tasks of this part of view, in order to facilitate and to allow the development of other parts (mechanical construction, robots control and game strategy) to be a full robot football.
Keywords: Keypoint Descriptors; Robot Soccer; Computer Vision; SIFT; GPU.

Recebido em: 30/03/2015
Revisado em: 20/05/2015
Aprovado em: 29/06/2015_

1 INTRODUÇÃO

Futebol de robôs é uma iniciativa internacional voltada à pesquisa e à educação, visando promover desenvolvimentos ligados às áreas de inteligência artificial e robótica inteligente (KITANO et al., 1997).

Segundo Martins et al. (2006) o futebol de robôs desde a sua criação tem sido usado para pesquisa sobre desenvolvimento de robótica móvel e sistemas multiagentes, envolvendo as engenharias e a área da informática. Nesse jogo encontram-se vários problemas a serem estudados, como por exemplo, a construção mecânica, eletrônica, controle e reconhecimento dos objetos presentes.

De acordo com Martins et al. (2006) um sistema de visão computacional usado no futebol de robôs tem que ser rápido, eficiente, capaz de tolerar ruídos e variações luminosas. Várias técnicas podem ser usadas para o reconhecimento dos objetos do futebol de robôs, como as descritas por Bianchi e Reali-Costa (2000), que realiza uma calibração inicial das cores dos objetos e atualiza essas cores com a média das três últimas medidas. Isso é realizado, pois as cores mudam dependendo da região em que cada objeto se encontra. Após ter a cor correta de cada objeto em determinado momento é realizado um rastreamento dessas cores, encontrando assim cada objeto.

Segundo Shen (1998), jogadores robóticos necessitam realizar processos visuais de reconhecimento em tempo real, navegar em um espaço dinâmico, rastrear objetos em movimento, colaborar com outros robôs e ter controle para acertar a bola na direção correta. Já foram utilizados vários algoritmos como de Bianchi e Reali-Costa (2000) que é baseado em limiares adaptativos, ou seja, realiza a localização e identificação através do rastreamento de cores dos objetos, juntamente com um algoritmo que localiza o centro de objetos circulares desenhados na superfície superior dos robôs.

Os algoritmos mais utilizados para a localização dos objetos do futebol de robô são algoritmos que utilizam cores para identificar cada robô, porém é possível identificar esses objetos do futebol de robô utilizando outros tipos de algoritmo, como por exemplo, os algoritmos de identificação de pontos chave, utilizado nesse trabalho.

A escolha do algoritmo utilizado nesse trabalho se deu através dos resultados apresentados por Silva et al. (2013). Entre os algoritmos SIFT, SURF, FAST, STAR, MSER, e GFTT, os dois que tiveram melhores resultados na avaliação, segundo os autores, foram o SIFT (*Scale Invariant Feature Transform*) (LOWE, 2004) e ORB (RUBLEE et al., 2011). Nessa avaliação, realizada pelos autores, foram utilizadas imagens com

algumas degradações, como borramento, variação de escala, iluminação, rotação, ruído e todas as degradações juntas. Porém, o algoritmo ORB (RUBLEE et al., 2011) apesar de ser o mais rápido, segundo os autores, não encontrou a marcação dos robôs por causa da distância necessária entre o campo e a câmera, que dificulta a identificação dos pontos chave utilizando esse algoritmo. Dessa forma, foi utilizado nesse trabalho, o segundo colocado do trabalho de Silva et al. (2013), o algoritmo SIFT, que encontrou os marcadores dos robôs com a distância necessária entre o campo e a câmera.

Este trabalho propõe o uso do algoritmo de descrição de pontos chave SIFT para identificar os objetos do futebol de robô. Para encontrar a bola foi utilizada a transformada de Hough (HOUGH, 1959).

As demais seções deste trabalho estão organizadas da seguinte maneira: na Seção 2 são relatados os trabalhos relacionados, usados como base para o desenvolvimento deste trabalho; na Seção 3 é detalhado o referencial teórico; na Seção 4 é apresentada a metodologia proposta; a Seção 5 apresenta os experimentos realizados e os resultados obtidos; a Seção 6 apresenta a conclusão obtida com este trabalho.

2 TRABALHOS RELACIONADOS

No trabalho de Grittani et al. (2000) utilizou-se a transformada de Hough (HOUGH, 1959) para encontrar círculos e o detector de bordas Canny (CANNY, 1986). O modelo de marcação utilizado no trabalho é mostrado na Figura 1.

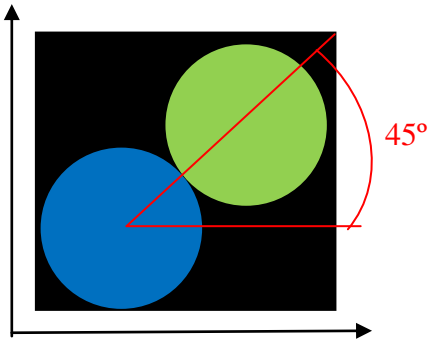


Figura 1. Modelo de marcação para diferenciação dos robôs.
Fonte: (GRITTANI et al., 2000).

No trabalho de Martins et al. (2006) foi proposto um sistema de visão computacional que rastreia as cores dos objetos no espaço de cores RGB (*Red*, *Green*, *Blue*). Os autores realizaram uma calibração inicial para identificar as cores dos objetos, pois as posições dos robôs são determinadas por meio das cores encontradas, de cada robô. Como as cores sofrem alterações durante o jogo, ficando mais claras ou mais escuras, dependendo da posição dos objetos, faz-se necessário realizar uma atualização das cores capturadas em frames anteriores, para que se tenha a cor mais exata possível dos objetos. Nesse caso é feita uma média das cores dos três últimos frames para

encontrar a cor mais ideal de cada robô, ou seja, a cor com menos variação de iluminação. A posição da bola é encontrada por meio do rastreamento da cor, e após isso são traçadas duas retas, uma na horizontal e outra na vertical, o centro da bola é o ponto no meio da reta vertical, como mostra a Figura 2, em que *Pini* é o ponto inicial e o *Pcentral* é o ponto do centro da bola.

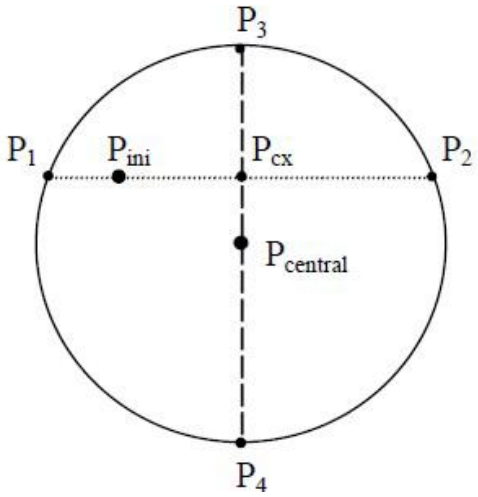


Figura 2. Determinação do centro da bola.
Fonte: (MARTINS et al., 2006).

Os autores Martins et al. (2006) identificaram os robôs usando um algoritmo de identificação de cores. Nesse processo foi encontrado primeiramente o círculo maior, e posteriormente o círculo menor (Figura 3), que está sempre a uma distância conhecida, em que se faz necessário percorrer a metade do diâmetro do círculo menor, até encontrar a cor esperada. É utilizado o mesmo algoritmo da bola para encontrar o centro dos dois círculos. O último passo é encontrar o centro do robô, que se obtêm ao traçar uma reta entre os dois centros dos círculos e encontrar o meio dessa reta.

A Figura 3 mostra um exemplo de um desenho superior de um robô com os centros dos círculos demarcados.

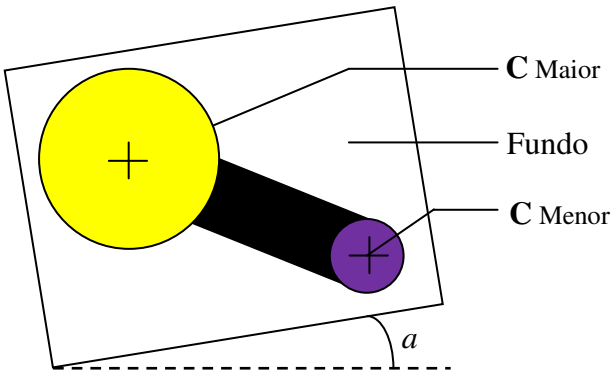


Figura 3. Identificação dos robôs.
Fonte: (MARTINS et al., 2006).

O trabalho de Assis et al. (2006) visou participar da IV Competição IEEE Brasileira de Robôs em 2006. A categoria escolhida pelos autores, a *Very Small Robots Soccer* do IEEE, representa uma categoria em que os robôs tem dimensões de 75mm x 75mm x 75mm e usam um sistema de transmissão de dados por RF (Rádio frequência). Antes de iniciar o processo de detecção dos objetos é realizada uma calibração inicial, para se ter a cor correta dos objetos, pois o sistema está sujeito a interferência, devido a variação de luminosidade. A Figura 4 mostra a tela utilizada para realizar o processo de calibração inicial. No trabalho foi utilizado o padrão de cores HSI (*Hue*, *Saturation and Intensity*), em que somente a Matiz (*Hue*) foi utilizada para identificar a cor, e com isso encontrar a cor de cada objeto.

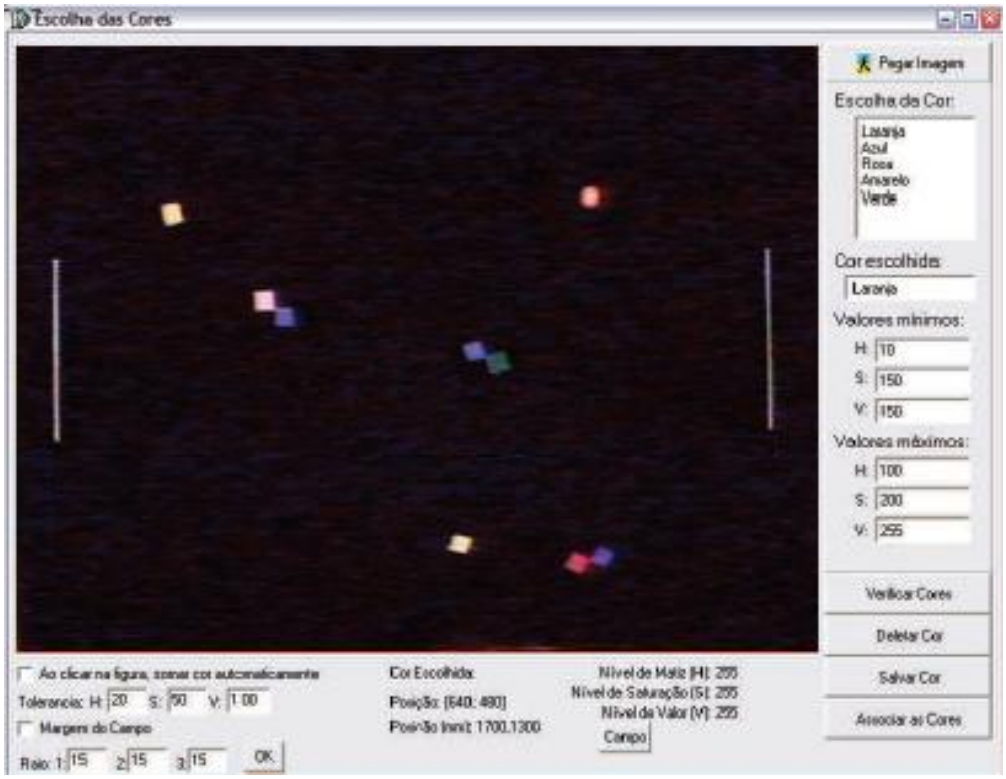


Figura 4. Tela da calibração inicial.

Fonte: (ASSIS et al., 2006).

No trabalho desenvolvido por Steckert et al. (2013), foram utilizadas marcações circulares. O algoritmo para encontrar os objetos é baseado na segmentação por um limiar (*thresholding*) e na transformada de Hough (HOUGH, 1959). Na segmentação por limiar são escolhidos pontos na imagem que estão entre um intervalo previamente determinado de cores, enquanto que na transformada de Hough são detectados pontos que possuem formas semelhantes ao modelo parametrizado, no caso, o círculo. Primeiramente é necessário realizar a calibração inicial para evitar "ruídos", que podem prejudicar a localização dos círculos. Conforme mostra a Figura 5, os pixels brancos são os que estão dentro do intervalo previamente escolhido. Para isso, alguns ajustes são feitos nos parâmetros mínimos e máximos do espaço de cor RGB.

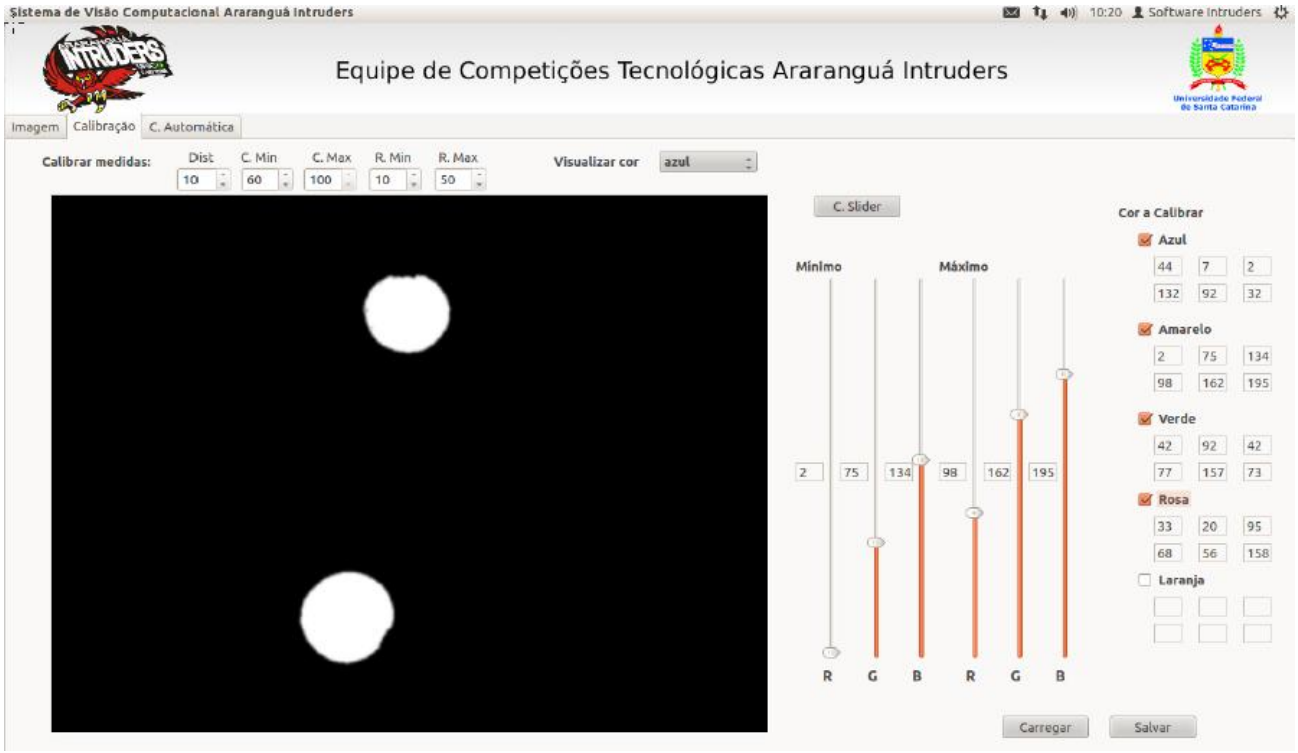


Figura 5. Tela de calibração inicial.
Fonte: (STECKERT et al., 2013).

Após realizar a calibração, são selecionadas as cores de interesse através da segmentação por limiar, excluindo as que não fazem parte dessa segmentação. Posteriormente é aplicada a transformada de Hough a fim de localizar formas circulares com o diâmetro especificado.

Para identificar cada robô, são utilizadas marcações com sequência de cores diferentes para cada um, e a cor do círculo central diferencia cada time.

A orientação de cada robô é dada pela distância do círculo mais próximo ao círculo central, conforme mostra a Figura 6.



Figura 6. Tela inicial do sistema de visão computacional da equipe de competições tecnológicas Araranguá Intruders.
Fonte: (STECKERT et al., 2013).

3 REFERENCIAL TEÓRICO

3.1 Algoritmo SIFT

A identificação de pontos homólogos (pontos correspondentes) em duas imagens não é uma tarefa simples, existindo muita pesquisa na área, para a sua execução automática. A primeira dificuldade está em

encontrar pontos de interesse (*keypoints*) em uma das imagens e, em seguida, localizá-los na outra. O algoritmo SIFT (*Scale Invariant Feature Transform*) (LOWE, 2004) é usado para encontrar pontos de interesse em pares de imagens. O algoritmo SIFT tem se mostrado muito eficiente para identificar e descrever pontos chave em uma imagem, o que é feito através de um mapeamento com diferentes vistas de um objeto ou cena, que resulta para cada ponto, um conjunto de informações que incluem: um vetor com 128 valores que descrevem cada ponto chave da imagem; a posição *x*, *y*; a escala *s*; e a orientação do gradiente em torno do ponto chave. O algoritmo consiste nas seguintes etapas (LOWE, 2004):

Detecção do Espaço-Escala: Os pontos chave são detectados aplicando uma filtragem em cascata que identifica candidatos, que são invariantes à escala, usando uma função que procura por descritores estáveis ao longo das diferentes escalas. O espaço-escala é definido como uma função $L(x,y,\sigma)$ na Equação 1, com uma imagem $I(x,y)$.

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \quad (1)$$

onde * indica a convolução em *x* e *y* com a Gaussiana $G(x,y,\sigma)$ (Equação 2).

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2)$$

Para detectar a localização dos pontos chave estáveis no espaço-escala, Lowe (1999) propôs o uso da função de diferença de Gaussianas (DoG) no espaço-escala convoluída com a imagem $I(x,y)$, resultando em $D(x,y,\sigma)$, a qual pode ser calculada a partir de duas escalas próximas separadas por um fator multiplicativo constante *k*, como na Equação 3.

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \quad (3)$$

O DoG é uma aproximação da escala normalizada do Laplaciano da Gaussiana $\sigma^2\nabla^2G$. O máximo e o mínimo de $\sigma^2\nabla^2G$ produz os descritores de imagem mais estáveis, quando comparado a diversas funções, tais como gradiente, Hessiano ou Harris.

Detecção de Extremos Locais: A partir de $D(x,y,\sigma)$, Lowe (1999) sugere que os máximos e mínimos locais devem ser detectados pela comparação de cada pixel com os seus 8 vizinhos na imagem corrente e 9 vizinhos nas escalas superior e inferior.

O candidato é preservado se ele for maior (para o máximo) ou menor (para o mínimo) que todos os seus 26 vizinhos. O próximo passo executa um ajuste detalhado nos dados da imagem local, determinando a posição e a escala. Pontos chave candidatos que possuem baixo contraste são rejeitados, por serem sensíveis ao ruído ou localizados ao longo das bordas. Para corrigir este problema, o algoritmo elimina os pontos

chave que possuem uma localização fracamente determinada.

Atribuição de Orientação: A escala do ponto chave é usada para selecionar a imagem suavizada pela Gaussiana *L*, com a escala mais próxima, em que toda a computação seja realizada de modo invariante à escala. O gradiente de magnitude *m*(*x*,*y*) é calculado com a Equação 4.

$$m(x, y) = \sqrt{\Delta x^2 + \Delta y^2} \qquad (4)$$

onde: $\Delta x = L(x + 1, y) - L(x - 1, y)$ e $\Delta y = L(x, y + 1) - L(x, y - 1)$. A orientação $\theta(x,y)$ é calculada pela Equação 5.

$$\theta(x, y) = \arctan(\Delta y / \Delta x) \qquad (5)$$

Em seguida, é construído um histograma de orientação (36 *bins*) a partir das orientações dos gradientes dos pontos amostrados, com uma região ao redor do ponto chave e, os picos neste histograma correspondem à direção dominante do gradiente local. Orientações com frequência acima de 80% do valor do pico no histograma são usadas para criar pontos chave com essa orientação. Desta forma, para múltiplas orientações de magnitude similar, existem múltiplos pontos chave criados no mesmo local e escala, mas com diferentes orientações.

Descrição dos pontos chave: Após a localização, escala e orientação da imagem terem sido atribuídas a cada ponto chave, é possível adotar um sistema de coordenadas bidimensional para descrever a região local da imagem e prover invariância com respeito a esses parâmetros. O próximo passo é calcular um descritor para a região da imagem local que é distinta e invariante a variações adicionais, tais como mudanças na iluminação ou ponto de vista 3D. Para alcançar a invariância de orientação, as coordenadas do descritor e as orientações do gradiente são rotacionadas em relação à orientação do ponto chave. Este procedimento é mostrado na Figura 7, onde o descritor de um ponto chave é criado pela determinação da magnitude e orientação do gradiente em cada pixel da imagem amostrada em uma região ao redor do ponto chave, como mostrado na Figura 7(a). Os descritores são ponderados em uma janela Gaussiana indicada pelo círculo. Estas amostras são acumuladas em histogramas de orientação, como mostrado na Figura 7(b).

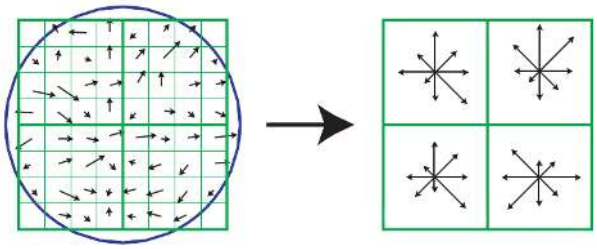


Figura 7. Descrição dos pontos chave. Fonte: (LOWE, 2004).

3.1.1 Correspondências entre duas imagens

Para se encontrar a correspondência entre duas imagens, é possível usar os pontos chave detectados com o algoritmo SIFT. Cada ponto chave está associado a um vetor de descritores com 128 dimensões, a comparação desses descritores torna possível encontrar a correspondência de uma imagem com outra. Lowe (1999) provou que a melhor correspondência para cada ponto chave é encontrada pela identificação de seus vizinhos mais próximos, que é definida minimizando a distância Euclidiana para os vetores de descritores.

Para evitar uma busca exaustiva, Lowe (1999) sugere o uso de uma estrutura de dados k-d *tree*, que suporta uma busca binária balanceada para encontrar o vizinho mais próximo dos descritores.

3.2 RANSAC

O algoritmo RANSAC (*RANdom SAmple Consensus*) proposto por Fischler e Bolles (1981) é um método de estimação robusto criado para extração de pontos de interesse de um conjunto de dados de entrada. Esse algoritmo é comumente utilizado em visão computacional para resolver o problema de correspondência entre duas imagens e estimar a matriz fundamental *M* correspondente entre as imagens.

A principal vantagem de utilizar o RANSAC é que a quantidade de pontos correspondentes entre as imagens que não são interessantes, não prejudica encontrar os pontos que realmente importam, por isso, ele é classificado como um estimador robusto, já que os estimadores de parâmetros clássicos, como por exemplo, os mínimos quadrados têm os seus resultados prejudicados caso na entrada haja muitos dados ruidosos.

Neste trabalho foi utilizado o RANSAC para remover os *outliers*, que são pontos de dados que não correspondem ao objeto procurado, deixando apenas os pontos de dados que realmente corresponde ao objeto procurado, conhecidos como *inliers*.

A Figura 8 mostra todos os pontos chave encontrados pelo algoritmo SIFT sem utilizar o algoritmo RANSAC, muitos pontos foram encontrados, porém, vários desses pontos não correspondem ao objeto procurado.

Conforme mostra a Figura 9, após usar o RANSAC, somente ficaram os pontos correspondentes (*inliers*) entre a cena e o objeto procurado.



Figura 8. Imagem com todos os pontos chave correspondentes sem utilizar o algoritmo RANSAC.
Fonte: (SILVA, 2012).

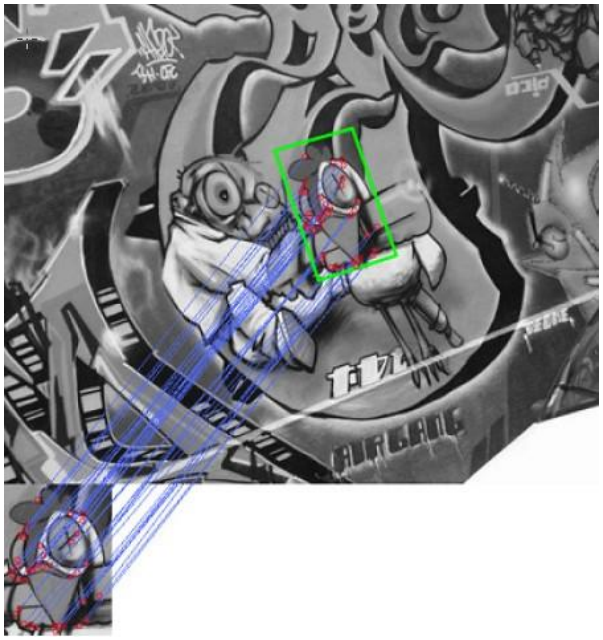


Figura 9. Imagem apenas com os pontos chave correspondentes (*inliers*) utilizando o algoritmo RANSAC.
Fonte: (SILVA, 2012).

3.3 Transformada de Hough

A Transformada de Hough foi desenvolvida por Paul Hough (HOUGH, 1959). É uma técnica para reconhecimento, em imagens digitais, que sejam parametrizadas, ou seja, que possuam uma equação com fórmula conhecida, tais como retas, círculos e elipses.

Segundo Macedo (2005), para a aplicar a transformada de Hough, normalmente é realizado um processamento anterior na imagem com o objetivo de identificar os contornos dos elementos que a compõem.

A ideia da transformada de Hough é transformar a imagem do espaço digital (x,y) para uma representação na forma dos parâmetros descritos pela curva que se deseja encontrar na imagem. Essa transformação é aplicada de modo que todos os pontos pertencentes a uma mesma curva sejam mapeados em um único ponto no espaço dos parâmetros da curva procurada.

Para isto, o espaço dos parâmetros é discretizado e representado na forma de uma matriz de inteiros, onde cada posição da matriz corresponde a um intervalo no espaço real dos parâmetros. Cada ponto da imagem que satisfizer a equação da forma paramétrica procurada incrementa de uma unidade o contador correspondente a sua posição, na representação discretizada (matriz).

O contador que tiver, no final do processo, o valor mais alto, corresponderá aos parâmetros da curva descrita na imagem.

Para encontrar o círculo, nesse caso a bola, utiliza-se a definição matemática da Equação 6, onde *a* e *b* são as coordenadas do círculo e *r* é o raio do círculo.

$$r^2 = (x-a)^2 + (y-b)^2 \qquad (6)$$

3.4 GPU

A unidade de processamento gráfico (GPU - G*raphics Processing Unit*) tornou-se uma parte integrante dos sistemas de computação geral da atualidade. Ao longo dos anos, tem havido um aumento significativo no desempenho e capacidades de GPUs. A GPU moderna não é apenas um motor gráfico poderoso, mas também um processador programável altamente paralelo com aritmética de pico e largura de banda de memória que supera substancialmente o seu homólogo CPU. Uma ampla gama de problemas complexos pode ser resolvida com o uso da GPU, o que tornou a GPU uma alternativa atraente aos microprocessadores tradicionais em sistemas de computação de alto desempenho do futuro (OWENS et al., 2008).

Nesse trabalho foi utilizada a GPU para melhorar o tempo despendido pelo algoritmo SIFT para encontrar os robôs.

4 METODOLOGIA

Os algoritmos implementados, neste trabalho, foram escritos na linguagem de programação C++ em classes que estão descritas na sequência. A classe “Localizar” é responsável por obter a posição e direção robôs, bem como a posição da bola.

Na classe “Bola” são armazenadas as coordenadas da bola (coordenadas *x* e *y*), na classe “Robo” tem-se as coordenadas de um robô, ângulo de direção, os *keypoints* e os descritores provenientes do algoritmo SIFT.

A função “Procura_Bola()” da classe “Localizar”, realiza a busca da bola utilizando a transformada de Hough, procurando pelo círculo na imagem da cena (imagem da vista superior do campo do futebol de robôs), mostrada na Figura 10.

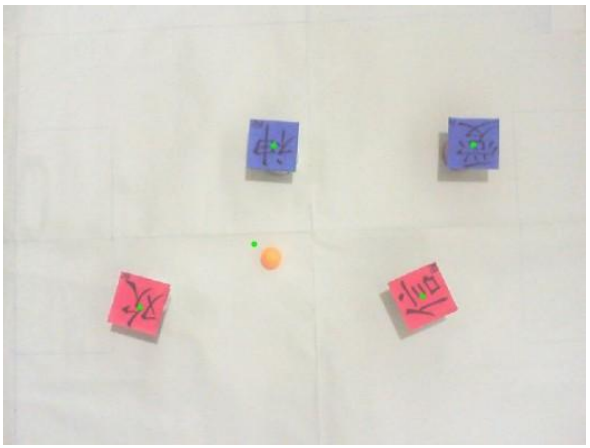


Figura 10. Imagem do campo de futebol de robôs visto de cima.

A bola é localizada a partir das cores que estão entre os intervalos determinados, no caso a cor laranja.

Para utilizar a transformada de Hough, inicialmente, deve-se retirar tudo que não for interessante, como ruído, ou até mesmo outros objetos que possam dificultar a busca,

para isso foi utilizada a função InRange da biblioteca OpenCV para encontrar a cor laranja referente a bola, como mostra a Figura 11. Essa função visa encontrar um valor que está entre um intervalo de valores previamente determinado, no caso a cor laranja, utilizando a Equação 7.

$$dst(I) = low(i)_0 \leq src(I)_0 \leq high(I)_0 \quad (7)$$

onde: $low(i)$ é o menor valor e $high(i)$ é o maior valor da faixa procurada na imagem $src(I)$, resultando na imagem $dst(I)$.

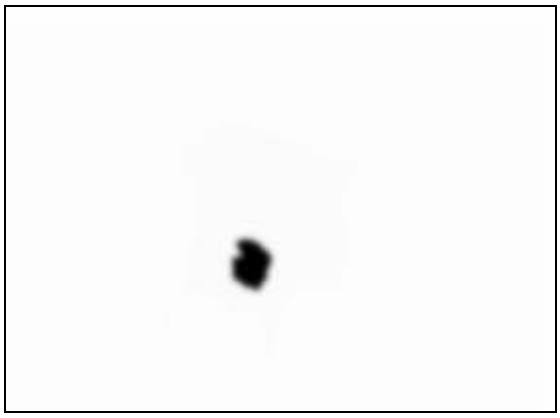


Figura 11. Imagem ajustada utilizando a função InRange, para em seguida aplicar a transformada de Hough.

Após realizar esse procedimento é aplicada a transformada de Hough, que encontra o círculo referente a bola, conforme mostra a Figura 12.

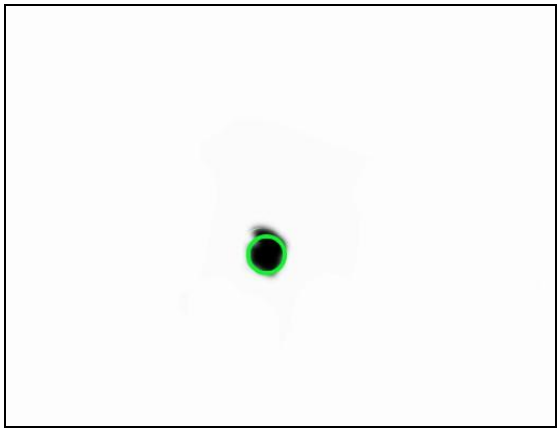
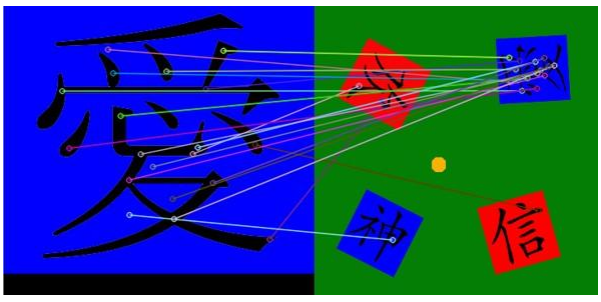


Figura 12. Transformada de Hough aplicada na imagem para a identificação da posição da bola.

A função "Procura_Robo()" retorna um objeto da classe "Robo" com todas as informações de sua localização e orientação. Para localizar o robô é utilizado o algoritmo SIFT na imagem da cena para encontrar pontos chave correspondente ao robô. Esses parâmetros são encontrados ao instanciar a classe "Localizar", que processa o algoritmo SIFT nas marcações utilizadas em cada robô e armazena as informações de descrição dos pontos chave de cada marcação. Após isso, somente é necessário processar o algoritmo SIFT na imagem da cena, pois as informações do objeto procurado já estão na instância da classe "Localizar". A Figura 13 mostra um exemplo contendo os *inliers* encontrados na busca da imagem do objeto procurado (Figura 13 (a)), e a imagem da cena (campo do futebol de robôs) (Figura 13 (b)).



(a) (b)

Figura 13. *Inliers* do objeto com a cena.

Após encontrar os pontos chave, faz se necessário calcular a orientação do robô, para descobrir em qual ângulo o mesmo está rotacionado. Esse ângulo é encontrado através da matriz de rotação, utilizando a

função trigonométrica, arco seno, das coordenadas (0,0) da matriz.

Tanto as informações de localização, coordenadas *x* e *y*, quanto a de orientação, são armazenadas no objeto robô que será retornado pela função "Procura_robo()". A Figura 14 mostra o diagrama de classes usadas no desenvolvimento do sistema de visão do futebol de robôs. Essas classes foram idealizadas buscando-se auxiliar projetos futuros que visam completar as outras partes do futebol de robô, como por exemplo, as estratégias utilizadas por cada equipe durante o jogo.

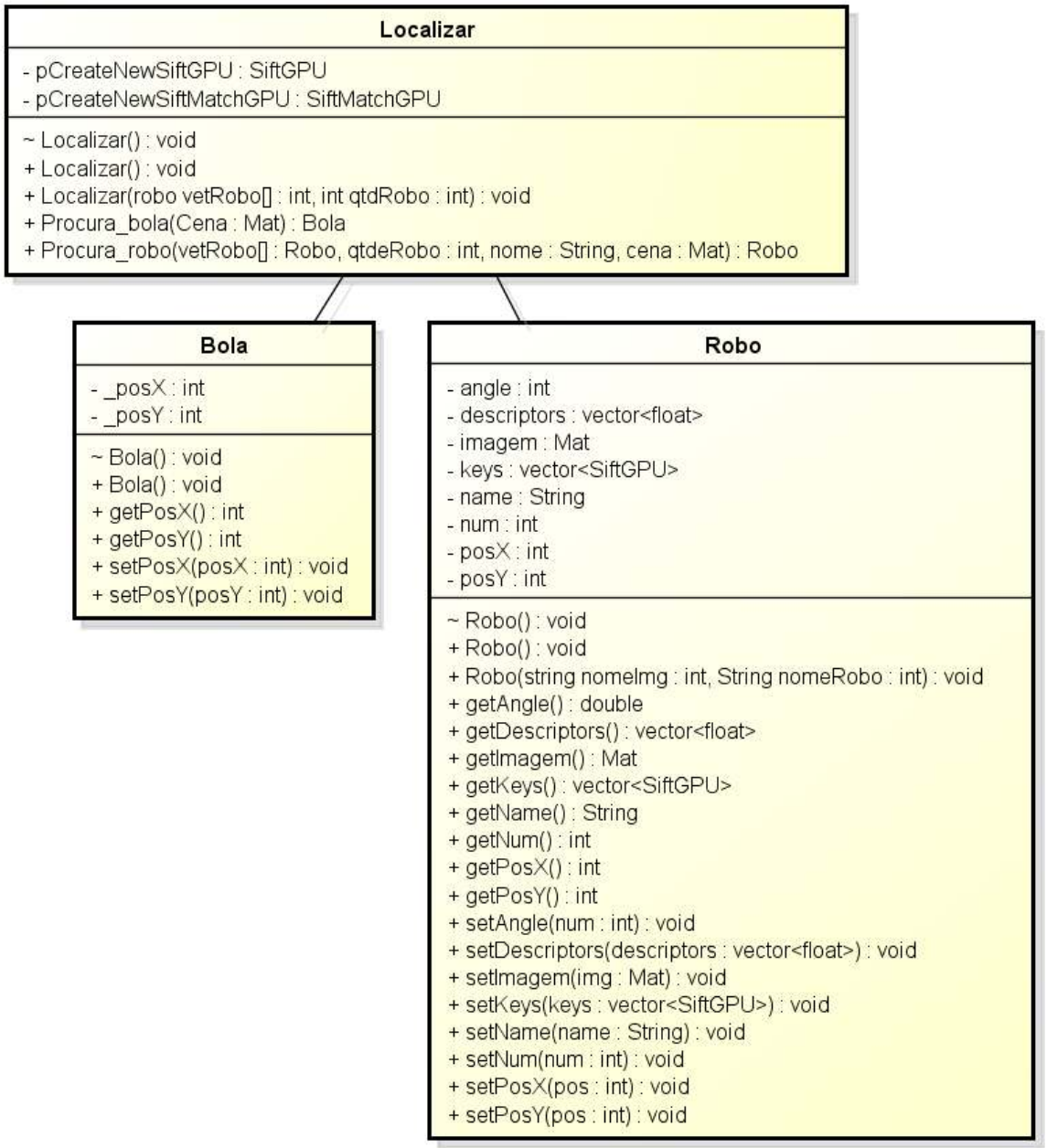


Figura 14. Diagrama de classes.

5 EXPERIMENTOS

Foram realizados vários experimentos nesse trabalho, o primeiro experimento foi para encontrar as melhores marcações para utilizar na parte superior dos robôs. As marcações escolhidas foram algumas palavras em chinês, por suas características serem muito específicas e serem facilmente encontrada pelo algoritmo SIFT. Após esse experimento, foram testados os dois algoritmos que obtiveram melhor desempenho no trabalho de SILVA et al. (2013), SIFT (LOWE, 2004) e ORB (RUBLEE et al., 2011) utilizando a biblioteca OpenCV. Como mostra a Figura 15, o algoritmo SIFT encontrou corretamente uma das marcações que é usada nos robôs com a distância da câmera ao campo (1.30 metros), porém com um custo computacional de 3 segundos.

A Figura 16 mostra que o algoritmo ORB não conseguiu encontrar uma das marcações que é usada nos robôs, por causa da distância necessária entre a câmera ao campo, que não pode ser muito grande.



Figura 15. Teste com o algoritmo SIFT.



Figura 16. Teste com o algoritmo ORB.

Apesar do algoritmo SIFT (LOWE, 2004) ter encontrado a marcação, o tempo de processamento (3 segundos aproximadamente) não é o suficiente para utilizá-lo nesta aplicação utilizando somente

a CPU do computador, isto porque até o algoritmo identificar todos os objetos e a bola, os objetos já mudaram de posição. Por isso, foi necessário buscar uma maneira de aumentar a velocidade de processamento, e a opção encontrada foi a execução do algoritmo SIFT utilizando a GPU (WU, 2007). Utilizando GPU, consegue-se ter um desempenho superior, o que pode alcançar, em alguns casos, a execução de um algoritmo aproximadamente dez vezes mais rápida. A GPU, no algoritmo SIFT, é utilizada para processar paralelamente a construção das pirâmides Gaussianas, detectar os DoG (*Difference of Gaussian*), mesclar o processamento com a CPU, para construir listas de pontos chave, juntamente com suas orientações e descritores (WU, 2006). Utilizando esse recurso, o tempo de processamento diminui, passando de três segundos para aproximadamente quatrocentos milésimos de segundo.

A Figura 17 mostra o tempo computacional gasto para o algoritmo encontrar cada marcação, utilizando a GPU e CPU.

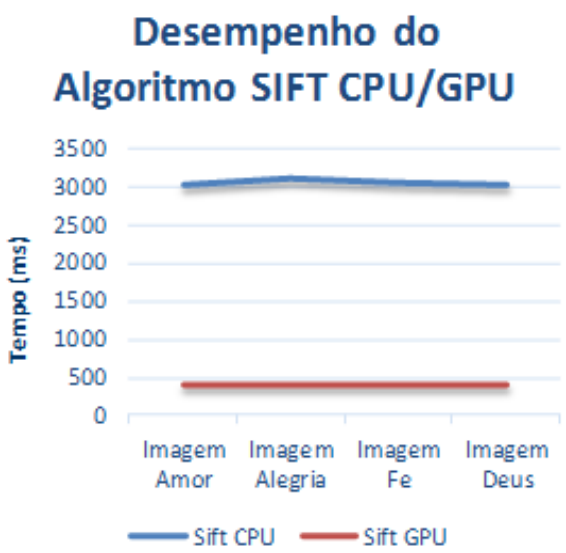


Figura 17. Desempenho do algoritmo SIFT utilizando a CPU e a GPU.

As marcações usadas nos robôs, nesse experimento, foram palavras chinesas, por serem facilmente encontradas pelo algoritmo SIFT, conforme mostra a Figura 18.



(a) (b) (c) (d)

Figura 18. Marcações utilizadas nos robôs. Em a) Deus, em b) Amor, em c) Alegria e em d) Fé.

Na Figura 19 é mostrado um campo simulado do futebol de robôs, contendo as quatro marcações dos robôs. No experimento de desempenho do algoritmo SIFT na CPU e GPU foi utilizada a Figura 19 como cena.

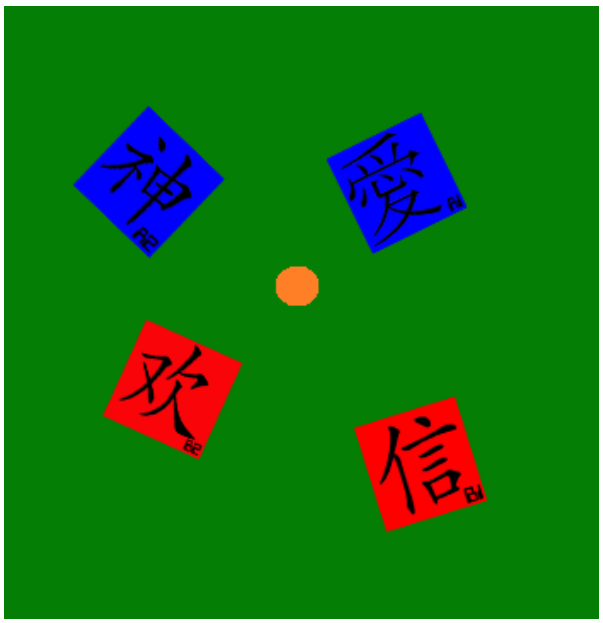


Figura 19. Campo simulado de robôs utilizado como cena.

6 CONCLUSÃO

Os resultados encontrados, neste trabalho, mostram que é possível inserir um algoritmo de descrição de pontos chave na parte da visão computacional do futebol de robôs.

O código fonte escrito neste trabalho foi desenvolvido para que seja aproveitado em trabalhos futuros, por isso foi utilizado o desenvolvimento orientado a objetos, que facilita a compreensão e o reaproveitamento do código.

O foco deste trabalho foi a investigação e o desenvolvimento da parte da visão computacional do futebol de robôs utilizando um algoritmo de descrição de pontos chave, até então não utilizado para essa finalidade. As outras partes que constituem um futebol de robôs (construção mecânica, controle dos robôs e estratégia de jogo) ficam em aberto para pesquisas em outros trabalhos.

REFERÊNCIAS

ASSIS, W.O. et al. Construção de robôs jogadores de futebol. IN: CONGRESSO DA SBC, 26. **Anais...** Campo Grande, MS, 2006. p.93-98.

BIANCHI, R.A.C.; REALI-COSTA, A.H. O sistema de visão computacional do time futepoli de futebol de robôs. 2000, São Paulo, SP. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA – CBA 2000, 13. **Anais...** Florianópolis, SC, 2000. p.2156-2161.

HOUGH, P.V.C. Machine analysis of bubble chamber pictures. In: INTERNATIONAL CONFERENCE ON HIGH ENERGY ACCELERATORS AND INSTRUMENTATION, 16. 1959.

KITANO, H. et al. RoboCup: a challenge problem for ai. **AI Magazine**, v.18, n.1, p.73-85, 1997. http://dx.doi.org/10.1145/267658.267738

LOWE, D.G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, p.91-110, 2004. http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

LOWE, D.G. Object recognition from scale-invariant features. In**:** INTERNATIONAL CONFERENCE ON COMPUTER VISION, Corfu, Grécia, 1999. p. 1150-1157. http://dx.doi.org/10.1109/iccv.1999.790410

MACEDO, M.M.G. **Uso da Transformada de Hough na Vetorização de Moldes e Outras Aplicações.** Tese (Mestrado) - Universidade Federal Fluminense, RJ, 2005.

MARTINS, M.F.; TONIDANDEL, F.; BIANCHI, R.A.C. Reconhecimento de objetos em tempo

real para futebol de robôs. In: JORNADA DE ROBÓTICA INTELIGENTE – ENCONTRO NACIONAL DE ROBÓTICA INTELIGENTE, 26. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Anais...** Campo Grande: Sociedade Brasileira de Computação, 2006. p.173-182.

OWENS, J.D. et al. **GPU Computing.** 2008. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4490127&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4490127>. Acesso em: 12 dez. 2014.

RUBLEE, E. et al. ORB: na eficiente alternative to SIFT or SURF. In: Metaxas, D.N. et al. (eds.) ICCV, **IEEE**, p.2564-2571, 2011.

SILVA, F.A. **Georreferenciamento automático de placas de sinalização com imagens obtidas com um sistema móvel de mapeamento.** 2012. Tese (Doutorado) – Universidade de São Paulo, São Carlos - SP. http://dx.doi.org/10.11606/t.18.2012.tde-31072012-115700

SILVA, F.A. et al. Evaluation of keypoint detectors and descriptors**.** In: WORKSHOP DE VISÃO COMPUTACIONAL (WVC 2013), 9. **Anais...** Rio de Janeiro: FGV, 2013.

STECKERT, T.; TRAMONTIN, E.D.; PEREZ, A.L.F. Sistema de geração de estratégias e visão computacional da equipe de futebol de robôs Araranguá Intruders. **Anais...** Santa Catarina: UNESC, 2013.