COMEC 4nd Edition Volume I

Bússola Visual para Futebol de Robôs
Visual compass for Robot Soccer

Fabrício Tonetto Londero[1], **Rodrigo da Silva Guerra**[1]

[1]Centro de Tecnologia– Universidade Federal de Santa Maria (UFSM)
Avenida Roraima, no 1000 – Cidade Universitária– Santa Maria – RS – Brasil

fabriciotonettolondero@gmail.com, rodrigo.guerra@ufsm.br

Abstract. *This paper describes a work on an alternative solution to the orientation problem by humanoid robots in RoboCup soccer matches. Using only a webcam, the robot does an analysis of frames received with frames previously stored to stipulate the orientation of the robot on the field. This technique, if successful, will present a solution to a very common problem in robot soccer matches: the own goal.*
Key Word: Visual compass. Visual orientation. Color-based Guidance.

Resumo. *Este artigo descreve um trabalho sobre uma alternativa de solução para o problema de orientação enfrentado por robôs humanoides em partidas de futebol da RoboCup. Usando apenas uma webcam, o robô faz uma análise de frames recebidos com frames armazenados previamente para prever sua orientação no campo. Esta técnica, se bem-sucedida, irá apresentar uma solução para um problema muito comum em jogos de futebol de robôs: o gol contra.*
Palavras Chaves: Bússola Visual. Orientação visual. Orientação baseada em cores.

1. Introdução

A RoboCup, principal competição de futebol de robôs do mundo, tem como objetivo ampliar as dificuldades a cada ano, para que, em 2050, haja um time de robôs humanoides apto a jogar contra os humanos vencedores da Copa do Mundo FIFA de futebol [Rob 2015]. Por exemplo, nos torneios de futebol da liga de robôs humanoides a bola era laranja e em algumas ligas ainda é laranja, o que facilita a sua identificação. A partir da edição de 2015, na liga de robôs humanoides, a bola passou a ser branca em pelo menos em 50% da sua superfície e o piso do campo foi trocado de carpete para grama artificial. Estes fatores afetaram drasticamente o desempenho das equipes naquele ano, de modo que apenas cinco de dezesseis times participantes conseguiram marcar qualquer gol [CBRobotica 2015].

Como as traves não possuem mais distinção e os lados do campo são simétricos [HUM 2015], as equipes necessitam elaborar uma nova forma para que os robôs se orientem dentro do campo e, principalmente, não marquem gols contra. Existem relatos de partidas da RoboCup no qual um time não entrou em campo, e saiu vencedor devido ao(s) gol(s) contra marcados pelo adversário. Um exemplo de oportunidade no qual a utilização de uma bússola visual poderia reverter o ocorrido.

Baseado no que foi apresentado e em uma carência de implementação acerca da orientação visual para a equipe Taura Bots(UFSM) de futebol de robôs, surgiu o interesse no estudo e elaboração deste trabalho sobre bússola visual para a orientação do robô dentro de um campo de futebol, que é simétrico.

Este artigo está estruturado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados. A Seção 3 traz o referencial teórico. Na Seção 4 é descrito como se deu o desenvolvimento da bússola visual e na Seção 5 são tratados os testes efetuados. Por fim, conclusões e trabalhos futuros são apresentados na Seção 6.

2. Processamento de Imagens e Visão Computacional

Muito se discute acerca dos conceitos que separam visão computacional e processamento de imagens, não sendo ainda bem definido. Caberia ao processamento de imagens, efetuar os devidos procedimentos em uma imagem, para torná-la adequada para a sua real utilização e/ou gerar uma nova imagem. Por sua vez, a visão computacional, que também recebe uma imagem como entrada, fica incumbida de gerar uma saída que seria a interpretação desta entrada, sendo total ou parcial [Marengoni and Stringhini 2009].

Para Gonzalez [Gonzalez et al. 2004], a área de processamento de imagens tem como característica, o extenso experimento de soluções em um problema, pois uma solução de processamento para uma imagem, pode não ser o adequado para outra imagem, por mais similares que elas sejam. Ou seja, é necessário grande esforço em aplicar diferentes níveis de ensaios e experimentos, para então chegar numa solução que se pode considerar aceitável.

Visão computacional pode ser definida, segundo Bradski e Kaebler [Bradski and Kaehler 2008], como a transformação de dados de uma câmera fotográfica ou de vídeo em uma decisão ou uma nova forma de representação, com o objetivo de atingir um objetivo prévio. Os autores salientam da importância de informações prévias, tais como, a existência de alguma pessoa na cena, a distância entre a câmera e algum objeto específico, se a câmera estava em movimentação e entre outras, para facilitar a transformação anteriormente citada e a precisão da mesma. Embora o ser humano seja uma criatura visual, pensa-se que o trabalho com visão computacional possa ser algo simples, o que é um grande equívoco.

Na Figura 1 pode-se notar a diferença entre processamento de imagens e visão computacional, onde mostra uma imagem original (esquerda) com problemas na iluminação que acarreta em dificuldade na identificação da placa do veículo, e na direita a imagem pós-processamento, onde se pode identificar com mais facilidade a placa do veículo em questão. Com a imagem de saída (pós-processada), pode-se aplicar a visão computacional para extrair informações da imagem, como no caso, a placa do veículo [Marengoni and Stringhini 2009], uma típica aplicação de visão computacional.

Com o que foi apresentado, fica evidente a interdependência entre visão computacional e processamento de imagens. Atualmente, são poucas as áreas da tecnologia que não façam uso de alguma forma de processamento de imagens, incluindo aplicações na área da saúde, segurança e redes sociais. Quando se fala de visão computacional, este estudo ainda é bastante novo, contendo poucos estudos e muito a ser feito [Bradski and Kaehler 2008]. Pode-se utilizar a Figura 2 para demonstrar a complexidade do estudo, onde o olho humano visualiza e identifica facilmente um veículo, mas

Figura 1. Imagem original (esquerda) e a pós-processada (direita), possibilitando à visão computacional de extrair informações (placa do veículo) [Marengoni and Stringhini 2009]

a câmera de um computador "enxerga" apenas sua representação em números, no caso, a matriz apresentada é a representação da área de abrangência do espelho retrovisor do lado esquerdo do veículo. Para efetuar correções em uma imagem, devem-se manipular estes números, existindo fórmulas e práticas para efetuar diferentes tipos de alterações, correções, melhorias ou simplesmente preparação de uma imagem. Mesmo com este conhecimento, se torna formalmente impossível a criação de um código para identificar automaticamente, sem intervenção humana, espelhos retrovisores em veículos baseado na leitura e comparações dessas matrizes numéricas [Bradski and Kaehler 2008].

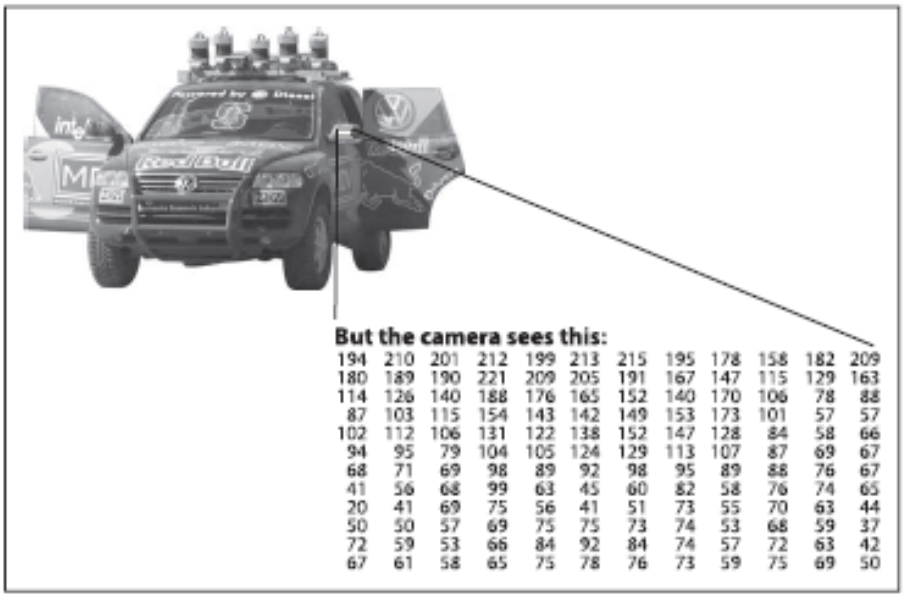


Figura 2. Veículo e como um computador "enxerga"o espelho retrovisor [Marengoni and Stringhini 2009]

Para se diminuir o grau de dificuldade quanto ao processamento de imagens e aumentar a compreensão básica, se deve tomar conhecimento quanto à procedência das imagens, tais como raio-x e ressonância magnética. Logo, a abrangência do estudo de processamento de imagens não é global, sendo desmembrado para uma melhor geração

de resultados e eficiência nos estudos, diminuindo consideravelmente a complexidade por ser estudado de maneira mais específica [Gonzalez and Richard 2002].

3. Trabalhos Relacionados

Apesar de escassos, existem trabalhos na literatura acerca da bússola visual. Dentre os mais similares a proposta deste trabalho, se encaixam os trabalhos de de Kok et al. [de Kok et al. 2013], que é uma extensão do trabalho de Sturm e Visser [Sturm and Visser 2009] e o trabalho de Bader [Bader et al. 2013]. Ambos utilizam informações extracampo para a orientação dos robôs.

Na proposta apresentada por Sturm e Visser [Sturm and Visser 2009] faz uso de redução de cores para apenas 10 cores, com o intuito de facilitar o desenvolvimento para melhorar o desempenho. Então, separa-se os *frames* em colunas verticais com uma largura pré-definida e em cada uma dessas colunas, se faz uma procura por um blob de cores anteriormente definida e distribuída em áreas externas ao campo. Esta abordagem fica limitada à procura dessas marcações localizadas fora do campo, tendo a certeza que estas cores não se repetiram. Ao encontrar o blob, efetua-se a contagem da troca de cores, e assim, estipulasse qual a orientação atual do robô. Sendo assim, a abordagem proposta não se preocupa em avaliar a similaridade entre imagens previamente armazenadas e as imagens provenientes da locomoção dos robôs, este trabalho também teve ênfase no futebol de robôs e possuiu resultados satisfatórios.

A abordagem apresentada por de Kok [de Kok et al. 2013] é bastante similar com a proposta deste trabalho assim como a proposta anterior, porém com algumas diferenças na forma de implementação; onde o foco do trabalho foi o futebol de robôs, com o intuito de auxiliar os robôs na orientação durante uma partida. Por exemplo, os autores dividem o campo de futebol em uma matriz e posicionam os robôs em cada uma destas regiões resultantes, recolhendo *frames* (imagens) de todas as direções para cada região. A abordagem elaborada por de Kok [de Kok et al. 2013] faz uso de comunicação entre os robôs para trocar informações referentes aos *frames* retirados de cada região, sendo desnecessário que todos os robôs se posicionem em cada uma das regiões, tendo em vista que recebem as demais informações dos outros robôs da sua equipe. Os autores utilizam o espaço de cores YUV422 e reduzem as cores para apenas 8; e então extraem colunas com largura fixa, e geram uma matriz que contem a contagem de troca de cores para cada uma dessas colunas. Com as matrizes resultantes, então, é efetuada uma comparação com as imagens provenientes do robô em locomoção.

O principal foco de De Kok [de Kok et al. 2013] foi a atualização da bússola conforme a movimentação dos robôs na fase de mapeamento e a estipulação da localização, e não da orientação dos robôs, combinando a abordagem com um módulo que envolve o algoritmo de Monte Carlo. Os estes foram efetuas em um cenário artificial sem alterações e obteve ótimos resultados, mas de pouca importância devido ao cenário não sofrer nenhum tipo de alteração ou interferência, segundo os próprios autores.

A proposta de Bader [Bader et al. 2013] também faz uso de fatores extracampo para a orientação, removendo o campo dos *frames* para a análise. São gravados *frames* de todas as direções a partir do centro do campo, gerando, assim, uma perspectiva 360º, e com estas informações é construído um muro virtual dividido em quadrados. Em cada um destes quadrados, é armazenado o histograma de cores para modelar o fundo do muro.

Bader compara os histogramas do seu muro com novos histogramas gerados dos novos *frames* provenientes da locomoção do robô no decorrer de uma partida.

4. Desenvolvimento da Bússola Visual

Na abordagem a ser apresentada, baseada no trabalho desenvolvido por de Kok et al.[de Kok et al. 2013], a bússola visual efetua uma análise e armazenamento de informações referente ao cenário ao entorno do robô, ação necessária antes de cada partida, pois o público se realoca para assistir uma partida, o que acarreta grandes mudanças no cenário. Então, o cenário na qual o robô se encontra no decorrer da partida é comparado com o cenário pré-jogo para estipular a orientação do robô.

Quase todos os trabalhos de visão computacional começam com a definição do que é necessário quanto a processamento de imagens. Primeiramente foram desenvolvidos alguns filtros de processamento de imagens para a aquisição de uma imagem mais adequada para se trabalhar, utilizando a biblioteca de visão computacional e processamento de imagens OpenCV neste protótipo. Estes filtros serão descritos em seguida.

A Figura 3 ilustra o processo como um todo, construído na Notação BPMN – *Business Process Model and Notation* [BPMN 2011] em forma de um diagrama independente de metodologia.

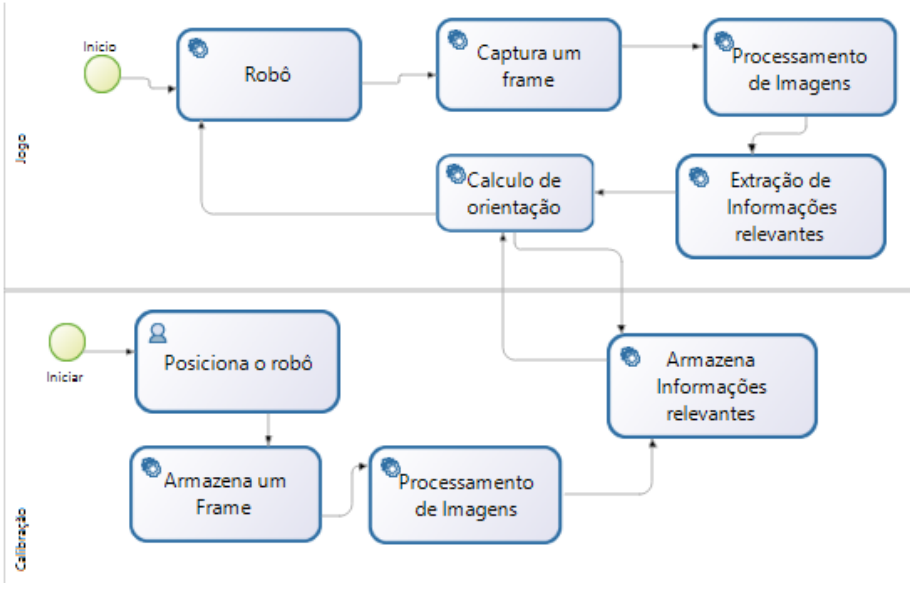


Figura 3. Notação BPMN do processo de calibração e funcionamento da bússola visual, para a estipulação da orientação

4.1. Tratamento de Imagens

Antes de cada partida de futebol, deve ser feita uma sequência de *frames* do campo, tirados de diferentes locais, tais como na Figura 4. O robô deve ser posicionado no centro de cada região (círculos) e capturar um quadro de cada direção, passo este denominado como calibração da Bússola.

Ao ser efetuada de maneira concisa, essa calibração diminui a chance de erros por parte do algoritmo de bússola visual, sendo assim, uma etapa crucial para o funcionamento desta abordagem.

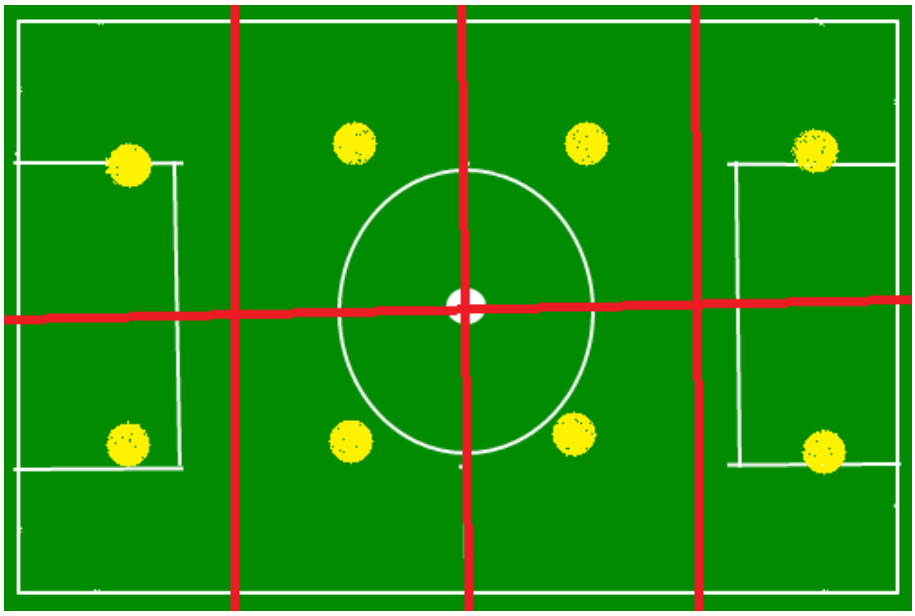


Figura 4. Representação de um campo de futebol dividido em regiões

Na sequência, para cada *frame* retirado, executam-se etapas de processamento de imagens com o objetivo de acarretar maior similaridade entre as cores presentes, podendo este ser um passo opcional, aplicado somente para deixar as cores mais homogêneas, ou seja, remover ruídos na imagem. No caso, foi aplicado transação morfológica de abertura (MORPH_OPEN da OpenCV) a qual foi obtida pela erosão da imagem seguida por uma dilatação, o que acarreta na remoção de pequenos ruídos [OpenCv Documentation 2016].

O código da erosão começa com a definição do tamanho da erosão e da definição da estrutura morfológica para então executar a função erode da biblioteca OpenCV.

```
1
2 int erosion\_size = 2;
3 Mat element = getStructuringElement(cv::MORPH\_OPEN,
4                 cv::Size(2 * erosion_size + 1, 2 * erosion_size
                     + 1),
5                 cv::Point(erosion_size, erosion_size));
6 erode(image, image, element);
7 .
```

A cor em uma imagem pode ser representada pelas intensidades dos canais vermelho, verde e azul no sistema de cores RGB(*red*, *green*, *blue*). Após ocorrer a erosão que pode ser vista na Figura 5 (b), a imagem foi transformada do padrão RGB para o HSV (*hue*, *saturation*, *value*). Este padrão permite uma melhor distinção entre cores puras por trabalhar com tonalidade ou matiz (*hue*), saturação e valor (*value*, que seria a luminosidade ou brilho da imagem). O espaço HSV separa melhor em *clusteres* as cores conforme as classes percebidas pelo olho humano. Independente do padrão utilizado, uma pequena alteração em um dos valores (de intensidade de canal, por exemplo), já não se considera como a mesma cor.

Apenas as transformações para HSV após uma erosão, que podem ser vistas

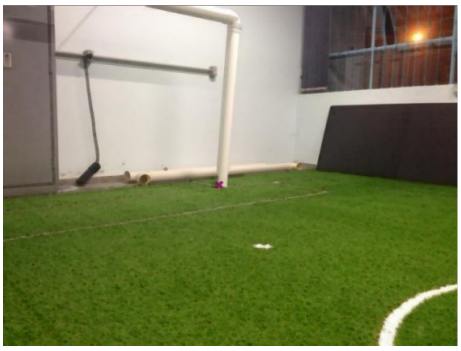
[a]  [b] 

Figura 5. Imagem original [a] e imagem após executar uma erosão [b]

na Figura 6 (a), não se mostram suficientes para a abordagem proposta, então, após a transformação, foi utilizado um algoritmo para redução de cores [Laganière 2011].

Um algoritmo de redução de cores faz-se necessário para este trabalho devido à abordagem trabalhar com identificação de troca de cores de cada coluna. A ausência da etapa de redução de cores resultaria em um trabalho massivo, porque quaisquer pequenas variações de cores implicariam novas mudanças a serem processadas pela abordagem. O código de redução pode ser visto em [Laganière 2011]. O resultado da redução pode ser visto na Figura 6 (b).

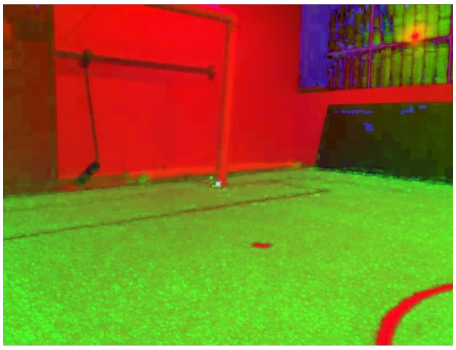
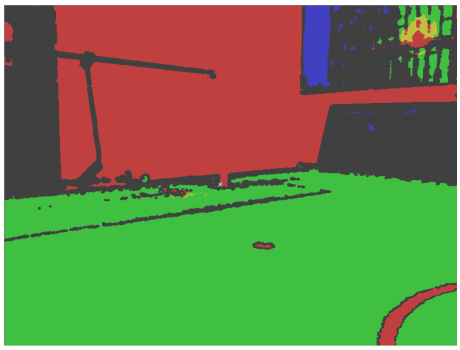
[a]  [b] 

Figura 6. Imagem HSV resultante [a] e após a redução de cores [b]

Pode-se ainda aplicar alguns outros filtros para melhorar a imagem e assim facilitar a identificação de padrões ou segmentação de objetos; julgou-se, contudo, suficiente o que foi aplicado até o momento.

Com o *frame* resultante, separa-se a imagem em colunas de 10 pixels de largura, nas quais são analisados e extraídos os seus histogramas (ocorrência de cada cor na imagem) e onde as trocas de cores ocorrem [de Kok et al. 2013]. Esses dados são necessários para análise das colunas dos *frames* que surgirão para comparação, conforme a movimentação do robô. A Figura 7 destaca as áreas de interesse. Estas áreas foram analisadas gerando um arquivo de texto contendo informações de cada coluna. A escolha por gerar estes arquivos de texto foi pela experiência adquirida em competições de futebol de robôs, nas quais ocorrem muitos desligamentos dos robôs por problemas de software e hardware, como mau contato na bateria, ou mesmo o término desta. Desta forma, mantêm-se os arquivos caso um *reboot* ocorra no computador do robô, não sendo necessário recalibrar a bússola visual.

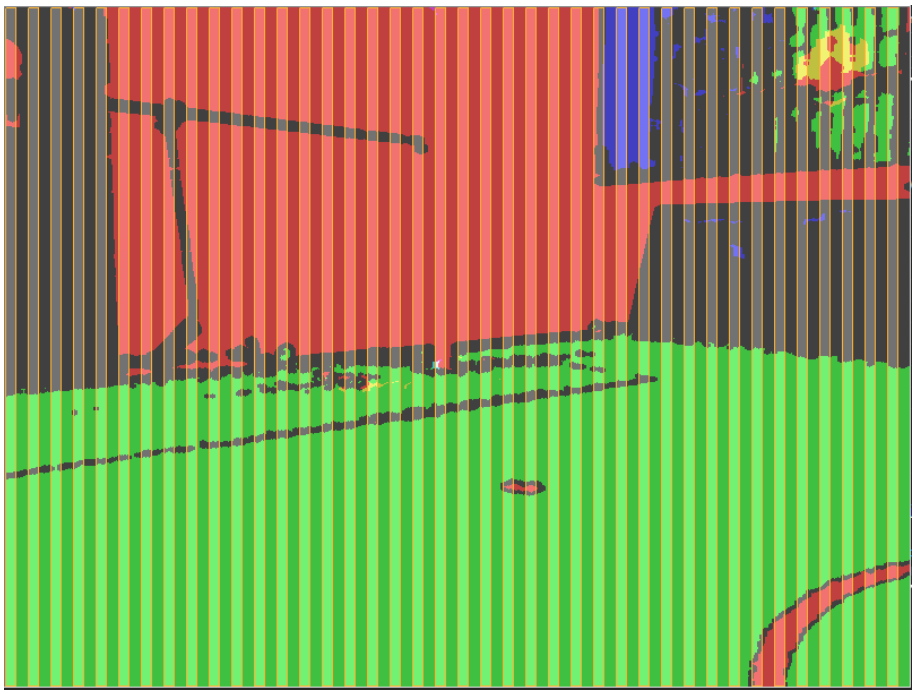


Figura 7. Imagem com destaques nas áreas de interesse (colunas de 10 pixels de largura)

O arquivo de texto anteriormente citado deve conter as trocas de cores que ocorreram, por exemplo:

```
1
2 64 64 64(cinza) para 192 64 64 (vermelho): 23
3 192 64 64(vermelho) para 64 64 64 (cinza): 8
4 64 64 64 (cinza) para 64 64 192 (azul): 14
5 64 64 192 (azul) para 64 64 64 (cinza): 4
6 .
```

Cada linha do arquivo traz a contagem de troca de cores, sendo a primeira cor a origem e a segunda o destino, seguida pela quantidade de vezes que esta troca de cor ocorreu. Cada arquivo deve representar uma coluna de 10 pixels anteriormente selecionada. Com essas informações, tanto da bússola visual como do novo *frame*, analisam-se as colunas para estipular a similaridade entre elas, a fim de direcionar o robô para o gol adversário ou evitar que o mesmo efetue um gol contra.

4.2. Cálculo de Similaridade

Para o cálculo de similaridade foi utilizado a Similaridade de Cosseno. Esta métrica é utilizada em mineração de dados e comparação de documentos. Existindo palavras em comum, método como a Distância Euclidiana e os cálculos de coeficiente de correlação de Pearson tornam-se inapropriados [Tan 2005].

Para se calcular a Similaridade de Cosseno, deve-se gerar um vetor dos atributos que estão sendo analisados, no caso da bússola visual, a contagem de trocas de cores que ocorrem em cada coluna. O vetor é utilizado para encontrar o produto escalar normalizado entre os dois vetores que estão sendo analisados, neste caso, um proveniente da bússola

visual e outro do robô durante a partida. Então, calcula-se o cosseno do ângulo entre os dois vetores; quanto mais próximo a 1, mais similar são os objetos, consequentemente, quanto mais próximo a -1, mais divergentes são os objetos.

$$similarity(x, y) = cos(\theta) = \frac{x \cdot y}{||x|| * ||y||}$$

Figura 8. Equação matemática da Similaridade de Cosseno

A Figura 8 apresenta a equação matemática da Similaridade de Cosseno, no qual X e Y são respectivamente atributos dos vetores que estão sendo comparados.

O primeiro passo foi gerar matrizes que correspondessem às trocas de cores que ocorrem em cada coluna, e então aplicar o algoritmo da Similaridade de Cosseno para efetuar a comparação entre essas matrizes.

	0	0	1	0	4	0	0	0
	0	0	0	25	76	0	0	0
	0	0	0	0	1	0	0	0
	0	27	0	0	2	0	0	0
	0	64	0	4	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Figura 9. Representação de uma matriz de troca de cores

As matrizes foram criadas baseada no número de cores existentes nas imagens, no caso, existem apenas oito cores resultantes do algoritmo de redução de cores. Cada uma dessas cores (amarelo, cinza, vermelho, azul, verde, rosa, branco e ciano) foram definidas como índices para as linhas e colunas da matriz, como pode ser visto na Figura 9. O conteúdo das matrizes representa a troca de cores existente entre as cores das linhas com as cores das colunas, por exemplo na Figura 9, houve 64 trocas de verde para cinza.

Após gerar uma matriz para cada coluna, percorrem-se as matrizes da bússola e a comparação é feita utilizando a Similaridade de Cosseno, uma a uma, com as matrizes dos *frames* teoricamente provenientes do (s) robô (s), ou seja, cada matriz da bússola é comparada com todas as matrizes dos frames.

Com a execução dos cálculos de similaridade, obtém-se valores que correspondem à similaridade de cada uma das matrizes (colunas), e assim, as similaridades de maiores valores correspondem a área de interesse, ou seja, a direção na qual o robô deve se direcionar. Mas para isso, apenas uma coluna não pode servir como parâmetro, e sim um aglomerado delas, levando as matrizes vizinhas como parâmetro.

5. Testes

Como a bússola visual aqui apresentada ainda não foi testada em competições de robótica e em um robô, os testes foram efetuados com imagens que simulassem o ambiente a

ser enfrentado pelo(s) robô(s). Foram armazenados *frames* para simular os arquivos da bússola após a calibração das mesmas e então foi feita uma comparação com outros *frames*, para se estipular a similaridade entre eles. Ambos passam pelos mesmos algoritmos de processamento de imagens que resulta em imagens tais como a da Figura 6.

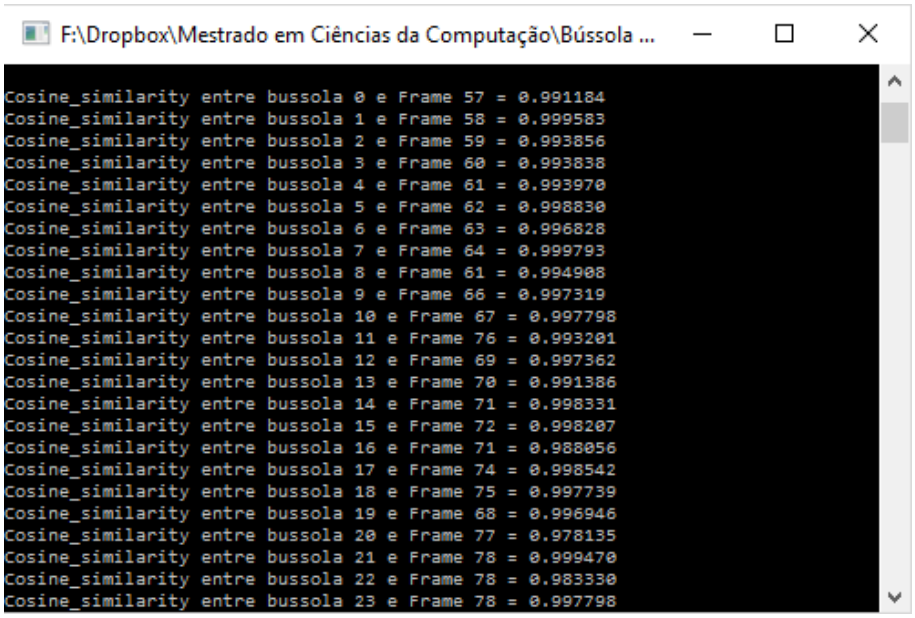


Figura 10. Execução do protótipo

Um exemplo de resultado obtido pode ser visto na Figura 10, na qual são comparadas todas as colunas (matrizes) da bússola (Figura 11(a)) com todas as colunas dos *frames* teoricamente provenientes do robô (Figura 11(b)). Então, é apresentado o índice referente às colunas da bússola e qual coluna do *frame* proveniente do robô é definido como o mais similar com a matriz da bússola em questão, seguido pelo valor de similaridade resultante. Nota-se que existe uma sequência lógica de similaridade das matrizes da bússola no qual o índice 0, comparado com as matrizes dos *frames* analisados, possui como a coluna mais similar à de índice 57, e para a coluna 1 da bússola, a mais similar é a 58 do *frame*, e assim sucessivamente. Sendo assim, as imagens possuem uma(s) determinada região idênticas entre si, resultando uma similaridade suficiente para a orientação do(s) robô(s).

Após ter a similaridade para cada matriz calculada, soma-se à essa os dois próximos valores das áreas à esquerda e à direita. A soma resultante de maior valor é considerada a região mais similar. Sabendo quais as áreas de interesse pertencentes a região similar entre as imagens, calcula-se o ângulo com uma simples regra de três, onde se faz necessário saber o ângulo de abertura da imagem da bússola e o índice da matriz mais similar (bússola) e a contagem total de matrizes. Informação esta que um robô ou veículo autônomo pode utilizar para estimar sua orientação, adicionando esta informação a sua tomada de decisão, que pode ser algo como ir até algum objetivo e voltar para ”casa”, ou evitar gols contras no futebol de robôs, que é um problema muito comum e impactante.

(a) Original

(b) Utilizada nos testes

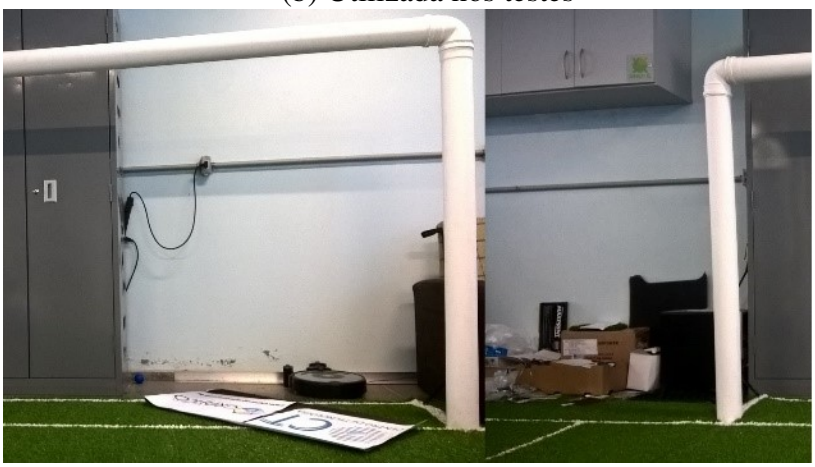


Figura 11. Imagem original (a) e imagem a ser comparada (b)

6. Conclusão

A presente solução mostra-se promissora para, com alguma evolução, funcionar tal qual uma bússola nos robôs que jogam futebol, ficando como objetivo futuro aplicar o trabalho em uma partida real. O próximo passo é a definição de como será feita a troca de informações com a inteligência artificial do robô. Neste protótipo não se teve grande ênfase no desempenho e na pouca utilização de recursos computacionais, características estas que devem ser trabalhadas num futuro próximo, além de adaptá-las para a utilização real em um robô humanoide que joga futebol, no caso, para a equipe Taura Bots.

Ainda existem arestas a serem corrigidas ou melhor definidas, tais como: que tipo de informação a bússola deve retornar ao robô e como esta comunicação é feita. As possibilidades são de o código da bússola visual retornar um valor booleano informando se o robô está direcionado para o lado certo (virado para o adversário) para então o mesmo fazer a busca por goleiras e afins para marcar o gol. Podem-se estipular valores para que a bússola visual retorne para a inteligência artificial do robô, dados para que o robô siga em frente, vire-se para a direita ou esquerda, ou para se alinhar corretamente ao campo adversário.

Outra possível abordagem que, teoricamente, é menos suscetível a erros é o uso

da bússola visual somente para análise do ambiente correspondente às goleiras, na qual se utiliza um algoritmo para localiza-las e, então, proceder à bússola visual para a análise de direção.

Referências

(2015). *RoboCup Humanoid League Technical Committee*. RoboCup Soccer Humanoid League rules and setup for the 2015 competition.

(2015). *RoboCup League Technical Committee*. RoboCup Soccer League rules and setup for the 2015 competition.

Bader, M., Prankl, J., and Vincze, M. (2013). Visual room-awareness for humanoid robot self-localization. *arXiv preprint arXiv:1304.5878*.

BPMN (2011). Notation (bpmn) version 2.0. *OMG Specification, Object Management Group*.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.".

CBRobotica (2015). Humanoid and standard platform leagues. Disponível em ¡http://www.cbrobotica.org/wp-content/uploads/Humanoid-and-Standard-Platform-Leagues.pdf/¿.

de Kok, P. M., Methenitis, G., and Nugteren, S. (2013). Victoria: Visual compass to orientate accurately.

Gonzalez, R. C. and Richard, E. (2002). Woods, digital image processing. *ed: Prentice Hall Press, ISBN 0-201-18075-8*.

Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2004). Digital image using matlab processing. *Person Prentice Hall, Lexington*.

Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Publishing Ltd.

Marengoni, M. and Stringhini, S. (2009). Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, 16(1):125–160.

OpenCv Documentation, M. T. O. (2016). Morphology transformations. Acesso em: 12 dez. 2015. Disponível em http://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html.

Sturm, J. and Visser, A. (2009). An appearance-based visual compass for mobile robots. *Robotics and Autonomous Systems*, 57(5):536–545.

Tan, P. (2005). Ms and kumar, v.: Introduction to data mining.