Desenvolvimento de Aplicativos de Visão Computacional para Plataforma Robótica Móvel

[1]*Luiz Eduardo G. Martins,* [2]*André de Andrade Bindilatti,* [3]*Antonio Valério Netto*

[1]UNIFESP – Universidade Federal de São Paulo
Instituto de Ciência e Tecnologia – São José dos Campos
legmartins@unifesp.br

[2]UNIMEP – Universidade Metodista de Piracicaba
Faculdade de Ciências Exatas e da Natureza
andre_a_a@yahoo.com.br

[3]Xbot – Robótica Móvel Inteligente para Educação, Pesquisa e Entretenimento
valerio@xbot.com.br

Resumo. O objetivo deste artigo é apresentar os resultados preliminares do desenvolvimento de aplicativos de Visão Computacional para a plataforma robótica móvel *RoboDeck*. Os aplicativos desenvolvidos têm a finalidade de reconhecimento de sinais que permitam ao robô navegar de forma autônoma em um ambiente fechado. Os sinais a serem reconhecidos são setas para à direita, esquerda, em frente e pare. Testes comparativos foram realizados e mostram os resultados em termos da precisão e tempo de resposta no reconhecimento dos objetos.

Palavras-Chave: *Visão Computacional, Robótica Móvel, RoboDeck.*

1. Introdução

Nos últimos anos vem sendo cada vez maior o emprego de técnicas de Visão Computacional em sistemas que necessitam capturar, processar e analisar imagens, com a finalidade de tomada de decisões. Este artigo apresenta resultados preliminares sobre o desenvolvimento de aplicativos de Visão Computacional para uma plataforma de robótica móvel. A plataforma adotada foi o *RoboDeck,* desenvolvida pela empresa brasileira XBot. O objetivo do desenvolvimento dos aplicativos de Visão Computacional é munir o *Robodeck* da capacidade de "enxergar" o ambiente que o rodeia, de tal forma que o robô se torne autônomo em relação a sua locomoção.

O restante deste artigo está organizado da seguinte forma: a seção 2 apresenta uma visão geral sobre a plataforma robótica móvel *RoboDeck*; a seção 3 discute as principais técnicas de Visão Computacional que foram empregadas no desenvolvimento dos aplicativos; a seção 4 relata os aplicativos desenvolvidos, apresentando uma comparação entre as implementações e alguns testes preliminares sobre a eficácia de reconhecimento de objetos, bem como o desempenho dos aplicativos; a seção 5 apresenta os comentários finais e aponta para trabalhos futuros.

2. Plataforma Robótica Móvel

A robótica móvel tem se desenvolvido muito nas últimas décadas, e diversas plataformas podem ser encontradas para diferentes nichos de pesquisa e desenvolvimento. Com relação aos tipos de plataformas robóticas podemos classificá-las em: plataformas para robôs aéreos, terrestres e aquáticos . Nesse trabalho vamos nos concentrar nas plataformas robóticas terrestres, particularmente aquelas em que a movimentação do robô está baseada em rodas.

A plataforma robótica adotada neste trabalho é o *RoboDeck*, desenvolvido pela empresa brasileira *XBot* (sediada na cidade de São Carlos, Estado de São Paulo – Brasil. www.xbot.com.br). A seguir, apresentamos uma visão geral da plataforma *RoboDeck*.

2.1 Plataforma *RoboDeck*

O *RoboDeck* é uma plataforma robótica móvel (terrestre) baseada em rodas. A composição mecânica do robô está assim constituída [1]: quatro rodas independentes (omnidirecionais), cada uma controlada por um servo-motor de direção (com capacidade de direcionar ângulos de 45º para dentro e 45º para fora), as duas rodas dianteiras também são controladas por motores de tração; dois motores de tração responsáveis por colocar o robô em movimento, cada motor de tração possui um *encoder* capaz de medir a quantidade de giros do motor; um chassi de sustentação do robô com *payload* de 15 kg. A Figura 1 apresenta uma fotografia de um *RoboDeck* montado.

Figura 1 – Carroceria do *RoboDeck*.

O sistema de sensores do *RoboDeck* está configurado da seguinte forma [1]: quatro sensores infra-vermelho (com capacidade de detecção de objetos na faixa de 4 a 30 cm), sendo três localizados na parte da frente e um na parte traseira do robô; quatro sensores ultra-som, localizados na parte da frente, traseira, esquerda e direita do robô (com capacidade de detecção de objetos na faixa de 3 a 600 cm, cobrindo um feixe cônico de aproximadamente 45º); um sensor de temperatura (em Cº); um sensor de umidade relativa do ar (%); um acelerômetro para medir a força de gravidade aplicada ao robô (decomposta nos eixos x,y,z); uma bússola para informar a orientação do robô; um dispositivo de GPS capaz de informar a posição geográfica do robô (latitude, longitude e altitude), a data e horário do meridiano de Greenwich, a velocidade de deslocamento, e o sentido de deslocamento do robô em relação ao norte magnético; e uma câmera para captura de imagens, com interface USB (esse é um recurso opcional, disponível junto à placa de alta performance). A Figura 2 ilustra a disposição dos sensores do *RoboDeck*.

O *RoboDeck* suporta até seis baterias (recarregáveis) de 18 V / Ah, embora o aconselhado seja o uso de quatro baterias (uma de cada lado). As baterias do *RoboDeck* lhe dão uma autonomia de até 4,5 horas. O sistema eletrônico do *RoboDeck* está configurado da seguinte forma: uma placa mãe com um processador ARM9, responsável pela execução dos comandos robóticos de baixo nível; uma placa ZigBee responsável pelo sistema de comunicação de controle do robô; uma placa de alta performance AMD Geode™ para a execução do MAP (Módulo de Alta Performance), responsável por habilitar o sistema de captura de imagens do robô e aplicativos autônomos (outras placas de alta performance podem ser usadas, desde que suportem execução do sistema operacional *Linux*) ; uma fonte reguladora de tensão. A Figura 3 apresenta as placas de processamento do robô.

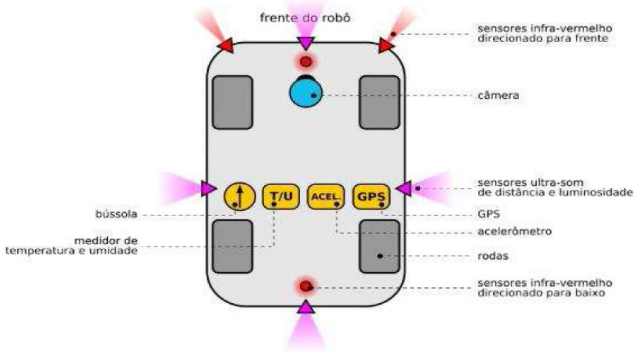

Figura 2 – Disposição dos sensores do *RoboDeck* [1].

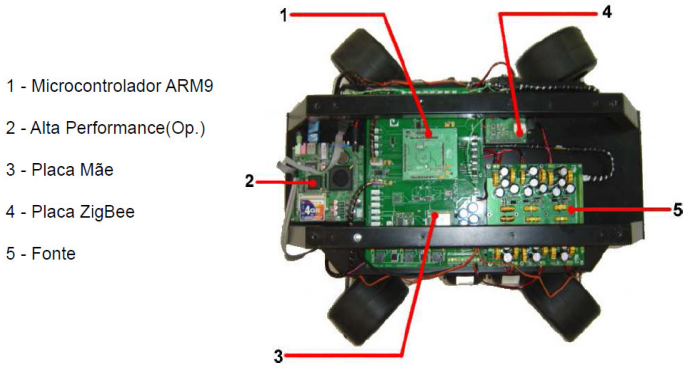

Figura 3 – Placas de processamento do *RoboDeck* [1].

A arquitetura de software do *RoboDeck* está dividida em quatro módulos [2]: Módulo de Controle de Seção (MCS), Módulo de Controle Robótico (MCR), Módulo de Controle de Comunicação (MCC) e Módulo de Alta Performance (MAP). A Figura 4 apresenta uma visão geral da arquitetura de software do *RoboDeck.* O MCR roda no processdor ARM9, e é responsável por comandar os sensores e atuadores do robô. O MCS implementa o controle de identidade do robô, sendo responsável por armazenar informações como identificador único, nome, versão do robô, versão do protocolo de comunicação, entre outras. O MCC é responsável pelo controle de comunicação do robô, permitindo que controladores externos enviem comandos ao robô, que passa a atuar na modalidade *master-slave*. O controle de comunicação foi implementado com

base no protocolo de comunicação sem fio *zigbee* [3]. O MAP permite a integração de aplicativos robóticos capazes de atuar como controladores do *RoboDeck*. Os aplicativos podem expandir os comandos básicos do robô, fornecendo ao mesmo maior autonomia, como por exemplo, navegação autônoma inteligente.

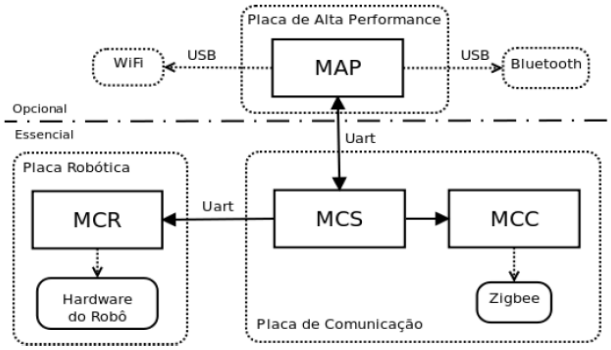

Figura 4 – Visão Geral da Arquitetura de Software do *RoboDeck* [2].

3. Técnicas de Visão Computacional na Robótica Móvel

No trabalho realizado até o momento, uma variação da técnica de *template matching* foi implementada como abordagem para o reconhecimento de formas em imagens digitais. Como alvo para reconhecimento, sinalização de setas indicativas para direita e esquerda foram utilizadas em testes práticos. O processamento realizado sobre as imagens nessa abordagem, pode ser descrito por meio dos seguintes passos:

(1) **Segmentação da imagem:** essa etapa é necessária para possibilitar que objetos de interesse sejam detectados na imagem e processados individualmente;
(2) **Detecção dos componentes isolados:** após obter uma imagem segmentada, é realizado um novo processamento para se extrair os diferentes componentes isolados na imagem, isto é, processar a imagem de segmentos de modo a extrair as regiões de pixels conexos;
(3) **Projeção dos componentes extraídos:** nessa etapa é feita a projeção dos componentes individuais obtidos na etapa anterior;
(4) **Comparação entre as projeções dos objetos da imagem com a projeção do *template*:** a projeção da imagem *template* é comparada com as projeções dos objetos obtidos no processamento da imagem de entrada. Uma medida de similaridade é utilizada como critério de comparação, por meio de um limiar os objetos são classificados como ocorrências ou não do *template*; se a pontuação da medida de similaridade for superior ao limiar uma ocorrência do objeto é reconhecida, caso contrário, o segmento candidato é rejeitado.

Para a implementação dos passos descritos, técnicas de Visão Computacional foram aplicadas. As subseções a seguir descrevem os conceitos a cerca de tais técnicas.

3.1 Limiarização de Imagens

A limiarização ou binarização de uma imagem, segundo Pedrini e Schwartz [3], trata-se de uma das técnicas mais simples de segmentação de objetos e consiste na definição de um ou mais limiares, por meio dos quais é estabelecido um critério para classificar os pixels em uma imagem entre zero ou um. Em outras palavras, a

limiarização consiste na discriminação de pixels de uma imagem, de acordo com seus valores de intensidade, com respeito aos limiares preestabelecidos. Assim, o resultado dessa operação é uma nova imagem na qual os pixels possuem um entre dois valores.

Conceitualmente, os pixels em uma imagem limiarizada são definidos com valores de intensidade zero ou um. Contudo, na prática esta não é uma questão restrita; os valores em uma imagem limiarizada podem ser definidos como 0 e 255 (em imagens de 8-bits) por exemplo, ou quaisquer outros pares de valores. O aspecto relevante é que os pixels serão classificados entre dois valores distintos.

A ideia sobre a qual essa técnica fundamenta-se, parte da premissa de que níveis de cinza pertinentes a um objeto em uma cena, se concentram em um intervalo diferente dos níveis de cinza dos pixels que compõe o fundo da imagem. Portanto, é possível afirmar que o sucesso desta técnica de segmentação, depende da definição de limiares que possam ser utilizados para distinguir pixels em uma imagem de acordo com duas classes; segundo Russ [4], em geral essas duas classes representam os pixels que fazem parte do fundo e os pixels que fazem parte dos objetos presentes na imagem. Jain, Kasturi e Schunk [7] definem o processo de limiarização sobre perspectiva de três casos: (1) com o uso de um limiar fixo L; (2) com o uso de um intervalo fechado $[L_1, L_2]$; e (3) com o uso de um esquema mais geral, no qual diferentes intervalos de limiares são definidos. Neste último caso, um conjunto Z define os valores de intensidade cujos objetos de interesse supostamente irão possuir. Em símbolos, as definições de limiarização de Jain, Kasturi e Schunk, podem ser enunciadas nas seguintes formas:

- Quando um único limiar for utilizado:

$$B(x,y) = \begin{cases} 1 & se\ f(x,y) \leq L \\ 0 & caso\ contrário \end{cases};$$

- Quando os valores de intensidade se situam em um intervalo:

$$B(x,y) = \begin{cases} 1 & se\ L_1 \leq f(x,y) \leq L_2 \\ 0 & caso\ contrário \end{cases};$$

- Quando os valores de intensidade dos objetos de interesse são representados por um conjunto de intervalos de limiares:

$$B(x,y) = \begin{cases} 1 & se\ f(x,y)\ \epsilon\ Z \\ 0 & caso\ contrário \end{cases}.$$

Embora a limiarização seja uma técnica efetiva para algumas situações, em geral, essa técnica requer cenários onde a iluminação é um fator controlado e conhecimento de domínio possa ser integrado à solução, isto é, fatores relacionados às distribuições de níveis de cinza dos objetos e fundo de uma imagem necessitam ser previamente conhecidos, e aproximadamente fixos. Histogramas de níveis de cinza podem ser utilizados para o estudo de tais fatores. É um fato conhecido que a limiarização global (com limiares fixos para todos os pixels da imagem processada) é insuficiente para obter certos resultados quando a iluminação não é uniforme em uma imagem [6][8].

Para superar este problema, existem diversas técnicas que se baseiam no uso de limiares que podem variar em função dos valores locais dos níveis de cinza, ou ainda capazes de estabelecer automaticamente, com base na imagem processada, diferentes limiares capazes de proporcionar melhores resultados no processo de limiarização.

Usualmente, técnicas de limiarização desse gênero são referidas por limiarização adaptativa.

3.2 Extração de Segmentos

O processo de segmentação de uma imagem consiste no particionamento de tal imagem, em sub-regiões ou segmentos. Tais segmentos normalmente representam objetos de interesse presentes na imagem. Assim sendo, após a etapa de segmentação, é necessário realizar a extração das sub-regiões encontradas.

Com o uso de uma função disponibilizada pela biblioteca de código *OpenCV*, denominada $cvFindContours$, os segmentos presentes na imagem limiarizada são extraídos. A função $cvFindContours$ retorna uma sequência com os contornos presentes na imagem; os contornos são representados por estruturas de dados forneceidas pela própria biblioteca e, em particular, uma propriedade importante dessas estruturas é sua região retangular. A região retangular que envolve cada contorno representa uma sub-região da imagem, na qual estão presentes vizinhaças conexas de pixels.

3.3 Histogramas

Histogramas são importantes ferramentas estatísticas para a representação e sumarização de dados. Em seu sentido mais comum, representam distribuições de frequências a respeito de medições ou dados a cerca de algum estudo ou contexto arbitrário. Os histogramas possuem grande relevância, pois fornecem uma estrutura poderosa que viabiliza ou facilita a realização de diversas atividades ou técnicas de análise e processamento.

Conforme Burger e Burge [9], em processamento de imagens, histogramas são utilizados em diversas técnicas de realce e tratamento de qualidades visuais; podem ser utilizados na detecção de problemas de exposição e iluminação não uniforme no processo de aquisição de imagens; ou até mesmo como ferramentas forenses, quando utilizadas para se determinar que tipos de atividades de processamento foram aplicados previamente em uma determinada imagem.

Chang e Krumm [8] utilizam uma variação do histograma de cores – o histograma de co-ocorrência de cores – para o reconhecimento de objetos em imagens. Tal histograma se difere do histograma de cores convencional, por manter uma contagem da ocorrência de determinados pares de cores, espaçados por certas distâncias no espaço da imagem. Em sua abordagem, Chang e Krumm fazem uso de histogramas da co-ocorrência de cores do objeto de interesse, computados de diferentes perspectivas, e então o objeto é localizado nas imagens de entrada a partir da comparação com os histogramas do modelo.

Não obstante, de acordo com Bradski e Kaehler [6], histogramas também podem ser utilizados para representação da distribuição de cores de um objeto; podem ser utilizados como descritores sobre a forma de um objeto, por meio do histograma da orientação dos gradientes de suas arestas; e ainda, como a representação da função de distribuição de probabilidade, com respeito à hipótese sobre a localização de um objeto em uma imagem.

Formalmente, um histograma pode ser visto como uma função discreta $h(i) = n_i$, sendo i um inteiro em um intervalo fechado $[0, K-1]$; K é o número de intervalos ou classes sobre os quais os dados são distribuídos; e o valor n_i corresponde a

frequência ou a contagem de itens pertinentes a coleção de dados subjacentes, associada ao i-ésimo intervalo ou i-ésima classe.

O significado atribuído às frequências e intervalos de um histograma depende do contexto de sua aplicação. Por exemplo, no caso do histograma dos níveis de cinza de uma imagem monocromática, os possíveis níveis de cinza podem ser divididos em K intervalos de igual tamanho; e assim sendo, cada n_i representa a contagem de pixels presentes na imagem, com um valor de intensidade de cinza dentro dos limites do i-ésimo intervalo.

Uma operação importante é a comparação de histogramas, uma vez que histogramas podem ser utilizados como forma de sumarizar ou organizar informações sobre características ou atributos de segmentos ou objetos em imagens. Existem vários métodos para a comparação de histogramas. Bradski e Kaehle citam alguns deles: correlação; chi-quadrado; interseção; e Bhattacharyya.

3.4 Projeções de Imagens

Conforme descrevem Pedrini e Schwartz [3], as projeções horizontais e verticais de uma imagem binária consistem, respectivamente, na soma dos pixels em cada linha da imagem (projeção horizontal); e na soma dos pixels em cada coluna (projeção vertical). De acordo com esta definição, é possível observar que histogramas podem ser adotados como estruturas convenientes para o cálculo de tais projeções. Também é importante salientar que uma mesma projeção pode resultar de diferentes objetos ou imagens.

Em dos Santos *et al.* [5] projeções verticais e horizontais são utilizadas em um método para a extração de texto de páginas de documentos, para fins de aplicações em sistemas de reconhecimento de documentos. No referido trabalho, páginas de documentos são submetidas a um processo de limiarização, e então um conjunto de operações, envolvendo o processamento dos histogramas das projeções verticais e horizontais, são utilizadas em um algoritmo para a extração automática de segmentos de linhas de texto.

4. Aplicativos de Visão Computacional para a Plataforma *RoboDeck*

Dois aplicativos para o reconhecimento de formas e objetos foram desenvolvidos; uma aplicação baseada no *template matching* por meio do cálculo de correlação; e uma segunda aplicação baseada na abordagem de histrogramas descrita anteriormente. O intuito ao implementar uma abordagem tradicional, tal como o *template matching* baseado na correlação, foi o de proporcionar uma base para comparações. As aplicações foram implementadas na linguagem de programação *C++*, com a utilização da biblioteca de código *OpenCV*.

A *OpenCV* trata-se de uma biblioteca de código aberto, destinada a aplicações de visão computacional. É escrita em C e C++ e oferece suporte multiplataforma, ou seja, é compatível com várias plataformas de desenvolvimento, tais como Linux, Microsoft Windows e Mac OS X [8]. Essa bibliotéca de código fornece a implementação otimizada de muitos algoritmos e rotinas de uso comum em aplicações de visão computacional. O desenvolvimento dos aplicativos foi realizado no ambiente Windows, utilizando-se uma *webcam* como dispositivo para a aquisição de imagens. Para a realização de testes, uma folha impressa contendo a seta a ser detectada foi exposta

frente à câmera e como resultado, as aplicações exibiam as imagens capturadas, e desenhavam um quadro ao redor da área contendo o objeto quando reconhecido.

Para avaliação do comportamento das aplicações em um ambiente próximo ao de um robô móvel inteligente, foi utilizado uma giga de testes baseada nos módulos da plataforma robótica *RoboDeck*, desenvolvido pela empresa Xbot. A referida giga de testes possuia um módulo embarcado de alto desempenho, equipada com uma *webcam* para a captura de imagens. Como plataforma de desenvolvimento, a giga de testes contou com uma distribuição Linux, o Debian. Uma vez que o código utilizado na implementação das aplicações mencionadas estava em conformidade com o padrão ANSI, e tendo em mente que a *OpenCV* trata-se de uma biblioteca multiplataforma, as aplicações puderam ser facílmente compiladas em ambos os ambientes (Windows e Linux).

4.1 *Correlation Template Matching*

Nessa abordagem um *looping* realiza a aquisição de imagens por meio do dispositivo de captura disponível, e utilizando um *tamplete*, realiza o cálculo do coeficiente CCN do template com relação a cada região possível da imagem. Dado um *template* com dimensões $W \times H$, se a imagem processada possuir dimensões $M \times N$, então existem $(M - W + 1) \times (N - H + 1)$ formas de posicionar o *template* sobre a imagem para o calculo de correlação. Logo uma matriz de tamanho apropriado é utilizada para o armazenamento dos coeficientes de CCN, obtidos após o processamento da imagem com o método já descrito.

Para estimar a posição de uma provável ocorrência do objeto de interesse, uma chamada a uma função da OpenCV, nomeada $cvMinMaxLoc$, é utilizada para encontrar o ponto de máxima local na referida matriz dos coeficientes de CCN. Caso o valor de máxima local for superior a um limiar parametrizado, então a posição da imagem correspondente a esse coeficiente é informada como a posição de uma ocorrência do *template*. Toda a lógica relacionada com o processo de busca do *template* foi encapsulada dentro de uma classe nomeada $TemplateMatch$. Desse modo, para estender a capacidade da aplicação para o reconhecimento de múltiplas formas, provenientes de diferentes templates, é necessário apenas a inclusão de código para instânciar novos objetos da classe *TemplateMatch*, com diferentes imagens *template*.

4.2 Projeção por Histogramas

Essa abordagem se baseia no uso das projeções de objetos extraídos de uma imagem por meio de um processo de segmentação, e de sua comparação com a projeção de uma imagem de referência do objeto a ser reconhecido. Para facilitar o cálculo das projeções e do processo de comparação, histogramas foram utilizados como forma de representação para as projeções. A projeção de um objeto proporciona uma estrutura mais conveniente para fins de processamento de sua forma, isto é, as projeções de um objeto fornecem um meio simplificado para expressar informações sobre características da forma geral de tal objeto. No contexto desse trabalho, as projeções foram utilizadas para facilitar o reconhecimento do objeto.

Nesta segunda implementação, as imagens foram processadas por meio de uma operação de limiarização. Uma vez que as setas impressas eram de cor preta (níveis de cinza próximos de zero), definir limiares que pudessem realizar uma boa segmentação (para discriminar a seta do fundo em que se encontra) não foi uma tarefa complicada. Com o uso de uma função disponibilizada pela *OpenCV* [6], os segmentos presentes na

imagem limiarizada foram extraídos. A partir das regiões retangulares que envolvem cada segmento, o histograma de sua projeção vertical foi calculado. Regiões com áreas inferiores a um limiar de tamanho foram descartadas, evitando o processamento desnecessário de regiões muito pequenas para serem candidatas a uma ocorrência do objeto. Na projeção das imagens, histogramas foram instânciados para o armazenamento dos valores das projeções.

O processo de projeção foi feito dividindo a imagem em faixas horizontais ou verticais (para as projeções horizontais e verticais respectivamente), uma vez que a imagem foi limiarizada e seus segmentos poderam ser vistos como regiões de pixels binários, o número de pixels não nulos (isto é, dos pixels que foram discriminados como parte de um objeto) em cada faixa resultou no histograma de projeção. O número de intervalos ou faixas utilizados na projeção do histograma foi um fator importante. Na implementação em questão, esse valor foi um dado parametrizado e precisou ser escolhido com cautela. Se uma região for dividia em faixas muito largas, a capacidade expressiva do histograma resultante, como meio para a representação da forma presente em tal região, é comprometida; grande parte dos aspectos característicos da forma é perdida. De modo análogo, utilizar faixas muito estreitas pode tornar a projeção muito característica de um exemplar específico da forma do objeto, comprometendo a capacidade de generalização do modelo.

Após gerar os histogramas das formas extraídas da imagem segmentada e do *template* do objeto de interesse, a busca pelo objeto se resumiu na comparação dos histogramas computados com o histograma do *template*. A localização do objeto detectado foi estabelecida como sendo a região retangular envolvendo o segmento reconhecido como ocorrência do objeto. Qualquer um dos métodos de comparação de histogramas citados na sessão 3.3 podem ser utilizadas na etapa de comparação dos histogramas. O critério para rejeitar ou reconhecer um dado segmento como uma possível ocorrência do objeto de interesse, deve ser estabelecido de acordo com a interpretação de cada medida de comparação.

De acordo com Bradski e Kaehler, a comparação pelo critério de interseção para a comparação de histogramas é melhor em aplicações onde a precisão não é um fator tão importante quanto o desempenho; ao passo que os critérios chi-quadrado e Bhattacharyya proporcionam medidas mais acuradas, porém ao custo de um desempenho inferior.

4.3 Teste Comparativo

Para avaliar o desempenho das aplicações implementadas, foi utilizado uma amostra com quarenta imagens digitais com resolução de 640×480; dez imagens exibindo um retângulo; dez imagens exibindo uma seta indicando para a direita; dez imagens exibindo uma seta indicando para esquerda; e dez imagens sem nenhuma seta. O objetivo desse teste foi avaliar a precisão ao detectar o objeto correto, a capacidade entre distinguir uma seta indicando para a direita e para a esquerda, e quanto à robustez do algoritmo com respeito a falsos positivos, isto é, indicar a presença do objeto de interesse em imagens nas quais o objeto não estava presente.

Nos testes realizados, foram utilizadas imagens de um retângulo com o intuito de verificar se as aplicações confundiriam os sinais de interesse com outros sinais semelhantes; testes com imagens sem a presença de uma seta foram utilizadas para avaliar taxas de falsos positivos; e imagens com setas indicando para esquerda e direita para testar a capacidade de distinguir entre ambos os sinais. O tempo de processamento

para o reconhecimento de cada imagem individual também foi monitorado. Uma vez que aplicações envolvendo a detecção de objetos para oferecer autonomia para robôs possuem o requisito de processamento em tempo real, o propósito dessa medida foi o de fornecer uma estimativa do tempo de processamento médio para cada aplicação, no processamento de uma imagem.

A Tabela 1 ilustra os resultados de teste para a aplicação baseada no *template matching*; e a Tabela 2 ilustra os resultados para a aplicação baseada no uso de histogramas. A Figura 2 exibe um gráfico comparativo do desempenho de ambas as aplicações. Para cada conjunto com dez imagens, as respostas corretas obtidas pelas aplicações, em cada situação, foram sumarizadas e seu percentual de acerto é ilustrado no gráfico.

Tabela 1 – Resultados do teste da aplicação baseada no *template matching*.

Teste	**Quantidade**	
	Seta para direita	**Seta para esquerda**
Imagens com um retângulo	0	0
Imagens com uma seta para direita	8	0
Imagens com uma seta para esquerda	0	6
Imagens sem a presença de setas	0	0
Média do tempo de execução por imagem (milissegundos)	294,7 milissegundos	

Tabela 2 – Resultados do teste da aplicação baseada na projeção por histogramas.

Teste	**Quantidade**	
	Seta para direita	**Seta para esquerda**
Imagens com um retângulo	0	2
Imagens com uma seta para direita	4	4
Imagens com uma seta para esquerda	0	4
Imagens sem a presença de setas	1	0
Média do tempo de execução por imagem (segundos)	21 milissegundos	

Para as imagens com uma seta para direita ou esquerda, a resposta correta é quando a aplicação identifica uma seta para direita e para esquerda respectivamente; nas imagens com um retângulo, a resposta correta é quando o algoritmo não identifica nenhuma seta; e para imagens sem a presença de setas, a resposta correta é quando a aplicação não identifica nenhuma seta.

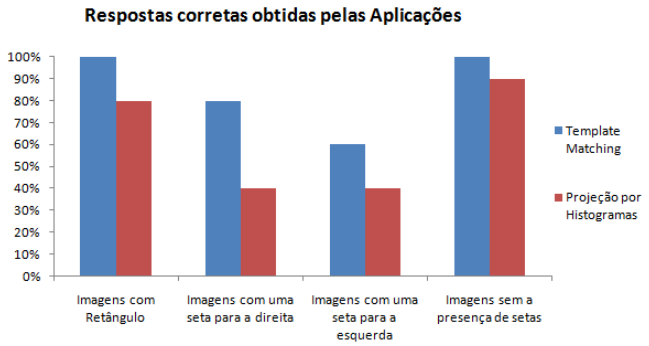

Figura 5 – Gráfico comparativo do desempenho das aplicações.

5. Comentários Finais

O emprego de técnicas de Visão Computacional para a robótica móvel inteligente vem crescendo muito nos últimos anos. Este artigo apresentou resultados preliminares da utilização de duas técnicas de Visão Computacional para a identificação de sinais em uma plataforma de robótica móvel, que foram a correlação por *template matching* e a projeção por histogramas.

A plataforma robótica adotada foi o *RoboDeck* da empresa XBot. Os resultados apontaram para um desempenho melhor, em termos do tempo de resposta, para a técnica de projeção por histograma. Embora a precisão do reconhecimento de sinais tenha sido inferior ao da técnica *template matching*. Em sistemas robóticos, que precisam responder ao estímulo do ambiente em tempo real, o tempo de resposta é um fator crítico. Portanto, acreditamos que a técnica de projeção por histogramas seja adequada para o desenvolvimento dos aplicativos robóticos propostos, justamente em função do tempo de resposta significativamente menor (quando comparado ao *template matching*).

Como trabalhos futuros, serão revisados os algoritmos baseados em histogramas, para melhorar a precisão no reconhecimento dos sinais. Outros aplicativos robóticos também estão em fase de testes, envolvendo rastreamento e perseguição de objetos em movimento.

Referências Bibliográficas

[1] XBot. *RoboDeck*: manual do usuário. Versão 1.0, 2010.

[2] MUNÕZ, M. E. de S. Relatório Técnico: Projeto do Software do *RoboDeck*, versão 0.2, Cientistas & Associados, 2009.

[3] PEDRINI, H.; SCHWARTZ, W. R.. Análise de Imagens Digitais – Princípios, Algoritmos e Aplicações. São Paulo: Thomson Learning, 2008.

[4] RUSS, J. C.. The Image Processing Handbook. 6th Edition, New York: CRC Press, 2011.

[5] DOS SANTOS, R. P.; CLEMENTE, G. S.; REN, T. I.; CAVALCANTI, G. D. C.. Text Line Segmentation Based on Morphology and Histogram Projection. In:

International Conference on Document Analysis and Recognition - ICDAR, pp. 651-655, 2009.

[6] BRADSKI, G.; KAEHLER, A.. Learning OpenCV: Computer Vision with OpenCV Library. California: O'Reilly Media, Inc., 2008.

[7] JAIN, R.; KASTURI, R.; SCHUNK, B. G.. Machine Vision. New York: McGraw-Hill, 1995.

[8] CHANG, P.; KRUMM, J.. Object Recognition with Color Concurrence Histograms. In: Computer Vision and Pattern Recognition - CVPR , pp. 2498-2504, 1999.

[9] BURGER, W.; BURGE, M. J.. Principles of Digital Image Processing: Fundamental Techniques. London: Springer, 2009.