mαcrϕ@ufmg

Universidade Federal de Minas Gerais
Curso de Graduação em Engenharia de Controle e Automação
Grupo de pesquisa MACRO - Mecatrônica, Controle e Robótica

IMPLEMENTAÇÃO DE UM ALGORITMO DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE TOMADAS ELÉTRICAS

Frederico Fernandes Afonso Silva

Belo Horizonte, Brasil
2015

Frederico Fernandes Afonso Silva

IMPLEMENTAÇÃO DE UM ALGORITMO DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE TOMADAS ELÉTRICAS

Monografia submetida à banca examinadora designada pelo Colegiado Didático do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Minas Gerais, como parte dos requisitos para aprovação na disciplina Projeto Final de Curso II.

Orientador: Prof. Bruno Vilhena Adorno, PhD

Belo Horizonte, Brasil
2015

Agradecimentos

Agradeço aos meus pais, pelo incentivo e apoio constantes. São eles os verdadeiros responsáveis pela minha educação.

Agradeço aos meus amigos, que fizeram essa caminhada de cinco anos muito mais agradável. Em especial à Gilvâni, cujo apoio foi de suma importância em alguns aspectos de implementação de código. Espero que possamos manter contato, mesmo que os caminhos tendam a se separar após a formatura.

Agradeço à Universidade Federal de Minas Gerais e aos professores que contribuíram para a minha formação como engenheiro, em especial ao professor Bruno Vilhena Adorno, cuja orientação e supervisão auxiliaram a realização desse projeto e nortearam o meu desenvolvimento profissional.

Resumo

Este projeto tem como objetivo o desenvolvimento de um sistema de visão computacional que possibilite a detecção e reconhecimento de tomadas elétricas em meio a um ambiente desconhecido e não dotado de artifícios, como marcações especiais ou conhecimento prévio do posicionamento das tomadas.

O sistema faz uso de uma câmera RGB-D (*Microsoft Kinect*) para a interação com o ambiente e utiliza o descritor conhecido como Histograma de Gradientes Orientados para realizar o treinamento de uma Máquina de Vetores de Suporte. Esse procedimento permite o reconhecimento da tomada em imagens RGB, que é então associado à sua posição em uma nuvem de pontos. Isso possibilita a obtenção da pose completa da tomada no espaço tridimensional.

O sistema alcançou resultados satisfatórios, possibilitando o reconhecimento de tomadas elétricas, mesmo que essas estejam parcialmente oclusas.

Abstract

The main objective of this project is the development of a machine vision system that allows the detection and recognition of an electrical outlet in an unknown environment, without any special marks or previous knowledge of the electrical outlet's position.

The system uses a RGB-D sensor (Microsoft Kinect) to interact with the environment and uses the descriptor Histogram of Oriented Gradients for training a Support Vector Machine. This procedure allows the recognition of the electrical outlet in the RGB image, which is then associated to its equivalent position in the cloud point. Thereby, the complete pose of the electrical outlet is obtained in the tridimensional space.

The system achieved satisfactory results, enabling the recognition of electrical outlets, even if they suffer partial occlusion and total or partial rotation.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviações

BRF Base Radial Function. 15

HOG Histogram of Oriented Gradients. 7, 8, 11, 18, 19, 20, 24, 27, 28

MACRO Mechatronics, Control, and Robotics. 3

MARIA Manipulator Robot for Interaction and Assistance. 7, 19, 31

OpenCV Open Source Computer Vision. 20, 31

OpenNI Open Natural Interaction. 23

PCL Point Cloud Library. 23, 24, 31

ROI Region Of Interest. 24, 28, 29, 30

ROS Robot Operating System. 19, 23, 24, 27, 30

SIFT Scale Invariant Feature Transform. 7

SURF Speeded Up Robust Features. 7, 8

SVM Support Vector Machine. vi, 8, 11, 13, 14, 18, 19, 20, 27, 28, 33

Capítulo 1

Introdução

A robótica tem como objetivos o estudo do projeto, construção e operação de robôs. Essa área de pesquisa em constante desenvolvimento e os ambientes industriais com crescente competitividade exigem que os robôs sejam capazes de interagir com o mundo de maneira cada vez mais dinâmica, independente e precisa. Paralelamente ao crescimento da gama de aplicações industriais, os robôs vêm também ganhando espaço no ambiente doméstico.

Aplicações residenciais exigem dos robôs um paradigma diferente do que ocorre no meio industrial. Enquanto na indústria tem-se uma preocupação maior com robustez e precisão, os ambientes domésticos possuem padrões de segurança mais rígidos devido ao inerente contato com seres humanos. Para atuar em tais ambientes, os robôs necessitam de sistemas auxiliares que os possibilitem atender a esses requisitos. Um sistema com crescente utilização nesse contexto é o sistema de visão computacional.

A visão computacional consiste na ciência e tecnologia que se destina a prover aos sistemas artificiais a capacidade de obter informações por meio de imagens. Possui como importantes aplicações a inspeção automática, o controle de processos, a navegação de veículos autônomos e os sistemas de vigilância automática.

Na robótica a visão computacional aplica-se como uma ferramenta de auxílio que possibilita ao robô a identificação e a localização de objetos de interesse. Especificamente, na área da robótica de assistência, essa característica é de extrema importância, devido à complexidade e vasta variedade de objetos presentes em ambientes tipicamente humanos. Outra característica de suma importância nessa área consiste na autonomia dos robôs envolvidos no processo de assistência.

Uma das grandes limitações à autonomia dos robôs está no fator energético. Mesmo a bateria mais durável eventualmente precisará ser recarregada. Para lidar com essa situação, atualmente existem duas soluções mais utilizadas: recarga manual e recarga utilizando as chamadas "docking stations".

A recarga manual apresenta como principal desvantagem o fato de não ser automática, exigindo uma pessoa encarregada do processo de recarga. Já as *docking stations* (Figura 1.1) necessitam ser posicionadas em um ambiente livre de obstáculos e, se porventura forem obstruídas, o robô será incapaz de utilizá-las, necessitando igualmente da intervenção humana.

Sendo assim, uma alternativa que vem sendo alvo de pesquisas é o desenvolvimento de um robô capaz de se conectar de maneira independente em tomadas elétricas (May-

ton et al., 2010; Meeussen et al., 2010). Essa abordagem, se realizada de maneira a não necessitar de intervenções no ambiente convencional, é capaz de prover autonomia completa ao robô, pelo menos no que diz respeito ao ponto de vista energético.

Uma maneira de permitir ao robô a exploração do ambiente e detecção de tomadas é dotar o mesmo de um sistema de visão computacional. Tal sistema será o alvo de discussão deste trabalho.

Figura 1.1: Robô modelo UBR-1 e sua *Docking Station*, cortesia da Unbounded Robotics.

1.1 Objetivos do Projeto

O projeto, desenvolvido no Laboratório de Manipuladores Robóticos, da Universidade Federal de Minas Gerais (UFMG), faz parte dos trabalhos desenvolvidos pelo grupo de pesquisas MACRO (Mechatronics, Control, and Robotics) e propõe um sistema de visão computacional que possibilite ao robô a detecção e reconhecimento de tomadas elétricas, em meio a um ambiente desconhecido e não dotado de artifícios, como marcações especiais ou conhecimento prévio do posicionamento das tomadas.

1.2 Estrutura da Monografia

O trabalho é dividido em sete capítulos: o Capítulo 2 apresenta algumas pesquisas relacionadas, bem como uma breve discussão das capacidades e limitações da área até o momento; o Capítulo 3 abrange todos os conceitos teóricos necessários para um melhor entendimento do projeto; o Capítulo 4 descreve a arquitetura do sistema; o Capítulo 5 apresenta os resultados do trabalho; o Capítulo 6 apresenta uma discussão sobre o sistema desenvolvido e conclui a monografia.

Capítulo 2

Revisão Bibliográfica

No que diz respeito ao problema de garantir autonomia energética aos robôs, pode-se dividir as soluções propostas em duas categorias: aquelas que envolvem a modificação do ambiente e aquelas que visam manter o ambiente inalterado. Tais abordagens serão discutidas na primeira seção deste capítulo.

A segunda seção deste capítulo irá tratar dos progressos no campo da visão computacional, sejam eles decorrentes do desenvolvimento de novos algoritmos e paradigmas ou à incorporação de novos *hardwares*.

Por fim, a terceira e última sessão desse capítulo apresenta uma breve revisão do assunto discutido, com enfoque nas técnicas de maior pertinência a esse projeto.

2.1 Autonomia Energética

A ideia de um robô capaz de se auto recarregar foi inicialmente demonstrada pelo neurofisiologista e roboticista W. Grey Walter em 1950, por meio de seus robôs analógicos Elmer e Elsie (Figura 2.1) (W. Walter, 1950). Tais dispositivos eram dotados de dois tubos de vácuo, um sensor de toque e um sensor de luz. Diversas situações de teste foram propostas, mas de maneira geral os dispositivos eram atraídos para a luz e essa atração se tornava maior quando as suas baterias estavam se esgotando, devido a um aumento do ganho do fotodetector. A luz indicava aos robôs o local onde eles poderiam se recarregar.

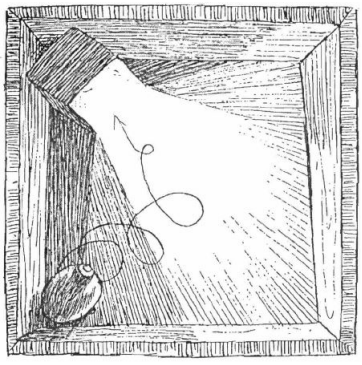


Figura 2.1: Robô Elmer (W. Walter, 1950).

Muito provavelmente advindas dessa ideia surgiram as hoje largamente difundidas estações de recarga (Figura 1.1). Tais estações consistem de um dispositivo específico que determina ao robô a região à qual ele deve se dirigir quando sua bateria estiver se esgotando e uma recarga seja portanto necessária. Diversas pesquisas foram feitas abordando essa solução.

Uma demonstração da capacidade de fornecer aos robôs autonomia energética por meio do uso de estações de recarga foi feita por Silverman et al. (2003). Em sua pesquisa os autores utilizaram uma versão modificada do robô ActivMedia Pioneer 2DX (Figura 2.2) para realizar o monitoramento da porta de seu laboratório. Quando a bateria atingisse determinado nível de descarga, o robô deveria procurar pela estação de recarga e, uma vez recarregado, retornar ao seu posto. Os resultados do trabalho comprovaram a eficiência das *docking stations* como uma alternativa que possibilite aos robôs a realização de tarefas por longos períodos de tempo sem a necessidade da intervenção humana.

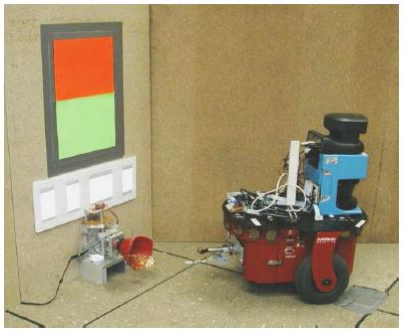


Figura 2.2: ActivMedia Pioneer 2DX (Silverman et al., 2003).

Uma modificação às estações de recarga foi proposta por Kartoun et al. (2006). A pesquisa utiliza uma versão adaptada do robô móvel ER-1 Evolution Robotics e propõe uma modificação que torna possível a recarga simultânea do robô e do laptop utilizado pelo mesmo. Para localização da estação e confirmação de uma conexão bem sucedida o sistema conta com uma câmera colorida e um par de LEDs infravermelhos. Outras duas propostas de modificações de estações de recarga são apresentadas por Roh et al. (2008).

Conforme demonstrado por essas e outras várias pesquisas, o uso de estações de recarga, bem como de suas modificações, possui robustez e eficiência comprovadas. Devido a isso, atualmente diversas aplicações comerciais fazem uso de estações de recarga. Destacam-se nessa área os modelos domésticos Roomba, da IRobot (Jones, 2006), e o AIBO, da Sony (Fujita, 2001). Entretanto, essa abordagem apresenta a inerente desvantagem de ser limitada a ambientes não obstruídos, uma vez que caso a estação de recarga seja bloqueada o robô será incapaz de manter sua autonomia energética. Visando contornar esse problema, uma segunda linha de pesquisas se desenvolveu, propondo a utilização de tomadas elétricas convencionais na recarga dos robôs.

Em 1964 um robô móvel analógico (Figura 2.3) foi desenvolvido pelo APL *Adaptative Machines Group* e posto a percorrer os corredores da *Johns Hopkins University* e se conectar em tomadas elétricas convencionais (Watson & Scheidt, 2005). O disposi-

tivo desenvolvido era extremamente especializado e atendia tão somente a tarefa de se conectar em tomadas.

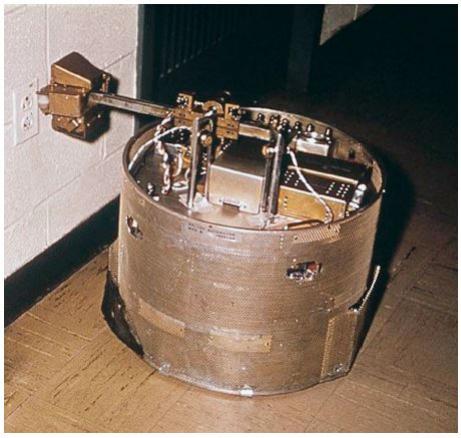


Figura 2.3: Robô analógico Beast (Watson & Scheidt, 2005).

No campo dos robôs de propósito geral, dois trabalhos recentes destacam-se. Uma pesquisa utilizando o modelo *MARVIN Mobile Manipulation Platform* foi realizada por Mayton et al. (2010). O objetivo dos desenvolvedores do projeto foi a criação de um robô capaz de se conectar em tomadas elétricas não modificadas, guiando-se através do campo elétrico emitido pela rede de 60 Hz. Apesar da proposta se apresentar completamente genérica a princípio, é previamente fornecido ao robô um mapa bidimensional do ambiente, contendo as posições e alturas estimadas das tomadas. Além do mais, apesar de não requerer alterações no ambiente, o projeto parte do princípio de que todas as tomadas na residência seguem o *National Electrical Code* (NEC) e atendem ao padrão WD6-2002, da *National Electrical Manufacturer's Association* (NEMA), com dimensões estritamente definidas. Atendendo à essas restrições, o dispositivo foi posto a testes e obteve uma taxa de sucesso de 93% (28 acertos em 30).

Em 2010 foi desenvolvido, pela *start-up* de robótica *Willow Garage*, um sistema robótico capaz de navegar pelo ambiente de um escritório, abrir portas e se conectar em tomadas elétricas (Meeussen et al., 2010). Utilizou-se um protótipo do modelo *PR2 Mobile Manipulation Platform* (Figura 2.4), ao qual foi fornecido um mapa bidimensional do ambiente, contendo anotações manuais das posições aproximadas das tomadas e portas, bem como informações sobre de que lado encontram-se as maçanetas e qual o sentido de abertura das portas. O projeto não requer modificações ao ambiente, mas parte do princípio de que o mesmo já encontra-se de acordo com as especificações dos códigos de construção norte americanos, em especial o *Americans with Disabilities Act* (ADA), que estabelece restrições ao tamanho mínimo das portas e ao tipo e posicionamento das maçanetas. O sistema de detecção de tomadas também foi posto à prova apenas para um caso bem específico, de tomadas 2x2 laranja e brancas (Figura 2.4), pois alegadamente essas já estavam presentes em todo o prédio. A taxa de sucesso do processo de conexão em tomadas foi de 95% (18 acertos em 19). Os pesquisadores afirmam ter modificado posteriormente o sistema de detecção para torná-lo mais genérico e capaz de identificar tomadas de outras cores, apesar do artigo não apresentar testes para essas situações.

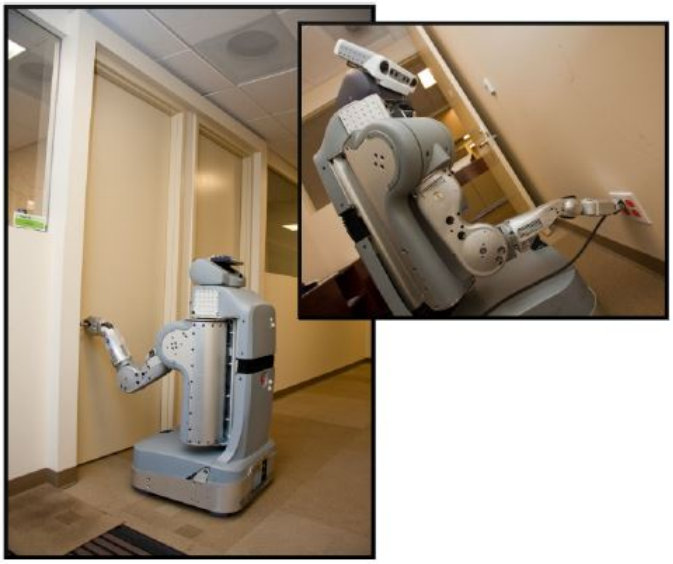


Figura 2.4: Robô PR2 (Meeussen et al., 2010).

Analisando-se as pesquisas que vêm sendo realizadas, fica evidente portanto a carência do desenvolvimento de um sistema que possibilite a detecção de tomadas elétricas de maneira inteiramente genérica, sem necessitar do conhecimento prévio de posicionamento ou marcações especiais e contrastes de cores. Advinda dessa necessidade, dá-se a justificativa desse projeto.

2.2 Visão Computacional

O projeto visa ser implementado no manipulador robótico móvel denominado MARIA, sigla para *Manipulator Robot for Interaction and Assistance*, apresentado na Figura 2.5. Como o dispositivo em questão já é dotado de uma câmera RGB e de um *Kinect* (Han et al., 2013), optou-se pelo desenvolvimento de um sistema de visão computacional para a detecção de tomadas.

O processo básico de identificação de padrões e objetos consiste na definição de "pontos de interesse"em seções distintas da imagem, como bordas, quinas e estruturas mínimas significativas. A esses pontos dá-se o nome de detectores, e para cada um deles a sua vizinhança é representada por um vetor de características. Esse vetor recebe o nome de descritor e é utilizado na pesquisa em outras imagens. Diversas propostas de diferentes estruturas de detectores e descritores foram feitas ao longo dos anos. Dentre esses destacam-se o SIFT, o SURF e o HOG que serão discutidos a seguir.

O descritor conhecido como SIFT (Figura 2.6), *Scale Invariant Feature Transform*, define a criação de um vetor de características invariante a mudanças de escala e rotações na imagem, e parcialmente invariante a mudanças de iluminação e perspectivas (Lowe, 2004). A rapidez alcançada pelo algoritmo se deve a uma redução do custo de extração dos vetores de características através da aplicação de uma filtragem em cascata, onde as operações de maior custo computacional são aplicadas apenas às localizações que passaram nos testes iniciais.

Figura 2.5: MARIA - *Manipulator Robot for Interaction and Assistance*, robô utilizado no grupo de pesquisa MACRO.

A construção dos descritores das características da imagem ocorre através da aplicação de um algoritmo de Diferença de Gaussianas que determina a localização de pontos-chave, que são invariantes a mudanças de escala e orientação. Aos pontos mais estáveis são atribuídas uma ou mais orientações. O descritor dos pontos-chave é obtido através da medição dos gradientes da imagem nas vizinhanças de cada ponto-chave. Esses são então transformados em uma representação de significativa invariância à mudanças de iluminação e distorção na forma.

O descritor conhecido como SURF, *Speeded Up Robust Features*, foi proposto por Bay et al. (2008). Os criadores do método tiveram como foco o desenvolvimento de detectores e descritores invariantes a mudanças de escala e rotação da imagem. Distorções, variações de escala não isotrópicas e efeitos de perspectiva foram assumidos como efeitos secundários que são cobertos em algum grau pela robustez dos descritores.

O detector proposto baseia-se em uma aproximação da matriz Hessiana, denominada pelos autores de *Fast-Hessian*, que utiliza imagens integrais para redução do custo computacional. Por sua vez, o descritor baseia-se na distribuição das respostas da Transformada de Haar ao redor dos pontos de interesse. Os autores propõem também um novo passo de indexação baseado no sinal do Laplaciano que aumenta a velocidade de identificação e a robustez do descritor.

O descritor de características denominado HOG (Figura 2.7), *Histogram of Orien-*

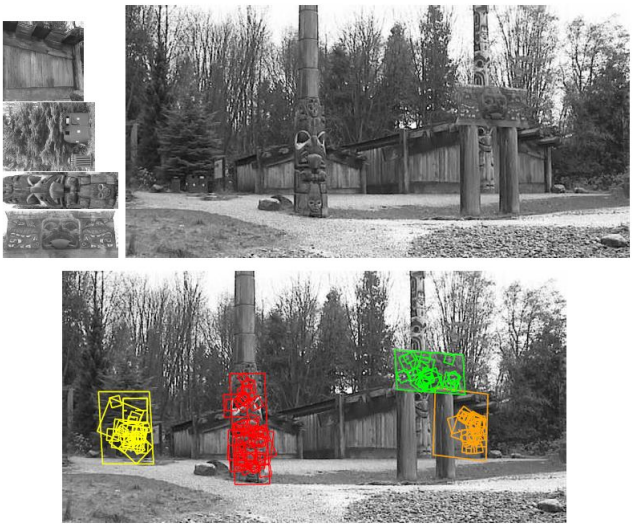


Figura 2.6: Detecção de objetos utilizando-se SIFT (Lowe, 2004).

ted Gradients, foi proposto por Dalal & Triggs (2005). O procedimento consiste da varredura da imagem com uma janela de tamanho fixo, que é subdividida em uma rede uniforme de células. Para cada célula se calcula o gradiente de orientação sobre os pixels e se armazena essa informação em um histograma unidimensional. A ideia geral é a de que se seja capaz de determinar informações sobre a aparência e forma do objeto a partir de informações da distribuição local dos gradientes. Os grupos de células adjacentes são denominados blocos e são utilizados para normalizar o contraste localmente. O conjunto de todos os blocos de histograma concatenados dá origem ao descritor. Esse é então utilizado no treinamento de um SVM (*Support Vector Machine*)[1] linear.

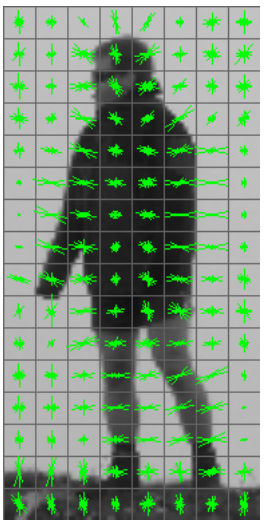


Figura 2.7: Descritor HOG.[2]

[1]Maires detalhes serão dados no Capítulo 3.2, pág.13
[2]Imagem retirada de: http://www.juergenwiki.de/work/wiki/doku.php

2.3 Conclusão Parcial

A autonomia energética é um fator chave para garantir aos robôs a capacidade de executar tarefas por longos períodos sem a intervenção humana. Apesar da solução comercial atual, estações de recarga, ser bastante eficiente para ambientes desobstruídos, o mesmo não pode ser garantido em meio à presença de obstáculos. Dotar os robôs da habilidade de se conectar em tomadas elétricas convencionais, sem a necessidade de alterações no ambiente, é portanto desejável.

Uma maneira eficiente de prover tal habilidade aos robôs se dá por meio dos sistemas de visão computacional. O uso de algoritmos adaptativos vem se provando promissor nesse setor. O desenvolvimento desse sistema será o alvo de estudo desse projeto.

O capítulo seguinte irá apresentar uma revisão teórica dos principais conhecimentos necessários para o entendimento da arquitetura do projeto.

Capítulo 3

Fundamentação Teórica

As seções desse capítulo tratam de definições e conceitos teóricos cujo entendimento é vital para a compreensão da arquitetura do sistema proposto. A primeira seção trata do Histograma de Gradientes Orientados. A seção seguinte descreve o funcionamento de uma Máquina de Vetores de Suporte (do inglês, *Support Vector Machine* - SVM).

3.1 Histograma de Gradientes Orientados

Uma visão geral da técnica de detecção de objetos utilizando-se o método de Histograma de Gradientes Orientados é apresentada na Figura 3.1. Esse método se baseia em uma avaliação de histogramas de gradientes orientados, previamente normalizados, em uma grade densa. A ideia básica por trás do processo é a de que objetos podem, normalmente, ser bem caracterizados pela intensidade local dos gradientes, mesmo na ausência de conhecimento preciso sobre o gradiente correspondente.

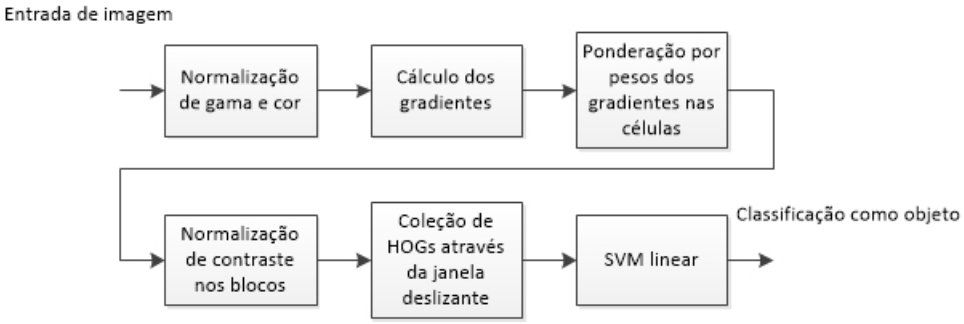


Figura 3.1: Arquitetura do sistema de detecção de objetos por HOG (imagem modificada de (Dalal & Triggs, 2005)).

O HOG é implementado dividindo-se a imagem em pequenas porções espaciais denominadas células. Dalal & Triggs (2005) trabalharam com imagens de 64×128 pixels e células de 8×8 pixels. A Figura 3.2 apresenta uma versão ampliada de uma dessas imagens e uma das células desenhada em vermelho.

[1]Disponível em: http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

Figura 3.2: Exemplo de uma imagem utilizada no treinamento do HOG. No canto direito superior (em vermelho) encontra-se o exemplo de uma célula. [1]

Para cada pixel na célula são calculados os vetores de gradientes, que são dados pela diferença de intensidade luminosa entre cada pixel e seus 4-vizinhos. Os pixels na borda da imagem são definidos como um vetor de norma zero. A Figura 3.3 apresenta um exemplo de cálculo de um desses vetores. Dado o vetor de gradientes $\mathbf{v} = [x\ y]$, tem-se $x = 94 - 56$ e $y = 93 - 55$, resultando no vetor $\mathbf{v} = [38\ 38]$.

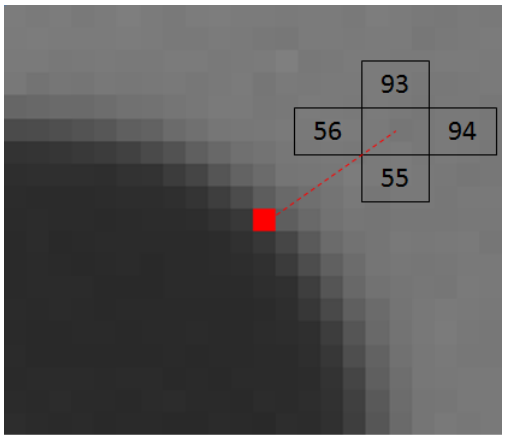


Figura 3.3: Exemplo do cálculo de um vetor de gradientes.[2]

Os vetores de gradientes são então organizados em um histograma de nove classes, conforme suas normas, que é normalizado para garantir maior robustez à variação de contraste na imagem (Figura 3.4).

[2]Disponível em: http://chrisjmccormick.wordpress.com/2013/05/07/gradient-vectors/
[3]Disponível em: http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

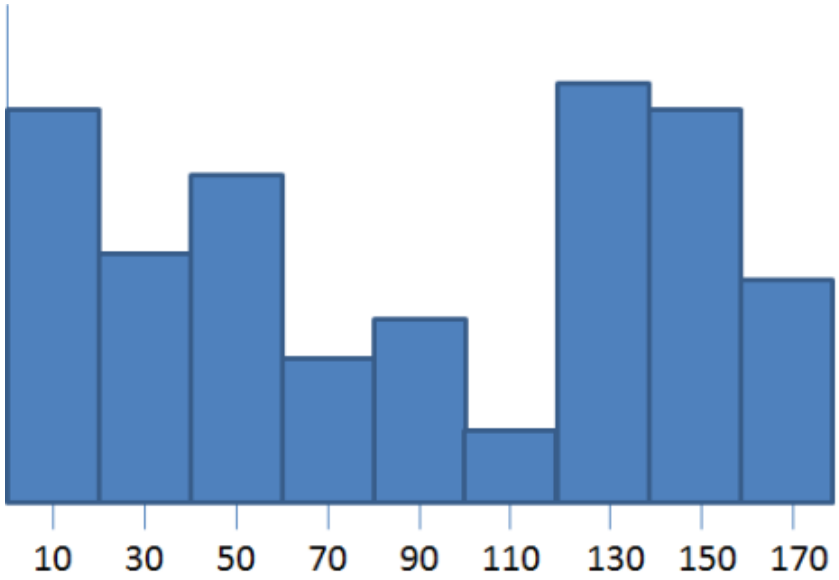


Figura 3.4: Exemplo de histograma normalizado no cálculo do HOG.[3]

As células são agrupadas em blocos de quatro (Figura 3.5), compondo um vetor u de 36 componentes, obtidos pelo agrupamento das nove classes de cada uma das quatro células em um único vetor, e normalizados conforme a equação seguinte:

$$\check{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}. \tag{3.1}$$

Figura 3.5: Exemplo da divisão da imagem em blocos para cálculo do HOG.[4]

A representação global é então formada pela combinação dos histogramas locais, possuindo 3780 valores (7 blocos horizontais ×15 blocos verticais ×4 células por bloco ×9 classes por histograma). Os descritores obtidos são então utilizados para o treinamento de um SVM linear.

3.2 Máquina de Vetores de Suporte

As Máquinas de Vetores de Suporte, do inglês *Support Vector Machines* (SVM) (Cortes & Vapnik, 1995), são parte de uma família de algoritmos de aprendizado supervisionado

[4]Disponível em: http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

chamados métodos baseados em *kernel*. Nesse tipo de algoritmo as previsões são feitas com base em combinações lineares da função *kernel* avaliadas nos dados de treinamento. Apesar de originalmente desenvolvidos para reconhecimento visual de letras, o SVM se popularizou em aplicações de classificação e regressão. O SVM é, em essência, um classificador binário, sendo capaz de realizar a separação apenas entre duas classes de dados. Extensões do SVM permitem a resolução de problemas de classificação multi-classes.

A determinação dos parâmetros do SVM é equivalente à resolução de um problema de otimização convexo, dessa forma a solução obtida é um ótimo global. A forma mais simples de um SVM, para um grupo de dados de treinamento $x_i \in \mathbb{R}^n$ e um conjunto de classes binárias $c_{1,2}$, será explicada a seguir. Seja a saída desejada y_i dada por

$$y_i = \begin{cases} 1 & \text{para } n_i \in c_1, \\ -1 & \text{para } n_i \in c_2, \end{cases}$$

então as duas classes c_1 e c_2 são linearmente separáveis pelo hiperplano (Figura 3.6) descrito por

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{3.2}$$

onde $\mathbf{x}$ é o vetor contendo os dados de treinamento, $\mathbf{w}$ é a normal ao hiperplano, b é um parâmetro de viés e $-b/\left\|w\right\|$ é a distância perpendicular do hiperplano à origem. Isso significa que existe pelo menos uma escolha de $\mathbf{w}$ e b tal que:

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b > 0 & \text{para } y_i = 1, \\ \mathbf{w}^T\mathbf{x}_i + b < 0 & \text{para } y_i = -1. \end{cases}$$

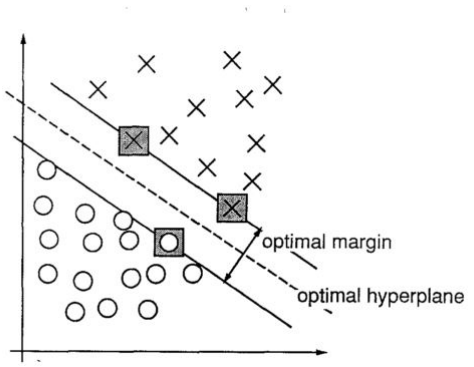


Figura 3.6: Separação em um espaço bidimensional. Os SVM, marcados com quadrados cinzas, definem as margens da máxima separação entre as classes (Cortes & Vapnik, 1995).

Portanto, a função de decisão se torna $F = sign(\mathbf{w}^T\mathbf{x} + b)$ com $\mathbf{x}$ como dados de teste. O SVM procura encontrar as variáveis $\mathbf{w}$ e b tais que a menor distância do hiperplano para qualquer uma das amostras (chamada de margem) seja maximizada. Cortes & Vapnik (1995) provaram que esse problema pode ser escrito formalmente como o problema de otimização convexa dado por

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \text{ sujeito a } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1. \tag{3.3}$$

Esse problema de otimização é mais facilmente resolvido em uma formulação Lagrangiana (Burges, 1998), uma vez que as restrições são mais fáceis de serem lidadas e os dados de treinamento irão aparecer na forma de produtos internos entre os vetores. Introduzindo-se os multiplicadores de Lagrange $\mathbf{x}_i \geq 0$, a função Lagrangiana obtida é dada por

$$L_p = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i y_i(\mathbf{w}^T\mathbf{x}_i + b) + \sum_i \alpha_i, \tag{3.4}$$

de onde obtém-se

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j). \tag{3.5}$$

L_D é conhecida como a forma dual da primária L_p, e suas restrições são diferentes de L_p. O termo $k(x_i, x_j)$ é uma função *kernel* que mapeia os dados não linearmente separáveis para uma dimensão superior onde uma separação linear pelo hiperplano é possível. Qualquer função que satisfaça as seguintes condições pode ser utilizada como *kernel*:

1. A função k precisa ser simétrica: $k(x_i, x_j) = k(x_j, x_i)$;

2. A matriz de *kernel* $K_{i,j} = k(x_j, x_i)$ precisa ser positiva semi-definida para qualquer x.

Neste trabalho utiliza-se como função de *kernel* a Função Base Radial, do inglês *Base Radial Function* (BRF) , conforme proposto e implementado por Joachims et al. (1999)[5]. A BRF é dada por:

$$k(x_i, x_j) = \exp^{-\gamma d(x_i, x_j)} \tag{3.6}$$

com $\gamma > 0$, onde d é alguma forma de métrica de distância. Alguns exemplos de possíveis métricas são:

- $d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^{\frac{1}{2}}$ para a BRF Sublinear;
- $d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^{1}$ para a BRF Laplaciana;
- $d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^{2}$ para a BRF Gaussiana;

Joachims et al. (1999) não informa a métrica utilizada em seu algoritmo.

[5]O algoritmo implementado por Joachims encontra-se publicamente disponível em: http://svmlight.joachims.org/

3.3 Transformada de Hough

A Transformada de Hough (Hough, 1962) é uma técnica utilizada para detecção de curvas parametrizáveis, como retas, círculos, elipses, etc. O tipo mais simples dessa transformada é conhecido com Transformada de Hough Linear e é utilizado para detecção de retas. Seu funcionamento será explicado a seguir.

Dado um ponto (x_o, y_o) no espaço, as retas que passam por esse ponto podem ser descritas como

$$x\cos(\Theta) + y\sin(\Theta) = \rho, \tag{3.7}$$

onde ρ é o comprimento do segmento de reta normal que parte da origem e Θ é o ângulo de ρ em relação ao eixo X (veja Figura 3.7.). Para qualquer ponto na mesma reta, os valores de ρ e Θ são constantes.

Em um problema de análise de imagens as coordenadas das bordas (isso é, os pontos (x_o, y_o)) são facilmente determináveis. Portanto, as variáveis para a Transformada de Hough Linear são os parâmetros ρ e Θ.

Para um ponto (x_o, y_o) no espaço cartesiano, indicam-se os possíveis valores ρ e Θ das retas que passam por esse ponto como pontos no espaço dos parâmetros de Hough. Essa transformação de retas em pontos é a Transformada de Hough para retas (veja Figura 3.7.). Se analisados em conjunto, esses pontos no espaço de Hough formam curvas de formas características (variáveis de acordo com a curva no espaço cartesiano em que a transformada foi aplicada).

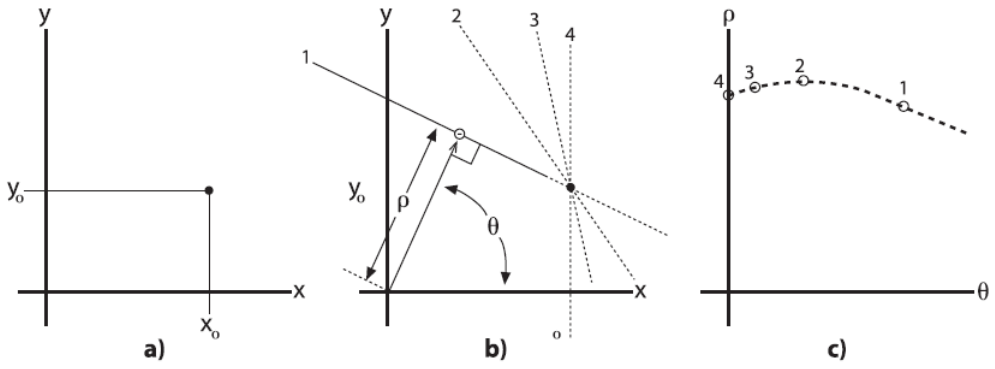


Figura 3.7: Um ponto (x_o, y_o) no plano (Fig. a) implica em múltiplas linhas, cada uma parametrizada por diferentes ρ e Θ (Fig. b). Cada linha implica em um ponto no espaço (ρ, Θ), que, se analisados em conjunto, formam uma curva de forma característica (Fig. c) (Kaehler & Gary, 2013).

Quando analisados no espaço de parâmetros de Hough, pontos colineares no espaço cartesiano se tornam evidentes, uma vez que resultam em curvas com um ponto (ρ, Θ) em comum (veja Figura 3.8.).

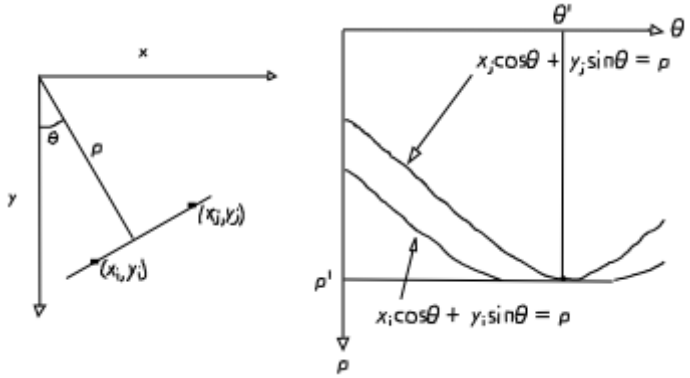


Figura 3.8: Pontos colineares no espaço cartesiano resultam em curvas com um ponto (ρ, Θ) em comum no espaço de parâmetros de Hough.[6]

A transformada é implementada realizando a quantização dos parâmetros no espaço de Hough em uma matriz de acumuladores. À medida que o algoritmo é executado, cada ponto (x_o, y_o) é transformado em uma curva (ρ, Θ) e o ponto correspondente no acumulador é incrementado. Picos na matriz do acumulador são fortes evidências de que uma reta existe na imagem.

Para detecção de círculos, o procedimento é bastante semelhante. Para a Transformada Circular de Hough a equação paramétrica é dada por

$$(x - a)^2 + (y - b)^2 = r^2, \tag{3.8}$$

onde a e b são as coordenadas do centro do círculo e r é o seu raio. Deve-se notar que a complexidade computacional do algoritmo começa a se elevar, uma vez que tem-se três coordenadas no espaço de parâmetros e o acumulador é 3D.

No presente trabalho se utiliza as funções de Transformada de Hough implementadas na biblioteca OpenCV. A Transformada Circular de Hough dessa biblioteca utiliza uma técnica diferente da original proposta por Hough (1962) afim de diminuir o custo computacional de um acumulador tridimensional.[7]

3.4 Estimação da Normal

Uma vez que a tomada tenha sido localizada é importante inferir a sua posição em um sistema de coordenadas de referência. A estimação da normal à sua superfície é, portanto, um passo necessário para a estimação de sua pose.

Dado um ponto p_i na superfície e sua vizinhança de pontos P^k, o problema de se determinar a normal a esse ponto é similar ao problema de se estimar a normal a um plano tangente à superfície, o que se resume em um problema de ajuste de mínimos quadrados em P^k (Rusu, 2010).

A solução para o vetor normal $\mathbf{n}$ é obtida pela análise dos autovetores e autovalores da matriz de covariâncias $C \in \mathbb{R}^{3\times3}$ de P^k, como indicado nas equações

[6]Disponível em: http://www.uio.no/studier/emner/matnat/ifi/INF4300/h09 /undervisningsmateriale/hough09.pdf

[7]Detalhes da implementação disponíveis em (Kaehler & Gary, 2013), pág. 158.

$$
\begin{aligned}
C &= \frac{1}{k}\sum_{i=1}^{k} \epsilon_i (p_i - \overline{p})(p_i - \overline{p})^T, \\
C\mathbf{v}_j &= \lambda_j \mathbf{v}_j, \\
j &\in 0, 1, 2,
\end{aligned}
$$

onde o termo ϵ_i representa um possível peso para p_i e é usualmente igual a 1 e $\overline{p}$ é a média de todos os pontos p_i. C é simétrica e positiva semi-definida e seus autovalores são números reais $\lambda_j \in \mathbb{R}$. Os autovetores $\mathbf{v}_j$ formam um sistema de coordenadas ortogonal, correspondente aos principais componentes de P^k. Os parâmetros x, y e z que representam o vetor normal $\mathbf{n}$ são extraídos do autovetor $\mathbf{v}_o$.

No presente trabalho se utilizou uma otimização da implementação proposta por (Brito de Sá, 2014).

3.5 Conclusão Parcial

Esse capítulo apresentou os conceitos teóricos do HOG, descritor utilizado para a identificação de objetos. Viu-se também uma maneira de classificar o objeto identificado com o auxílio de SVM. Uma abordagem para detecção de curvas parametrizáveis baseada na Transformada de Hough foi também discutida. Uma técnica para cálculo da normal a superfície da tomada também foi vista.

Capítulo 4

Arquitetura do Sistema

A primeira seção deste capítulo descreve a arquitetura física do sistema, apresentando os *hardwares* utilizados. A segunda seção descreve a arquitetura de *software* do sistema, bem como as bibliotecas utilizadas.

4.1 Arquitetura Física do Sistema

O sistema desenvolvido visa ser aplicado ao manipulador móvel MARIA (Figura 4.1) e faz uso dos seguintes componentes:

1. *Kinect* - Dispositivo dotado de uma câmera RGB e um sensor infravermelho de profundidade que é responsável por prover ao sistema os dados de imagem e profundidade.

2. *Notebook* - Dispositivo utilizando o ROS, onde é executado o sistema de visão computacional.

É importante ressaltar que o projeto destina-se à aplicação em ambientes fechados, uma vez que o sensor de profundidade do *Kinect* torna-se inoperacional em ambientes abertos.

4.2 Arquitetura de Código

O sistema baseia-se na arquitetura proposta por Dalal & Triggs (2005), constituindo-se do treinamento de uma SVM a partir dos descritores HOG, conforme visto no capítulo anterior. Ambos sistemas serão explicados em detalhes a seguir.

4.2.1 Sistema de Treinamento

O Sistema de Treinamento é utilizado para a geração dos vetores de gradientes a partir das imagens de treinamento. Uma implementação do algoritmo proposto por Dalal & Triggs (2005) encontra-se publicamente disponível na internet[1]. Ela foi desenvolvida

[1]Disponível no GitHub: https://github.com/DaHoC/trainHOG

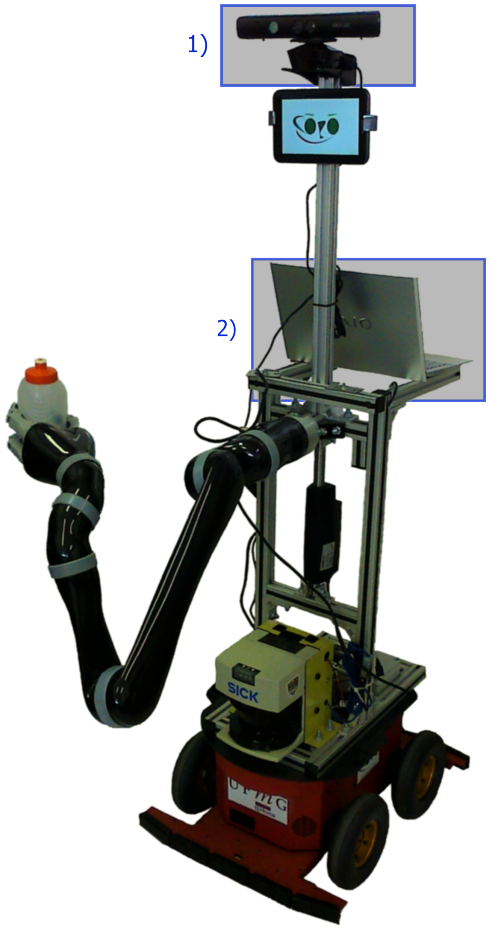


Figura 4.1: MARIA - *Manipulator Robot for Interaction and Assistance.*. Em 1) encontra-se o *Kinect* e em 2) o *notebook*.

pelo pesquisador, das áreas de robótica e inteligência artificial, Jan Hendriks. O algoritmo implementado no Sistema de Treinamento deste projeto foi desenvolvido à partir de uma adaptação do trabalho de Hendriks.

Para o desenvolvimento das funções de visão computacional, o sistema utiliza a biblioteca de código aberto OpenCV[2] (*Open Source Computer Vision*), que conta com diversas funções de tratamento de imagem, identificação de objetos e demais funções relacionadas à área. Dentre essas, encontra-se a API que implementa o HOG, conforme proposto por Dalal & Triggs (2005).

O funcionamento do Sistema de Treinamento segue o diagrama apresentado na Figura 4.2 e se resume em:

- O programa realiza a leitura de imagens divididas em dois diretórios: imagens positivas (imagens da tomada) e imagens negativas (imagens que não contenham tomadas);
- São calculados os descritores HOG para todas as imagens (vide Descritores HOG no Algoritmo 1);

[2]Maiores informações disponíveis em: http://opencv.org/

- Os descritores são fornecidos à uma SVM, que gera os vetores de suporte (vide Vetores de Suporte no Algoritmo 1);
- Os vetores são armazenados em um arquivo.

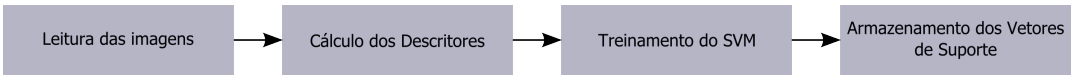


Figura 4.2: Arquitetura do Sistema de Treinamento

O Algoritmo 1 apresenta o pseudocódigo do Sistema de Treinamento.

A principal diferença dentre o Sistema de Treinamento deste projeto e aquele desenvolvido por Hendriks reside no fato de que esse sistema armazena os vetores de suporte em um arquivo para ser utilizado em um sistema de detecção a parte, que será implementado em um manipulador móvel, enquanto o projeto de Hendriks realiza o treinamento e detecção em um único algoritmo, mas destina-se ao uso de uma *webcam* convencional e limita-se à detecção bidimensional em imagens.

Algoritmo 1: Algoritmo do Sistema de Treinamento

Entrada: Imagens de treinamento
Saída: Arquivo contendo os Vetores de Suporte
Início

```
1   Realiza a listagem das imagens de positivas de treinamento
2   Realiza a listagem das imagens de negativas de treinamento
Descritores HOG
    para cada Imagem listada faça
3       Verifica dimensão e redimensiona se necessário (a imagem deve ter
        64 × 128 pixels)
4       Divide em células de 8 × 8
5       para cada Célula faça
6           para cada Pixel da célula faça
7               Calcula os vetores de gradientes
8
9           Organiza os vetores de gradientes em um histograma de 9 classes
10          Normaliza o histograma
11
12      Organiza os histogramas normalizados de cada célula em grupos de 4
13      para cada Grupo faça
14          Normaliza os histogramas agrupados
15
16      Descritor_HOG ⟵ Vetor contendo todos os histogramas
        normalizados, de todos os grupos
Fim Desc. HOG
Vetores de Suporte
    para cada Imagem positiva faça
17      Descritores_Positivos ⟵ Mapeamento não linear (Descritor_HOG)
18
19  para cada Imagem negativa faça
20      Descritores_Negativos ⟵ Mapeamento não linear (Descritor_HOG)
21
22  Calcula superfície ótima (que maximize a margem do classificador)
    utilizando Multiplicadores de Lagrange
23  Vetores_de_Suporte ⟵ Características de Descritores_Positivos e
    Descritores_Negativos que estejam à uma distância menor que a tolerância
    da margem da superfície de separação
Fim Vet. Suporte
24  Salva Vetores_de_Suporte em um arquivo
```

4.2.2 Sistema de Detecção

O Sistema de Detecção é responsável pela leitura das informações provenientes do *Kinect* e, a partir delas, determinação da pose completa da tomada.

O Sistema de Detecção é executado na plataforma ROS[3] (*Robot Operating System*), uma coleção de ferramentas, bibliotecas, e convenções para simplificar a tarefa de criar comportamentos complexos e robustos de robôs. O ROS provê serviços padrões de sistemas operacionais, tais como abstração de *hardware*, controle de dispositivos de baixo nível, implementações de funcionalidades comuns, transferências de mensagens entre processos e manutenção de pacotes.

Uma vez que o sistema operacional nativo do *Kinect* é o *Windows* e os códigos são executados fora desse, para poder se comunicar com esse dispositivo utiliza-se a biblioteca da OpenNI[4](*Open Natural Interaction*), companhia sem fins lucrativos que destinava-se ao desenvolvimento de interfaces de comunicação com dispositivos de interação humano-computador.

Para o tratamento das informações provenientes do sensor de profundidade, fez-se uso da biblioteca PCL[5] (*Point Cloud Library*), projeto de código aberto e que destina-se ao processamento de imagens 2D/3D e de nuvens de pontos. A biblioteca contém algoritmos do estado da arte nas áreas de filtragem, estimação de características e reconstrução de superfícies.

A arquitetura do Sistema de Detecção é apresentada na Figura 4.3 e é constituída de dois pacotes:

- Pacote do *Kinect*: Pacote nativo do ROS[6] que utiliza a biblioteca OpenNI e provê a interface de comunicação com o *Kinect*, através da publicação das informações do sensor em tópicos do ROS;
- Pacote de Detecção de Objetos: Executa o nó de detecção da tomada, que é responsável pelo reconhecimento e localização da pose dessa na imagem RGB e na nuvem de pontos.

O Pacote de Detecção de Objetos é composto de um único nó, cuja arquitetura é apresentada na Figura 4.3, e se resume em:

- O nó realiza a leitura dos vetores de suporte do arquivo;
- É realizada a inscrição no tópico no qual o *Kinect* publica as imagens e as nuvens de pontos (camera/depth_registered/points). O nó aguarda a publicação de alguma informação para dar continuidade à sua execução;
- Uma vez que o *Kinect* comece a operar, é realizada uma busca na imagem RGB pela tomada;

[3]Maiores informações disponíveis em: http://www.ros.org/

[4]A companhia foi comprada pela Apple em novembro de 2013 e atualmente o seu site encontra-se fechado. Entretanto, todos os códigos continuam disponíveis na GitHub. (http://apple.slashdot.org/story/14/03/02/1530202/apple-closes-openni-the-open-source-kinect-framework)

[5]Maiores informações disponíveis em: http://pointclouds.org/

[6]Maiores detalhes na documentação em: http://wiki.ros.org/openni_launch

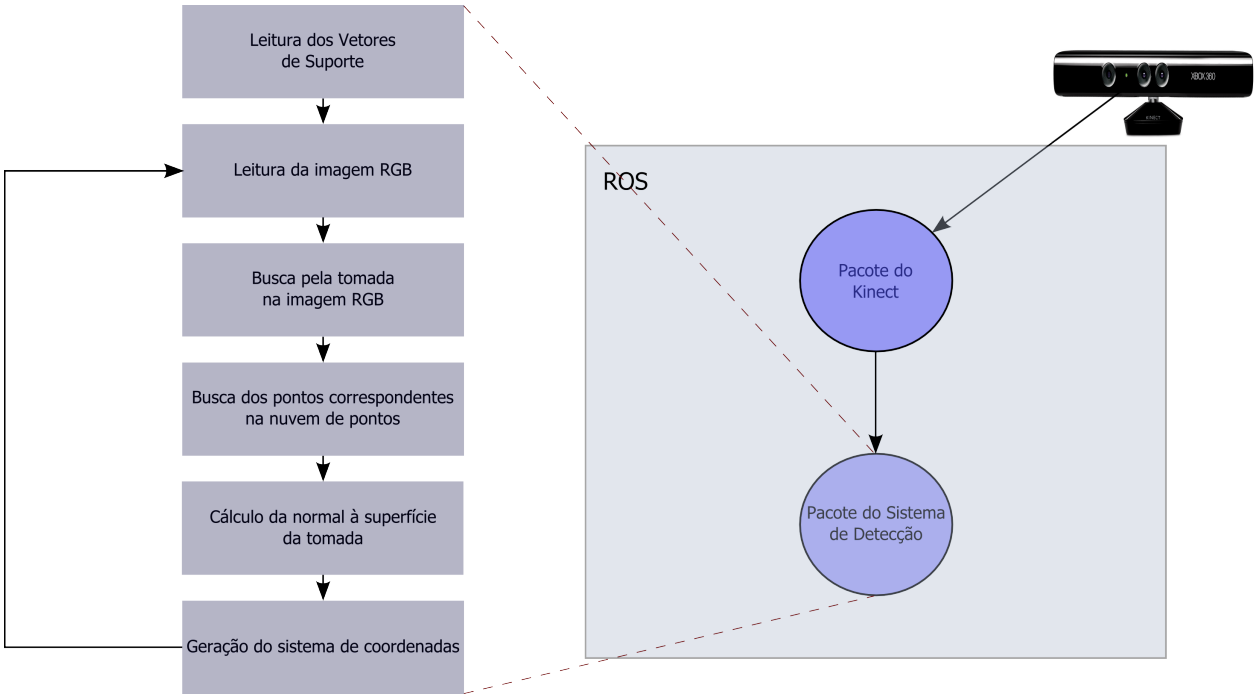


Figura 4.3: Arquitetura do Sistema de Visão Computacional

- Tendo sido detectada, a tomada tem sua localização indicada na imagem;
- O nó encontra os pontos correspondentes à localização da tomada na imagem na nuvem de pontos (vide o Algoritmo 4);
- Indica-se a localização da tomada na nuvem de pontos;
- É colocado um sistema de coordenadas no centro da tomada, ficando assim determinada a sua pose;

O Algoritimo 2 apresenta o pseudocódigo do Sistema de Detecção. Para se reduzir o tempo de processamento o algoritmo trabalha, em alguns momentos , com uma Região de Interesse, do inglês *Region Of Interest* (ROI). Isso possibilita que as etapas do processamento mais custosas sejam realizadas apenas na parcela da imagem realmente relevante.[7]

Como a distância dos furos da tomada ao seu centro são padronizados, o algoritmo utiliza essa informação para realizar uma filtragem de falsos positivos e determinar se os furos localizados são de energia ou de terra. O valor dessa distância em termos do sistema de coordenadas do *Kinect* foi determinado empiricamente e acrescido de uma tolerância. Dessa forma, as possíveis coordenadas $[xy]$ dos furos (estejam eles na vertical ou na horizontal), em relação ao centro da tomada, são previamente conhecidas. Caso apenas um furo de energia seja localizado, o outro é estimado espelhando-se o primeiro.

[7]Detalhes de implementação em: http://docs.opencv.org/2.4.9/modules/core/doc/basic_structures.html

Algoritmo 2: Algoritmo do Sistema de Detecção

Dados: Imagens e nuvem de pontos provenientes do Kinect
Resultado: Sistema de coordenadas que define a pose completa da tomada
Início

```
1  Lê os vetores de suporte do arquivo
2  Define os parâmetros da pesquisa do HOG (definidos empiricamente)
3  Inscreve-se no tópico do Kinect (camera/depth_registered/points)
4  enquanto Sistema estiver em execução faça
5      se Kinect publicar novos dados então
6          Realiza a pesquisa pela tomada na imagem RGB
7          para cada Tomada detectada faça
8              Desenha um retângulo sobre a tomada na imagem RGB
9              Localiza os furos da tomada (Algoritmo 3)
10             Desenha a reta do primeiro eixo do sistema de coordenadas na
               imagem RGB
11             Localiza a tomada na nuvem de pontos (Algoritmo 4)
12             Desenha um retângulo sobre a tomada na nuvem de pontos
13             Calcula a normal ao centro do sistema de coordenadas da tomada
               na nuvem de pontos (centro da tomada)
14             Define o segundo eixo do sistema de coordenadas da tomada como
               sendo a normal ao centro da tomada
15             Define o terceiro eixo do sistema de coordenadas da tomada como
               sendo o vetor ortogonal aos dois já existentes (calcula o produto
               vetorial entre esses)
16             Desenha o sistema de coordenadas na nuvem de pontos
17             Publica as coordenadas da pose do centro da tomada em um
               tópico do ROS
18
19
20
```

Algoritmo 3: Algoritmo de detecção do furo da tomada

Dados: Retângulo contendo a tomada
Resultado: Localização dos furos da tomada
Início

1. Define esse retângulo como a nova ROI
2. Busca pelo círculo central da tomada utilizando Transformada de Circular Hough
3. Define o menor retângulo que contenha o círculo central da tomada como a nova ROI
4. Busca pelos furos da tomada utilizando Transformada de Hough Circular
5. Filtra falsos positivos pela geometria da tomada (furos possuem distância bem definida do centro da tomada)
6. Define o primeiro eixo do sistema de coordenadas como sendo a reta que une os dois furos de energia da tomada

Algoritmo 4: Algoritmo de localização da tomada e seus pontos de interesse na nuvem de pontos

Dados: Tomada na imagem RGB
Resultado: Localização da tomada na nuvem de pontos
Início

1. Encontra a reta equivalente ao primeiro eixo do sistema de coordenadas na nuvem de pontos (função at() da PCL)
2. Obtém o ponto superior esquerdo do retângulo de detecção na imagem RGB
3. Encontra o ponto equivalente a ele na nuvem de pontos (função at() da PCL)
4. Obtém o ponto inferior direito do retângulo de detecção na imagem RGB
5. Encontra o ponto equivalente a ele na nuvem de pontos (função at() da PCL)

4.3 Conclusão Parcial

O sistema implementado depende de disponibilidade de dois componentes de *hardware*, um computador executando a plataforma ROS e um sensor *Microsoft Kinect*. Do ponto de vista de *software*, o sistema se divide em:

- Sistema de Treinamento: Realiza o treinamento de um SVM a partir de descritores HOG, conforme proposto por Dalal & Triggs (2005);
- Sistema de Detecção: Opera na plataforma ROS e realiza a detecção da tomada na imagem RGB e na nuvem de pontos, determinando a sua pose.

O capítulo seguinte irá apresentar alguns resultados obtidos pela utilização do sistema.

Capítulo 5

Resultados

A seguir serão apresentados testes do uso do algoritmo para reconhecimento das tomadas do laboratório de Manipuladores Robóticos da UFMG. A primeira seção discute como foi feito o treinamento do sistema e a segunda seção apresenta a execução do mesmo. A terceira seção apresenta análises de desempenho do algoritmo e a sessão final traz a conclusão do capítulo.

5.1 Sistema de Treinamento

O Sistema de Treinamento é utilizado para a obtenção das Máquinas de Vetores de Suporte, conforme explicado na Seção 4.2.1, pág.19. Para a obtenção das imagens de treinamento foram realizadas filmagens do laboratório com uma câmera de celular. A seguir, *frames* pertinentes do vídeo foram extraídos e redimensionados para 64×128 *pixels*, conforme proposto por Dalal & Triggs (2005).

A Figura 5.1 apresenta exemplos de imagens de tomadas utilizadas como exemplos positivos para o algoritmo e a Figura 5.2 apresenta o exemplo de uma imagem utilizada como exemplo negativo.[1] No total, foram fornecidos 410 exemplos positivos e 1388 exemplos negativos ao algoritmo, mantendo-se, aproximadamente, a proporção de 1:3 utilizada por (Dalal & Triggs, 2005).

Após fornecidas as imagens, o Sistema de Treinamento utiliza as funções do HOG disponíveis na biblioteca OpenCV para gerar os descritores para cada uma delas.[2]. Esses descritores são então fornecidos ao SVM para que sejam geradas as Máquinas de Vetores de Suporte[3], que são salvas em um arquivo.

5.2 Sistema de Detecção

A primeira etapa do processo de detecção da tomada consiste em uma busca na imagem RGB fornecida pelo *kinect*. Para tal, o Sistema de Detecção faz uso das SVMs fornecidas pelo Sistema de Treinamento. A Figura 5.3(a) apresenta uma imagem do sistema

[1] As figuras foram ampliadas para sua inclusão no texto para melhor visualização.

[2] Detalhes das funções do HOG disponíveis em: http://docs.opencv.org/2.4.9/modules/gpu/doc/object_detection.html

[3] Detalhes das funções do SVM disponíveis em: http://svmlight.joachims.org/

Figura 5.1: Exemplos de imagens utilizadas para o treinamento positivo do SVM. As imagens possuem diferentes poses e iluminações, e algumas apresentam também oclusões parciais.

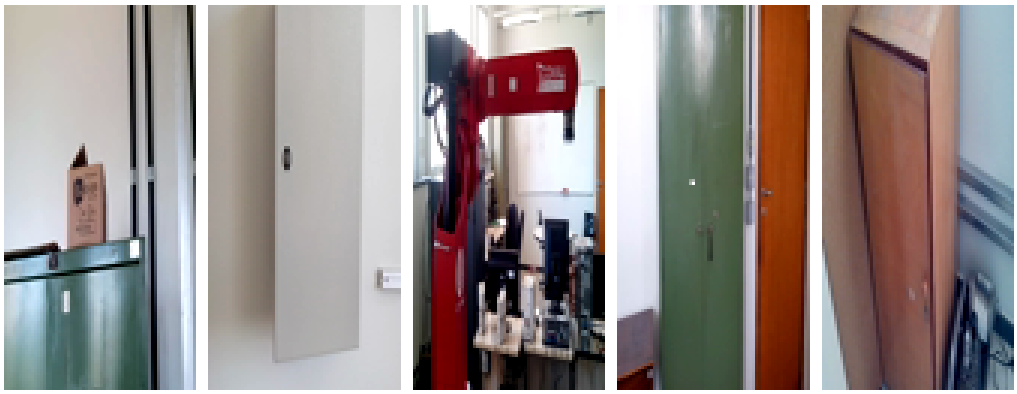
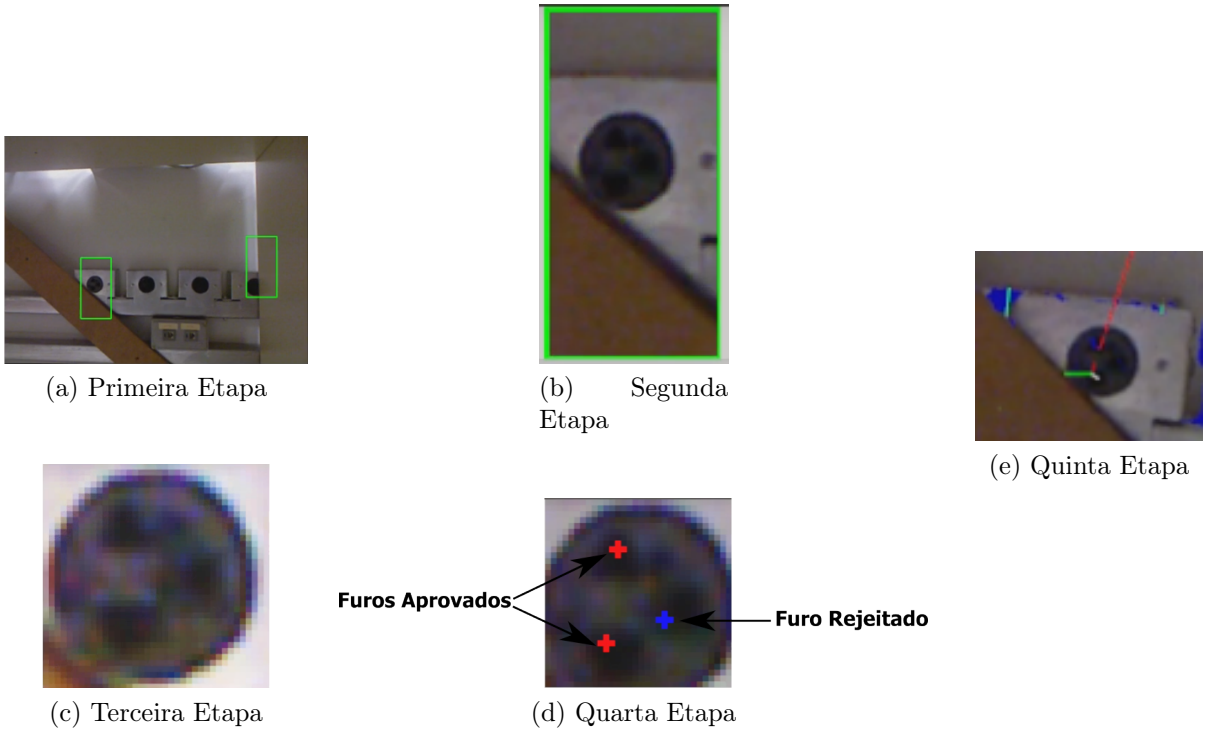


Figura 5.2: Exemplos de imagens utilizadas para o treinamento negativo do SVM. As imagens variam entre diversas partes e objetos presentes no laboratório.

em execução. Uma vez que as tomadas sejam localizadas, sua localização é indicada por meio de um retângulo verde. A seguir, o algoritmo define o retângulo verde de localização da tomada como a sua ROI. A Figura 5.3(b) apresenta uma imagem dessa etapa.

Utilizando a Transformada de Hough o algoritmo procura então pelo círculo central da tomada, o que permite que se localize o centro da mesma. Uma vez que esse círculo seja localizado, uma nova ROI é definida como o menor retângulo que o contenha. A Figura 5.3(c) apresenta uma imagem dessa etapa.

Posteriormente, o algoritmo utiliza novamente a Transformada de Hough, mas dessa vez para localizar os furos da tomada. Como os furos possuem uma distância padronizada do centro, essa informação é utilizada para eliminar falsos positivos e inferir a localização do segundo furo, caso apenas o primeiro seja localizado pela Transformada de Hough. A Figura 5.3(d) apresenta uma imagem dessa etapa. Os furos marcados em vermelho passaram no teste da validade de detecção, enquanto os marcados em azul indicam furos cujas posições foram consideradas inválidas devido à geometria padrão da tomada.



(a) Primeira Etapa

(b) Segunda Etapa

(e) Quinta Etapa

(c) Terceira Etapa

(d) Quarta Etapa

Figura 5.3: Funcionamento do Sistema de Detecção. As tomadas tem sua localização indicada por um retângulo verde (Fig. a). O retângulo verde contendo a tomada é definido como a ROI do algoritmo (Fig. b). A nova ROI do algoritmo é definida como o menor retângulo que contenha o círculo central da tomada (Fig. c). Furos marcados em vermelho passaram no teste da validade de detecção e os marcados em azul foram rejeitados (Fig. d). O sistema de coordenadas da tomada é definido pelo eixo que une seus furos, o eixo normal à sua superfície e o eixo ortogonal aos dois anteriores (Fig. e).

Uma vez que os furos tenham sido encontrados e suas posições validadas, encerram-se as etapas de detecção e dá-se início as etapas de estabelecimento do sistema de coordenadas que define a pose da tomada.

O primeiro eixo do sistema de coordenadas é definido como sendo uma reta que une os dois furos de energia da tomada. O segundo eixo é dado pela reta normal à superfície da tomada, reta essa estimada conforme os procedimentos explicados no Capítulo 3.4, pág.17. O terceiro e último eixo é definido como sendo ortogonal aos dois anteriores. A Figura 5.3(e) apresenta uma imagem do sistema de coordenadas da tomada.

5.3 Análise de Desempenho

Todos os testes foram realizados em um computador Intel®Core™2 Duo CPU E7400 @ 2.80GHz x 4, com sistema operacional Ubuntu 14.04.2 LTS de 64-bit e utilizando o ROS Indigo.

Para análise do desempenho do algoritmo, realizou-se a execução do Sistema de Detecção 1000 vezes e tirou-se a média do tempo gasto. O procedimento foi repetido 10

vezes, a fim de obter-se uma média mais precisa. A Tabela 5.1 apresenta os resultados obtidos. O tempo médio de execução para essa bateria de testes foi de $0,2870$ segundos.

Teste Número	**Tempo médio de execução (s)**
1	0,2935
2	0,2870
3	0,2997
4	0,2870
5	0,2786
6	0,2843
7	0,2799
8	0,2875
9	0,2874
10	0,2859
Média Final	0,2870

Tabela 5.1: Tempo de execução médio para cada 1000 execuções do algoritmo.

Realizou-se novamente 10 vezes o procedimento de 1000 execuções do algoritmo, mas dessa vez omitindo-se as janelas de visualização, tanto do OpenCV quando da PCL. A Tabela 5.2 apresenta os resultados obtidos.

Teste Número	**Tempo médio de execução (s)**
1	0,2753
2	0,2693
3	0,2693
4	0,2705
5	0,2629
6	0,2823
7	0,2630
8	0,2665
9	0,2667
10	0,2690
Média Final	0,2695

Tabela 5.2: Tempo de execução médio para cada 1000 execuções do algoritmo, sem o uso de janelas de visualização.

Pode-se perceber que a renderização das imagens e da nuvem de pontos para a visualização são responsáveis por aproximadamente $6,10\%$ do tempo de execução do algoritmo. Uma vez que o sistema tem por objetivo ser implementado no robô MARIA e a visualização dos resultados da detecção não se faz necessária, o tempo médio de execução do algoritmo é $0,2695$ segundos.

5.4 Conclusão Parcial

Foram apresentados resultados dos testes de execução do algoritmo, bem como o tempo de execução gasto pelo mesmo. O capítulo seguinte irá apresentar uma discussão das capacidades e limitações do sistema, além de sugestões de trabalhos futuros.

Capítulo 6

Discussão e Conclusão

A autonomia energética é um fator crítico para robôs móveis. Esse cenário não é diferente na robótica de assistência: uma vez que tem-se uma pessoa direta dependente do correto funcionamento do robô, é essencial garantir que esse seja capaz de se manter operacional.

Para garantia da independência energética dos robôs, vem-se desenvolvendo uma linha de pesquisas no intuito de torná-los capazes de se conectar autonomamente em tomadas elétricas convencionais. Um dos métodos de atingir esse objetivo consiste em prover ao robô um sistema de visão computacional.

O sistema desse projeto desenvolveu-se com essa finalidade. Foi elaborado um sistema de visão computacional capaz de permitir ao robô o reconhecimento de tomadas elétricas em meio ao ambiente, sem a necessidade de marcações especiais nas mesmas. Foi evidenciado sua capacidade de realizar tal reconhecimento mesmo em tomadas parcialmente oclusas.

Uma das limitações do sistema reside no fato de que, por utilizar um SVM tradicional, ele é capaz de realizar apenas o reconhecimento binário entre classes. Dessa forma, o sistema consegue apenas reconhecer um tipo de tomada em meio ao ambiente, sendo necessário um novo treinamento das máquinas de vetores de suporte para o reconhecimento de uma tomada significativamente diferente.

Outra limitação está no uso restrito do sistema à ambientes internos, uma vez que o sensor de profundidade do *Kinect*, um sensor infravermelho, perde a capacidade de prover informações em áreas externas. Problema semelhante ocorre em ambientes excessivamente iluminados.

Possíveis expansões do projeto podem permitir ao robô o reconhecimento de diferentes objetos de interesse, facilitando e ampliando as capacidades de interação humano-robô.

6.1 Trabalhos Futuros

Diferentes descritores possuem diferentes capacidades e limitações no que diz respeito à robustez a variações sofridas pela imagem, como variações de luminosidade, rotações e mudanças de escala. A avaliação do desempenho desses diferentes descritores como substitutos do HOG no sistema desenvolvido seria uma interessante linha de continuidade ao projeto.

Uma expansão do sistema poderia se dar pela exploração do uso de *SVM-Multiclass* (Hsu, 2002), uma adaptação das máquinas de vetores de suporte para a classificação de problemas de múltiplas classes. Isso permitiria ao sistema o reconhecimento de múltiplos tipos de tomadas simultaneamente.

Para acrescentar maior robustez ao sistema, uma técnica que realizasse o reconhecimento tanto na imagem RGB quanto na nuvem de pontos poderia ser implementada. Tal arquitetura é proposta por Spinello & Arras (2012) e faz uso de uma mistura hierárquica de especialistas para gerenciar as detecções em ambos ambientes.

Referências Bibliográficas

Bay, H., Ess, A., Tuytelaars, T., & Gool, L. V. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3), 346–359.

Brito de Sá, G. C. (2014). Implementação de um algoritmo de visão computacional para reconhecimento de maçanetas de portas. Projeto final de curso, Universidade Federal de Minas Gerais.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2), 121–167.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Dalal, N. & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 886–893.

Fujita, M. (2001). AIBO: Toward the Era of Digital Creatures.

Han, J., Shao, L., Xu, D., & Shotton, J. (2013). Enhanced computer vision with Microsoft Kinect sensor: a review. *IEEE transactions on cybernetics*, 43(5), 1318–34.

Hough, P. (1962). Method and means for recognizing complex patterns. *U.S. Patent 3,069,654*.

Hsu, C.; Lin, C. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13, 415–425.

Joachims, T., Dortmund, U., & Joachimscsuni-dortmundde, T. (1999). Making Large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, (pp. 41–56).

Jones, J. (2006). Robots at the tipping point: the road to iRobot Roomba. *IEEE Robotics & Automation Magazine*, 13(1).

Kaehler, A. & Gary, B. (2013). *Learning OpenCV*. United States of America: O'Reilly Media, Inc., first edition.

Kartoun, U., Stern, H., Edan, Y., Feied, C., Handler, J., Smith, M., & Gillam, M. (2006). Vision-Based Autonomous Robot Self-Docking and Recharging. In *2006 World Automation Congress* (pp. 1–8).: IEEE.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 91–110.

Mayton, B., LeGrand, L., & Smith, J. R. (2010). Robot, feed thyself: Plugging in to unmodified electrical outlets by sensing emitted AC electric fields. In *2010 IEEE International Conference on Robotics and Automation* (pp. 715–722).: IEEE.

Meeussen, W., Wise, M., Glaser, S., Chitta, S., McGann, C., Mihelich, P., Marder-Eppstein, E., Muja, M., Eruhimov, V., Foote, T., Hsu, J., Rusu, R. B., Marthi, B., Bradski, G., Konolige, K., Gerkey, B., & Berger, E. (2010). Autonomous door opening and plugging in with a personal robot. In *2010 IEEE International Conference on Robotics and Automation* (pp. 729–736).: IEEE.

Roh, S.-g., Park, J. H., Lee, Y. H., Song, Y. K., Yang, K. W., Choi, M., Kim, H.-s., Lee, H., & Choi, H. R. (2008). Flexible Docking Mechanism with Error-Compensation Capability for Auto Recharging System of Mobile Robot. 6(5).

Rusu, R. B. (2010). Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz*, 24(4), 345–348.

Silverman, M. C., Jung, B., Nies, D., & Sukhatme, G. S. (2003). Staying Alive Longer: Autonomous Robot Recharging Put to the Test.

Spinello, L. & Arras, K. O. (2012). Leveraging RGB-D Data: Adaptive fusion and domain adaptation for object detection. In *2012 IEEE International Conference on Robotics and Automation* (pp. 4469–4474).: IEEE.

W. Walter (1950). An imitation of life. *Scientific American*, 182, 42–45.

Watson, D. P. & Scheidt, D. H. (2005). Autonomous Systems. 26(4), 368–369.