Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Departamento de Engenharia Elétrica

PEDRO HENRIQUE MEIRA DE ANDRADE

ROBÔ MÓVEL COM VISÃO COMPUTACIONAL USANDO A BIBLIOTECA OPENCV PARA SEGUIMENTO DE LINHA

João Pessoa, Paraíba
Junho de 2016

PEDRO HENRIQUE MEIRA DE ANDRADE

ROBÔ MÓVEL COM VISÃO COMPUTACIONAL USANDO A BIBLIOTECA OPENCV PARA SEGUIMENTO DE LINHA

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Área de Concentração: Controle e Automação

Orientador:
Professor Doutor Alexsandro José Virgínio Santos

João Pessoa, Paraíba
Junho de 2016

PEDRO HENRIQUE MEIRA DE ANDRADE

ROBÔ MÓVEL COM VISÃO COMPUTACIONAL USANDO A BIBLIOTECA OPENCV PARA SEGUIMENTO DE LINHA

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Área de Concentração: Controle e Automação

Aprovado em / /

Professor Avaliador
Universidade Federal da Paraíba
Avaliador

Professor Doutor Alexsandro José Virgínio Santos
Universidade Federal da Paraíba
Orientador, UFPB

Dedico este trabalho à minha família, minha base e meu porto seguro. E aos meus amigos com quem sempre desabafei nos momentos tristes e alegres.

AGRADECIMENTOS

Agradeço a Deus por seu infinito amor por mim que me concedeu saúde, inteligência e força para enfrentar e vencer obstáculos.

Agradeço a Maria, minha Mãezinha do Céu, que foi uma intercessora fiel e um refúgio nos momentos tristes.

Agradeço a minha família, meu porto seguro. Agradeço a meus pais, Neto e Valéria, que me deram como maior herança o incentivo aos estudos. E ao meu irmão, Luís, companheiro de todas horas.

Agradeço a Andréa, minha namorada, que me ajudou nos momentos bons e difíceis durante todo o curso tendo sempre uma palavra de conforto.

Agradeço a meus amigos que aguentaram meus momentos de tristeza e que se alegraram comigo nos momentos de alegria.

Agradeço aos meus amigos do EJC, especialmente do Infinito Amor e dos Clorofilhos de Deus, que me motivam a ser uma pessoa melhor a cada dia.

Agradeço aos meus amigos do ECRI que me ensinam que a pureza de coração é possível se a buscarmos com afinco.

Agradeço aos meus amigos de turma, o eterno PPG, que sem eles eu não teria chegado tão longe. A André, Andréa, Geórgia, Gustavo, Jéssica, Larissa, Marcelo, Patrycia e Thaís, meu mais sincero agradecimento.

Agradeço a todos os professores que nos ensinaram que a vida é feita de desafios e que a graça é resolvê-los. E agradeço especialmente ao meu orientador, Alexsandro, pelo empenho dedicado a me ensinar, pelo auxílio e pela disponibilidade de tempo.

"Deus é bom o tempo todo
e
o tempo todo Deus é bom."

RESUMO

Robôs móveis são aplicados em diversas situações, tanto para execução de tarefas como sendo alvo de estudos. Uma das configurações mais simples é a categoria de robôs seguidores de linha, que são usados desde em competições acadêmicas a ambientes industriais.

Neste trabalho, a plataforma utilizada é um pequeno robô móvel tracionado a esteiras e dotado de uma câmera, capaz de seguir uma linha no chão. O controle cinemático é implementado em uma Raspberry Pi que também é responsável pelo processamento da imagem. A grande vantagem dessa metodologia é que com a câmera o robô pode adiantar prováveis curvas e alterar a velocidade e posição conforme a trajetória, tornando o movimento mais eficaz. Foi utilizada a biblioteca OpenCV que é a base para todo o tratamento da imagem e por consequência, do controle de velocidade e direção. O acionamento dos motores e a leitura dos sensores foram feitos usando a plataforma Arduino interligado a Raspberry Pi.

Palavras-chave: Robô móvel, visão computacional, Biblioteca OpenCV, Raspberry Pi, Arduino, Câmera, Seguidor de trajetória.

ABSTRACT

Mobile robots are applied in various situations, both for performing tasks such as being studied at schools and universities. One of the simplest configurations is the category of robots line followers, which are used in academic competitions since the industrial environments.

The developed platform is a mobile robot that will follow a line. The system will be equipped with a camera and will be controlled by the Raspberry Pi which will also be responsible for image processing. The main advantage of this method is that with the camera the robot can anticipate likely curves and change the speed and position as the trajectory. So this configuration provides a accuratest movement. The OpenCV library is the basis for all treatment of the image and therefore, speed control and direction was used. The sensor measures were made using the Arduino platform that is communicated with the Raspberry Pi by serial.

Keywords: Mobile robot, computer vision, OpenCV Library, Raspberry Pi, Arduino, follower, camera.

LISTA DE ILUSTRAÇÕES

.

SUMÁRIO

OBJETIVOS

OBJETIVO GERAL

Este trabalho tem como objetivo desenvolver um sistema robótico móvel tracionado por esteira e provido de câmera de vídeo para detecção trajetória capaz de seguir uma linha implantada no chão.

OBJETIVOS ESPECÍFICOS

- Estudar os fundamentos de processamento de imagem e a biblioteca OpenCV;
- Implementar um filtro capaz de detectar uma faixa instalada sobre o plano;
- Determinar um modelo cinemático para o veículo de esteiras;
- Implementar um controlador dinâmico para o sistema de acionamento dos motores do veículo.

1 INTRODUÇÃO

A área da Robótica sempre atraiu o interesse da sociedade, gerando expectativas e até criando "mitos" em relação ao tema. Ao longo dos anos muitas previsões foram feitas sobre o que se tornaria realidade e muitas se tornaram factíveis, outras ainda não.

A palavra robô é derivada da palavra tcheca *robota*, a qual é traduzida livremente por "trabalhador humilde". Ela apareceu pela primeira vez na peça de KarelCapek, R. U. R. (*Rossum's Universal Robots*) em 1921. Nessa peça, um inventor chamado Rossum cria uma raça de trabalhadores feitos de partes biológicas, inteligentes o bastante para substituir os seres humanos em qualquer trabalho. Eles eram servos para os humanos (ROMERO et al., 2014).

O avanço na Robótica foi muito impulsionado a partir do desenvolvimento crescente dos computadores e de suas capacidades de processamento, assim como do *hardware* que ficou cada vez menor, consumindo menos energia e permitindo maior autonomia. Esses avanços, tanto no *hardware* como no *software*, permitem o estudo de áreas novas como processamento de imagem, reconhecimento de voz, tomada de decisão, dentre outras.

Os robôs podem ser utilizados para tarefas 3D (*Dirty, Dulland and Dangerous*), traduzida livremente por "Suja, Maçante e Perigosa" (MURPHY, 2000). Por exemplo:

- Tarefas que coloquem a vida do ser humano em risco, como em estações espaciais e usinas nucleares;

- Uso humanitário (busca e salvamento);

- Cirurgias minimamente invasivas (videolaparoscopia);

- Substituição de humanos em tarefas repetitivas e tediosas.

- Uso doméstico para limpeza de ambientes como o *Roomba* que aspira o pó, ilustrado na Figura 1.

Figura 1. Robô aspirador de pó (Roomba da iRobot).
Disponível em: <http://store.irobot.com/irobot-roomba-880/product.jsp?productId=28516906>. Acesso em 18/04/2016.

Uma grande área em desenvolvimento na Robótica é a de robôs móveis. Ou seja, robôs que se locomovem dentro de um ambiente (seja terrestre, aquático ou aéreo) de diversas formas como veículos com rodas, com esteiras, com pernas mecânicas, etc, apresentando capacidades de percepção e atuação.

O grande desafio para a robótica móvel é que os robôs devem ser capazes de atuar em ambientes desconhecidos e de reagir perante situações inusitadas. Grande parte das dificuldades consistem em programar robôs considerando o grau de incerteza de seus atuadores e sensores. Além do fato que o ambiente geralmente é dinâmico e imprevisível (DUDEK et al., 2000) e (THRUN et al., 2005).

Os robôs móveis podem ser caracterizados pela sua capacidade de se deslocar, podendo ser de modo guiado, semiautônomo ou totalmente autônomo (JUNG et al., 2005). Estes têm sido usados em áreas como aspiração de pó, entrega de correspondência, vigilância predial, busca e salvamento, dentre outras.

Os veículos tracionados a esteiras, similares aos tanques de guerra, apresentam maior área de contato com o solo (comparando-se a veículos com rodas) tendo menos derrapagem da direção do movimento. Permitindo assim seu uso em terrenos impróprios para rodas como áreas com pedra, lama ou neve. A desvantagem é o grande atrito lateral produzido pelas esteiras em curvas, o que produz um gasto excessivo de energia e imprecisões cinemáticas em manobras curvilíneas.

O robô desenvolvido neste trabalho é um veículo dotado de esteiras que é capaz de identificar e seguir uma faixa no chão utilizando uma câmera para se guiar e planejar a rota da trajetória que será descrita.

2 REVISÃO BIBLIOGRÁFICA

A robótica até pouco tempo atrás estava concentrada nos robôs manipuladores para a indústria. Porém a área dos robôs móveis, que incluem os veículos aéreos, aquáticos, dotados de pernas, de esteiras; têm recebido uma maior atenção do setor tecnológico e de pesquisa nas universidades (JUNG et al., 2005).

O problema de seguimento de trajetória de Robôs Móveis Autônomos (RMAs) consiste em garantir que o veículo seguirá um caminho predeterminado de forma autônoma. Para conseguir este objetivo os RMAs devem estar providos de sensores, atuadores e de um sistema que receba, trate, interprete e estabeleça um controle de navegação (OLLERO, 2001).

A estratégia de controle para a navegação de RMAs é tipicamente organizada em cascatas ilustrada na Figura 2. No nível 4, o mais alto, são feitos o planejamento e a geração da trajetória. No nível 3 geralmente são executados os algoritmos de controle responsáveis pela condução do veículo baseados nos modelos cinemáticos do sistema. No nível inferior (nível 2) é executado o controle da dinâmica. Por exemplo, o controle da dinâmica lateral nos veículos dotados de rodas e esteiras. E por fim, os sensores e atuadores ficam na base da estrutura piramidal no nível 1 (ROMERO et al., 2014).

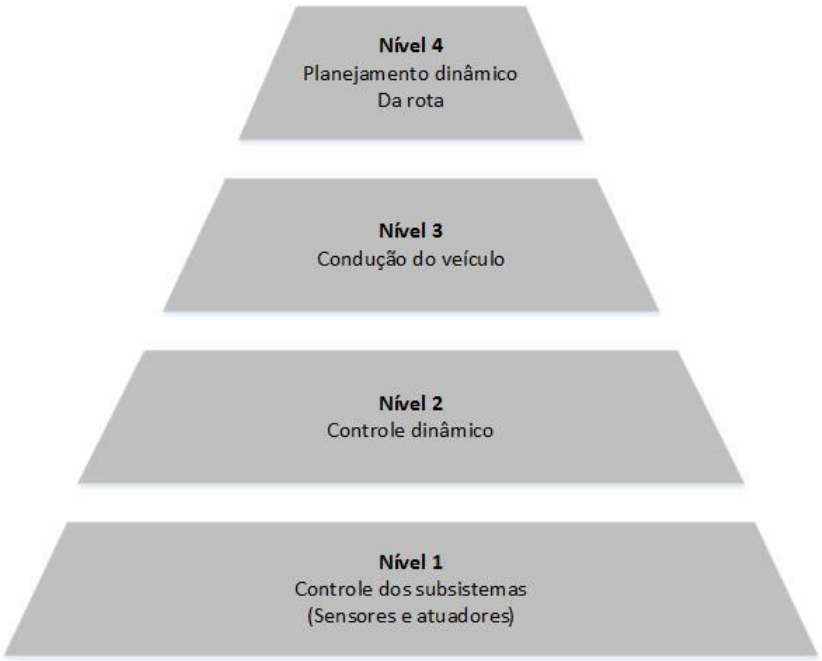


Figura 2. Níveis de controle de um RMA.

O problema de planejamento de rotas consiste em, dadas as posições inicial e final de um robô, descobrir uma sequência de movimentos a ser executada para que ele saia da primeira e chegue na segunda de forma exata, eficaz e sem colidir com possíveis obstáculos.

Existem diferentes técnicas para gerar a trajetória de referência e muitas delas usam a informação da posição atual do robô e, de maneira geral, realiza o cálculo do planejamento em dois passos: (i) escolhe-se um ponto, qualquer ou que tenha alguma relação predeterminada, sobre a trajetória de referência como ponto de destino; (ii) com a informação deste ponto e da posição atual, define-se o caminho a ser seguido (OLLERO, 1991).

3 VISÃO COMPUTACIONAL

3.1 INTRODUÇÃO

A diferença entre o processamento de imagem e a visão computacional é muito tênue. O processamento tem como entrada uma imagem e a saída pode ser uma imagem modificada ou um conjunto de dados ou informações extraídas. Enquanto que a visão computacional procura emular a visão humana e também assume como entrada uma imagem, mas o grau de interpretação é mais alto. Isto é, a saída pode ser uma imagem ou dado, mas a informação retirada é mais complexa.

Os processosque partem do processamento da imagem até a visão computacional são divididos em: baixo-nível, nível-médio e alto nível. Os processos de baixo-nível são aqueles de natureza mais simples como a redução de ruído, através de filtros, ou o tratamento do contraste em uma imagem. Os processos de nível médio são operações como a segmentação, em que a imagem é dividida geralmente por cor ou proximidade, e a classificação em que objetos podem ser reconhecidos. Já os processos de alto-nível estão ligados a uma maior emulação da visão humana, sendo relacionados a tarefas de cognição como reconhecer caracteres, pessoas ou seguir objetos (GONZALEZ, 2006).

Uma das principais bibliotecas para desenvolvimento de sistemas que envolvam visão computacional é a OpenCV que será mostrada a seguir.

3.2 BIBLIOTECA OPENCV

A biblioteca OpenCV que é uma biblioteca de código aberto disponível gratuitamente na internet desenvolvida pela Intel e é escrita em C e C++. É compatível com Linux, Windows e Mac OS X. Um dos objetivos da OpenCV é fornecer um uso simples de toda a sua infraestrutura de visão computacional. A biblioteca contém mais de 500 funções que são utilizadas em diversas áreas como imagens médicas, calibração da câmera, visão estéreo e na robótica (BRADSKI et al., 2008).

A biblioteca foi concebida com o objetivo de fazer a infraestrutura de visão computacional disponível universalmente. Como é um software livre há a colaboração

de programadores do mundo inteiro para o desenvolvimento de novos algoritmos e aperfeiçoamento dos existentes. Desde 1999, a OpenCV tem sido usada em muitas aplicações, produtos e em pesquisas científicas.

A biblioteca está dividida em cinco grupos de funções:

- Processamento de imagens.
- Análise estrutural.
- Análise de movimento e rastreamento de objetos.
- Reconhecimento de padrões.
- Calibração de câmera e reconstrução 3D (visão estéreo).

3.3 PROCESSAMENTO DE IMAGENS

Os sistemas que envolvem visão computacional geralmente necessitam da etapa de um processamento de imagem em que a imagem deve ser tratada para um formato ou tamanho conveniente. Também são feitas filtragens para diminuir ou remover ruídos da aquisição da imagem.

Os ruídos aparecem de diversas fontes, como o tipo de sensor utilizado, as condições climáticas, a iluminação do ambiente e a posição relativa entre o objeto de interesse e a câmera. Os filtros podem ser espaciais (filtros que atuam diretamente na imagem) ou de frequência (onde a imagem é inicialmente transformada para o domínio da frequência, é feita a filtragem e transformada de volta para o domínio do espaço) (MARENGONI, STRINGHINI, 2009).

O domínio espacial se refere à imagem em si, isto é, os métodos manipulam de forma direta os pixels da imagem. Por exemplo, uma operação muito utilizada é a binarização da imagem em que se estabelece um valor de corte *c* em que os valores acima de *c* são transformados no valor 1 (branco) e os valores menores são transformados em 0 (preto).

Já no domínio da frequência a imagem é representada por uma soma ponderada infinita de um conjunto de funções senoidais. Esta representação mostra que as mudanças rápidas na imagem indicam frequências altas e mudanças mais suaves indicam frequências baixas. Essa mudança de base é feita no processador utilizando a Transformada Discreta de Fourier (GONZALEZ, WOODS, 2006).

3.4 SISTEMAS DE CORES

Existem diversos modelos de representação das cores como o HSV (*Hue-Saturation-Value*), YUV (dedicado aos vídeos analógicos), sendo o RGB (*Red-Green-Blue*) muito utilizado.

O sistema mais conhecido é o RGB em que o principal propósito é a reprodução de cores em dispositivos eletrônicos como monitores de TV e computadores, retroprojetores, scanners, câmeras digitais. Uma cor no modelo RGB é descrita pela quantidade de vermelho, verde e azul que contém.

A representação mais usual para as cores em RGB é com 1 byte (8 bits) tendo uma escala de 0 a 255. Assim o vermelho completo é indicado por (255, 0, 0), o verde por (0, 255, 0) e o azul por (0, 0, 255). E todas as outras cores são combinações dessas três.

O sistema de cores HSV descreve o espaço de cores em três parâmetros:

Matiz (*Hue*): é o tipo da cor e abrange todo o espectro, desde vermelho até o violeta, mais o magenta. Varia de 0 a 360, mas pode ser normalizado para valores entre 0 e 100%.

Saturação: também chamado de “pureza”. Quanto menor esse valor mais tons de cinza aparecerão na imagem. Quanto maior o valor, mais “pura” é a imagem. Atinge valores de 0 a 255.

Valor (brilho): Define o brilho da cor. Atinge valores de 0 a 255.

3.5 CÂMERAS

As câmeras capturam determinados comprimentos de onda da luz incidente na lente da câmera. A luz capturada é convertida e passada para forma digital, podendo representar apenas a luminosidade, no caso das imagens preto e branco, ou ser representada pelas componentes básicas que formam uma cor. Geralmente as imagens coloridas são representadas pela intensidade de suas componentes (BRADSKI et al., 2008).

Existem câmeras especiais e outras associações de câmeras que são usadas pela Robótica para o processamento de imagem e para a visão computacional. Um exemplo são as câmeras capazes de compreender 360° do ambiente usando espelhos e lentes

especiais. Há também a associação de duas câmeras, que através da defasagem das imagens é possível determinar a profundidade dos elementos estudados. Hoje em dia essa configuração é chamada de RGB-D (RGB + *Depth*), referentes à imagem e à profundidade da cena.

4 MODELAGEM DO ROBÔ

4.1 INTRODUÇÃO

Existem diversos tipos de configurações para o deslocamento de robôs e cada uma apresenta um modelo cinemático e dinâmico respectivo. O uso de duas rodas com motores independentes e opcionalmente mais uma roda livre se baseia no acionamento diferencial. Já o uso de um esterçamento da direção tem outro modelo, assim como robôs aquáticos ou aéreos que possuem um modelo muito influenciado pela dinâmica devido ao menor atrito com a superfície apoiada. Finalmente, o robô de esteiras irá mudar de direção através do deslocamento das esteiras definindo uma cinemática conhecida como *skidsteering*, "direção de derrapagem", numa tradução livre (ROMERO et al., 2014).

Os robôs que podem se deslocar livremente (sem restrições de graus de liberdade) são denominados holonômicos como os veículos que apresentam rodas onidirecionais, já os carros e veículos com esteiras são exemplos de robôs não-holonômicos. A restrição não-holonômica impede que o robô execute deslocamentos normais às suas rodas, quando não há deslizamento (JOTA, FIGUEIREDO, 2004).

O robô desenvolvido é um veículo móvel tracionado por esteiras e tal característica implica que seus modelos dinâmico e cinemático são influenciados pelas características físicas do solo. Os dois principais fenômenos de interação das esteiras com o solo são a derrapagem e o deslizamento.

O deslizamento ocorre quando o chão está escorregadio e as esteiras estão sendo tracionadas, mas o carro não possui um deslocamento vetorial positivo.

A derrapagem está associada com o movimento do veículo com um atrito dinâmico no sentido lateral que atinge toda a esteira. Esse fenômeno ocorre quando o veículo executa uma curva e provoca o arrasto na esteira (SANTOS, 2015).

Neste trabalho o deslizamento e a derrapagem não serão estudados com profundidade. O solo será considerado liso e plano.

4.2 MODELO CINEMÁTICO

O veículo de esteira apresenta um acionamento diferencial. O modelo possui dois motores independentes, ilustrado na Figura 3. Considerando o plano cartesiano XY, temos que x e y são as coordenadas do robô, θ é o ângulo da direção, v e ω são as velocidades linear e angular, respectivamente; vel_D e vel_E são as velocidades linear de cada roda: direita e esquerda; e o raio da roda é representado por r e a distância entre elas é dada por L.

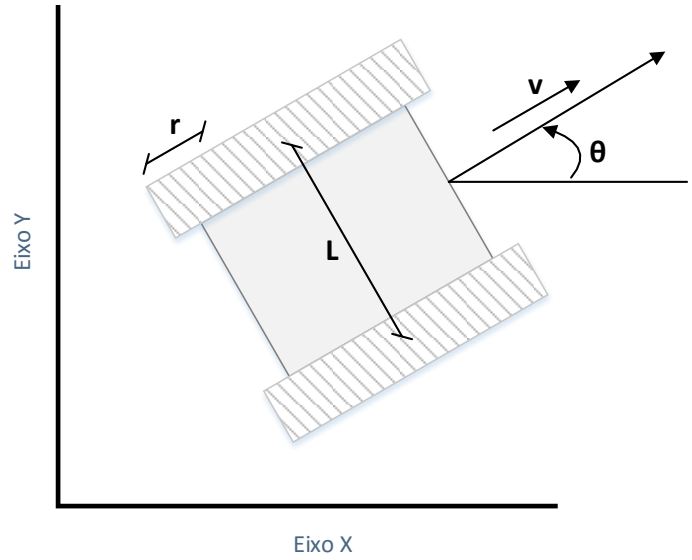


Figura 3. Modelo de um veículo de esteiras.

De acordo com Santos (2015), as equações que regem o modelo cinemático para um veículo de duas rodas com acionamento diferencial são:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \text{sen}\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} r & r \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \text{vel}_D \\ \text{vel}_E \end{bmatrix} \tag{2}$$

Combinando as equações (1) e (2):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r \cdot cos\,\theta & r \cdot cos\,\theta \\ r \cdot sen\theta & r \cdot sen\theta \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} vel_D \\ vel_E \end{bmatrix} \tag{3}$$

Então tem-se que as entradas do modelo cinemático são as velocidades das rodas direita e esquerda vel_D evel_E.

4.3 MODELO ELETROMECÂNICO

> "O motor de corrente continua básico, consistindo de armadura, escovas e enrolamentos é usado largamente em aplicações de robótica móvel pelo fato dos veículos serem alimentados por baterias de corrente contínua. O custo relativamente baixo de motores deste tipo popularizou o uso em pequenos veículos, cadeiras de rodas para deficientes e praticamente a totalidade de robôs móveis terrestres usados por pesquisadores acadêmicos (NGUYEN, MORRELL, et al. 2004)."

O projeto de um controlador depende do conhecimento do modelo do sistema e de todos os seus componentes, desta maneira o modelamento do motor de corrente contínua tem a tensão de armadura e o torque de carga como variáveis de entrada; e a velocidade e a posição do eixo do motor como variáveis de saída (JACOBINA, 2005).

O modelo dinâmico da máquina pode ser dado pelas seguintes equações baseadas no circuito equivalente ilustrado na Figura 4 e que possui as seguintes variáveis e parâmetros nas equações:

i_a : corrente de armadura [A], v_a : tensão de armadura [V],

e_a : força contra eletromotriz [V], v_e : tensão de excitação [V],i_e : corrente de excitação [A].

ω_r : velocidade angular do eixo [rad/s],

r_a : resistência de armadura [Ω], r_e : resistência de excitação [Ω],

l_a : indutância de armadura [H], r_e : indutância de excitação [H],

c_e : conjugado eletromagnético [Nm], c_m : conjugado de carga [Nm],

F_m : coeficiente de atrito [MKS], J_m : momento de inércia da máquina [MKS].

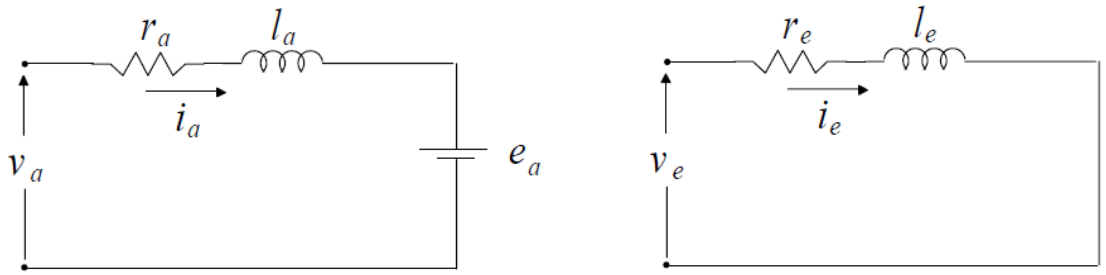


Figura 4. Circuito equivalente do motor de corrente contínua.

Equações elétricas:

$$v_a = r_a i_a + l_a \frac{di_a}{dt} + e_a \tag{5}$$

$$v_e = r_e i_e + l_e \frac{di_e}{dt} \tag{6}$$

Equação mecânica:

$$c_e - c_m - F_m \omega_r = J_m \frac{d\omega_r}{dt} \tag{7}$$

4.4 ENERGIA DISSIPADA ENTRE A ESTEIRA E O SOLO

A grande diferença de um veículo tracionado por esteiras para um tracionado por rodas com acionamento diferencial é o atrito causado pelo arrasto lateral das esteiras em movimentos curvilíneos.

O efeito do arrasto lateral são perdas por atrito entre a esteira e o solo. Nas curvas existe o aumento da carga atribuída aos motores de corrente contínua aumentando as perdas por efeito Joule. Quanto maior a interação do sistema esteira-solo, maiores serão as perdas (YOUN, TCHAMNA, et al., 2014).

Este efeito é desprezado no modelamento de veículos tracionados por rodas porque a superfície de contato é muito pequena, podendo ser aproximada a um ponto nos modelos mais usuais.

Os veículos de esteira não apresentam derrapagem lateral quando as velocidades das duas esteiras são iguais. Apenas quando há uma diferença entre as velocidades surge a ação da força de atrito. Admitindo o veículo como um corpo rígido, as forças de atrito não dependem da velocidade, apenas do peso do corpo e do coeficiente de atrito entre a esteira e o solo (SANTOS, 2015).

Neste trabalho consideraremos o solo como plano e não escorregadio e desta forma a modelagem é feita através do acionamento diferencial de um veículo de duas rodas, sendo para esse modelo uma boa aproximação adotada.

5 DESENVOLVIMENTO

O robô em estudo tem como principal objetivo seguir uma trajetória descrita por uma faixa no solo e para tal foi desenvolvido uma plataforma robótica. Foi utilizado um Arduino para o monitoramento dos sensores e acionamento dos motores; uma RaspberryPi para o software de visão computacional e a comunicação entre os dois módulos foi feita usando comunicação serial.

A plataforma Arduino foi utilizada, devido à sua simplicidade para programação e por conter bibliotecas já desenvolvidas que ajudam a implementação do trabalho. E o microcontrolador foi dedicado completamente ao monitoramento dos *encoders* para que todos os pulsos fossem computados.

A Raspberry Pi é um mini-computador foi utilizada para os processos de alto nível do sistema robótico como o processamento da imagem e o controle de trajetória a partir da imagem tratada. É uma plataforma de caráter aberto que suporta diversas distribuições Linux, possui baixo custo e é muito popular entre engenheiros, estudantes e hobbistas; sendo então uma boa opção para projetos de sistemas embarcados.

De acordo com Romero (2014) como visto anteriormente, a estratégia de controle para o controle de trajetória é dividida em quatro níveis, ilustrada na Figura 5. Os dois níveis mais altos que envolvem o planejamento da trajetória e o controle cinemático foram implementados na Raspberry. Os níveis 2 e 1 são, respectivamente, controle dinâmico e controle dos subsistemas (monitoramento dos *encoders*). Esses níveis foram implementados no Arduino.

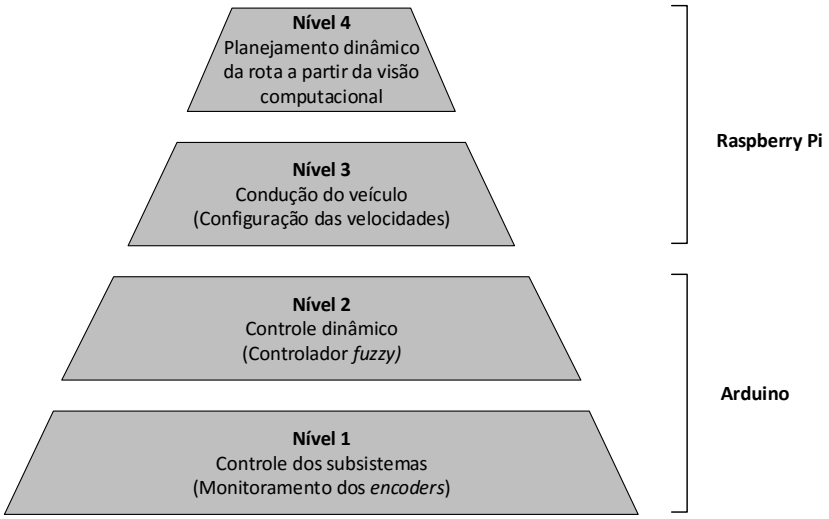


Figura 5. Pirâmide dos níveis da estratégia de controle do robô.

Diante das funcionalidades expostas acima, neste capítulo teremos a descrição do sistema que será dividido em *hardware* e *software*.

5.1 HARDWARE

5.1.1 ESTRUTURA MECÂNICA

A parte mecânica do robô possui dois motores independentes e cada um possui um *encoder* de quadratura associado. Os motores têm acionamentos independentes.

Foi necessária a confecção de duas placas de acrílico para aumentar a área do chassi afim de se colocar os componentes. A primeira placa foi parafusada na base da estrutura do robô e a segunda foi instalada sobre separadores.

No primeiro estágio foi parafusado a plataforma do Arduino, o módulo da Ponte H L298N e havia um espaço na placa por onde todos os fios dos *encoders* e dos motores subiam. No segundo estágio ficou a RaspberryPi, a câmera e as células da bateria.

A tensão máxima dos motores é de 7,2V e a corrente de rotor bloqueado é de 2,5A. Segundo o fabricante é possível atingir a velocidade de 1km/h.

A plataforma robótica desenvolvida é ilustrada na Figura 6.

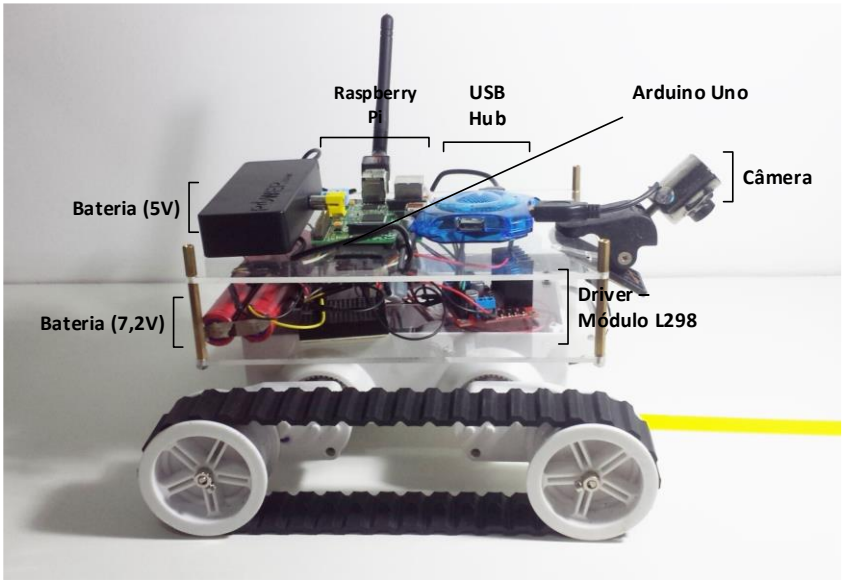


Figura 6. Plataforma robótica desenvolvida.

5.1.2 ENCODERS

Os *encoders*, também chamados de odômetros, são muito populares na robótica móvel. Eles são sensores proprioreceptivos, isto é, realizam medidas de um parâmetro interno de um robô. No caso,a medição da posição e da velocidade das rodas. Em curtas distâncias o valor calculado é preciso, mas geralmente existem fontes de erro como a derrapagem e a modelagem errada ou incompleta do sistema.

O *encoder* é constituído por um fotoemissor, fotorreceptor e um disco acoplado ao eixo da roda. O disco é seccionado em áreas opacas e transparentes em que periodicamente a luz é interrompida gerando um sinal relacionado a velocidade e a posição da roda.

O odômetro utilizado possui dois canais e há um defasamento entre os dois sinais gerados que possibilita saber a direção do movimento. O tratamento e a análise destes sensores são feitos pelo Arduino como será exposto mais à frente.

Figura 7. *Encoder* acoplado ao chassi do robô.

O *encoder* e seu circuito de condicionamento utilizados, ilustrados na Figura 7, apresenta 1000 revoluções a cada 3 voltas indicando uma boa resolução. E por ser de quadratura é possível saber o sentido de rotação do motor. O chassi que foi adquirido possuía quatro *encoders* e dois deles foram instalados em outro chassi igual que não possuía sensores, resultando em dois possíveis robôs operantes.

5.1.3 DRIVER DOS MOTORES

O circuito utilizado para o acionamento e o controle dos motores foi de uma ponte H de transistores bipolares que pode determinar o sentido da corrente e a polaridade da tensão.

O chaveamento de componentes eletrônicos através do PWM (*Pulse Width Modulation*) definido pelo Arduino determina a polaridade e o módulo da tensão no motor. Então sua principal função é o controle de velocidade.

Na Figura 8 pode-se observar um esquema do funcionamento da Ponte H. São quatro chaves que devem ser acionadas de forma binarizada, isto é, quando as chaves Ch1 e Ch4 estão fechadas as chaves Ch2 e Ch3 devem estar abertas.

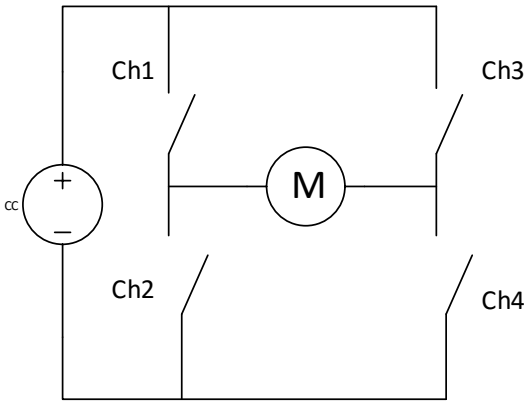


Figura 8. Modelo da Ponte H.

Acionando o conjunto Ch1 e Ch4 o terminal esquerdo fica com a tensão mais positiva que o direito fazendo a corrente fluir da esquerda para direita (adotando o fluxo de corrente convencional) e o motor gira para um sentido. Quando é feito o acionamento das chaves Ch2 e Ch3 o terminal direito fica mais positivo e a corrente flui da direita para a esquerda fazendo o motor girar para o outro sentido.

As chaves Ch1 e Ch2 não podem ser fechadas simultaneamente assim como as chaves Ch3 e Ch4 porque se causaria um curto-circuito na fonte de alimentação. Além disso, se for feito o acionamento de todas as quatro chaves é provocado um curto nos terminais do motor e a máquina CC.

No robô foi utilizado o módulo Ponte H L298N, ilustrado na Figura 9. Este módulo é projetado para acionar e controlar cargas indutivas, principalmente motores de corrente contínua. Sendo possível definir o sentido de rotação e controlar a velocidade a partir dos pinos PWM do Arduino.

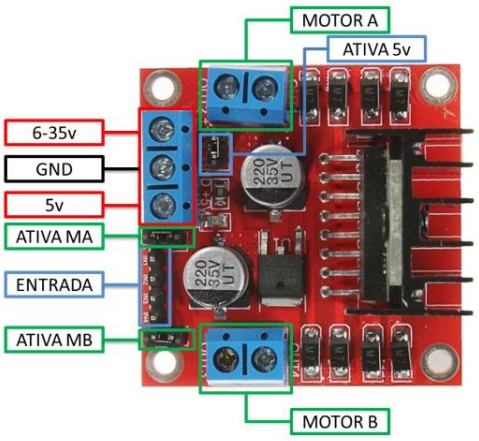


Figura 9. Módulo Ponte H L298N utilizado no robô.
Disponível em: < http://blog.filipeflop.com/motores-e-servos/motor-dc-arduino-ponte-h-l298n.html>.
Acesso em 16/04/2016.

Na Figura 9, as indicações Motor A e Motor B são as saídas conectadas aos pinos dos motores. Os pinos responsáveis pelo controle de velocidade são AtivaMA e AtivaMB que recebem os comandos PWM do Arduino. Se estes pinos estiverem com um *jumper,* não haverá controle de velocidade, pois estão internamente ligados aos 5V do módulo. E desta maneira a velocidade é sempre a máxima.

As entradas são indicadas por IN1, IN2, IN3 e IN4, em que as duas primeiras são para o motor da esquerda e as duas últimas para o motor da direita. E são elas que determinam a direção e o sentido do robô de acordo com a Tabela 1.

Tabela 1 – Acionamento dos motores e direção do robô.

Direção	*IN1*	*IN2*	*IN3*	*IN4*
Frente	5V	GND	5V	GND
Trás	GND	5V	GND	5V
Direita	5V	GND	GND	5V
Esquerda	GND	5V	5V	GND
Freio	GND	GND	GND	GND

No pino descrito por (6-35V) é conectada a fonte de alimentação externa que foram duas células de bateria de íons de lítio em série totalizando uma tensão de 7,2V e com uma carga nominal de 3,2Ah.

Este módulo possui um regulador de tensão interno que disponibiliza para o usuário uma saída de 5V quando o driver está operando entre 6 e 35V, caso se queira alimentar algum dispositivo eletrônico externo. Nesta configuração o *jumper* do Ativa5V deve ser mantido. Se a placa estiver operando entre 4 e 5,5V o pino de 5V é uma entrada que pode ser alimentada com a saída de 5V do Arduino e o *jumper* deve ser retirado.

5.2 SOFTWARE

O software foi desenvolvido em C++ usando a biblioteca OpenCV. O ambiente de desenvolvimento utilizado foi o QtCreator e depois era passado para a Raspberry Pi.

Foi desenvolvida a Classe *Capture* que tinha como objetivo configurar, abrir e fechar a câmera padrão. Também há o método da calibração para selecionar a cor da

faixa utilizada e o método do processamento da imagem, como ilustrado no diagrama da Figura 10. Os atributos da Classe são o número de linhas e colunas da captura da câmera.

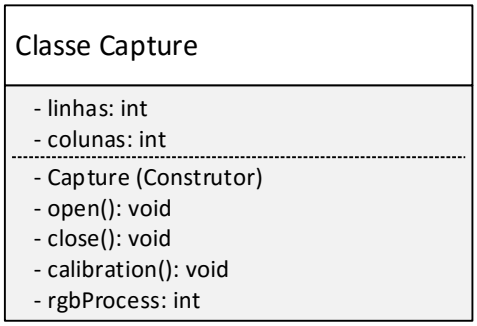


Figura 10. Diagrama da Classe *Capture* desenvolvida.

5.2.1 CLASSE *CAPTURE*

Foi desenvolvida uma classe em C++ chamada *Capture* com os métodos que serão mostrados a seguir.

- Construtor *Capture*: Configura a câmera padrão. O número de linhas e colunas pode ser dado como entrada pelo usuário de acordo com a câmera utilizada. Se estes parâmetros não forem passados o padrão utilizado é 640x480.
- Método *open:* Abre a câmera padrão. Se houver algum erro ou não houver a detecção da câmera uma mensagem é mostrada na saída padrão de erro.
- Método *close:* Fecha a câmera padrão.
- Método *rgbProcess*: Todo o processamento que será descrito no tópico seguinte é feito por este método que recebe como parâmetro de entrada um *frame* e retorna a referência, isto é, a diferença do ponto mapeado e o meio da tela dada em pixels. O método possui este nome, visto que o processo é feito sobre uma imagem de origem em RGB.
- Método *calibration*: O sistema HSV é muito utilizado para seleção de cores. É possível escolher a cor apenas com os três parâmetros do sistema, mas devido às variações de iluminação do ambiente a seleção é feita através de uma faixa de valores, denominados *Hmin*, *Hmáx*, *Smin*, *Smáx*, *Vmin e Vmáx*. Esses parâmetros são definidos em barras de rolamento, chamadas de *trackbars.*

5.2.2 TRATAMENTO DA IMAGEM

O processamento da imagem é feito com o objetivo de diferenciar bem a faixa no chão do resto do ambiente. Para que havendo um bom destacamento seja possível extrair as informações necessárias para o controle. Todo o processamento foi feito na RaspberryPi usando o método *rgbProcess* citado anteriormente. Na Figura 11 tem-se o fluxograma do tratamento da imagem capturada.

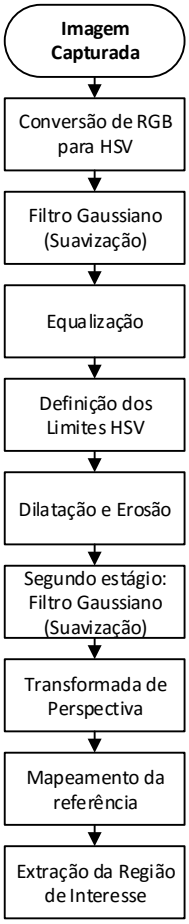
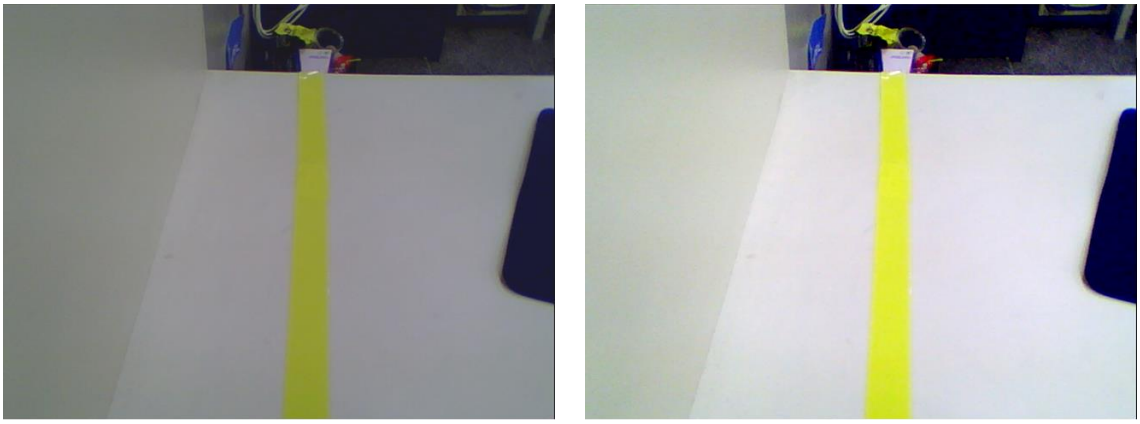


Figura 11. Fluxograma do tratamento da imagem capturada.

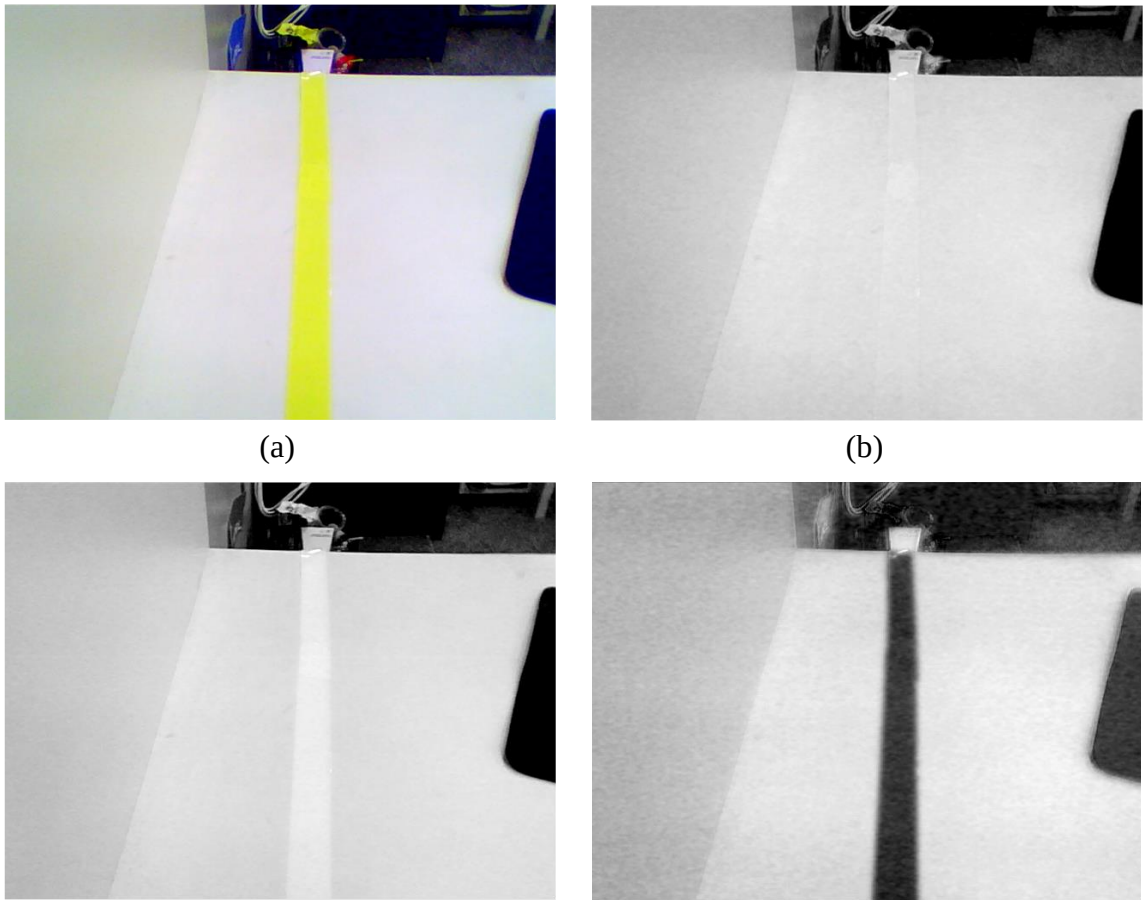
Um dos primeiros filtros usado no processamento de imagens é a equalização. Basicamente consiste na alteração das amplitudes para abranger toda a gama de reprodução de luminosidade. Na Figura 12 é apresentada a etapa de equalização da imagem capturada.



(a) (b)

FIGURA 12 – a) Imagem original b) Imagem equalizada

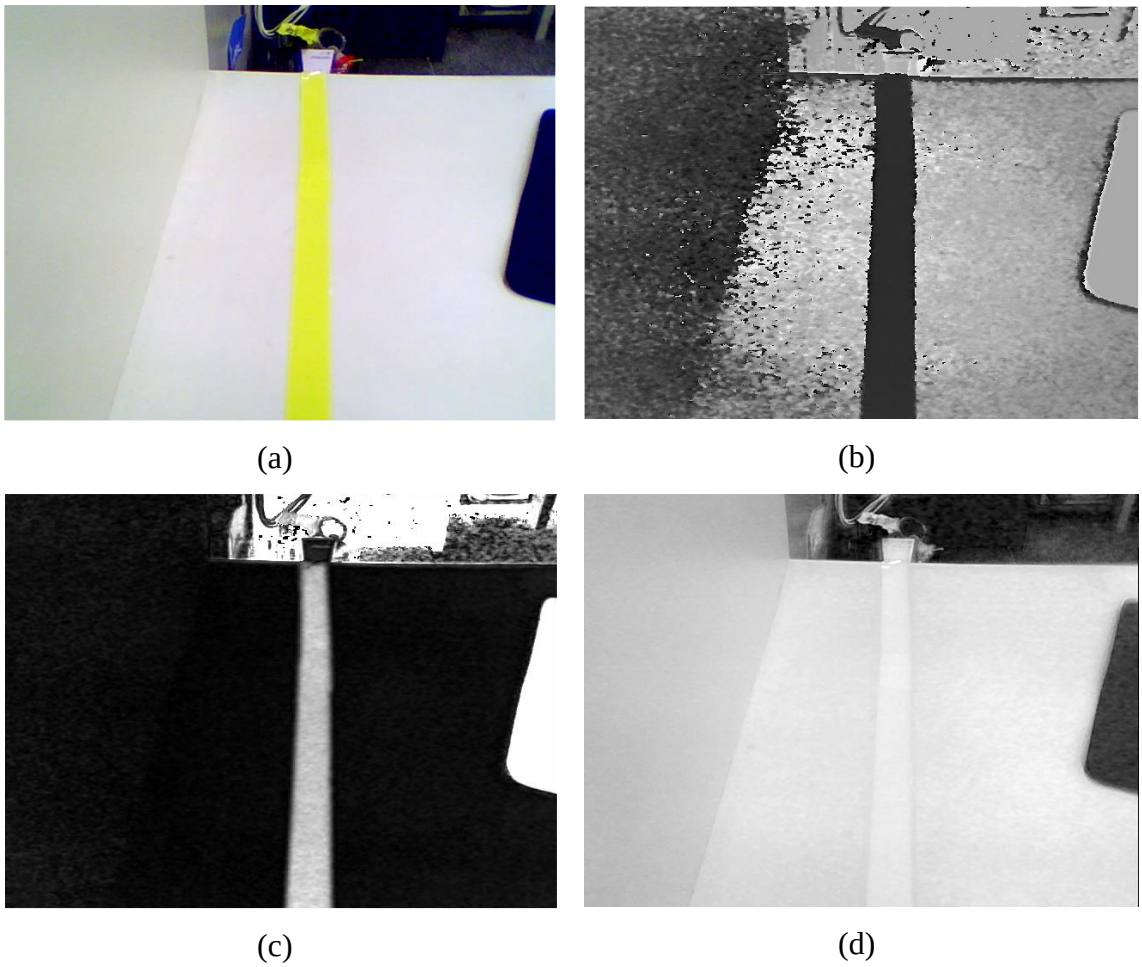
Existem diversas formas de representar as cores e a imagem capturada está no sistema RGB e as componentes são ilustradas na Figura 13.



(a) (b)

(c) (d)

Figura 13. A) imagem original equalizada b) Componente r (vermelho) c) Componente G (verde) d) Componente B (azul)



(a) (b)

(c) (d)

FIGURA 14 – Componentes da representação HSV a) Imagem original equalizada b) Componente H (cada intensidade de cinza representa uma cor distinta) c) Componente S (representa o grau de saturação ou pureza da cor) d) Componente V (representa a intensidade de luminância)

O terceiro bloco é a filtragem da imagem usando um filtro gaussiano com o objetivo de reduzir o ruído da imagem. O efeito visual desta técnica é a suavização da cena aparentando estar borrada, ilustrado na Figura 15. Sendo então a principal vantagem a diminuição do ruído em ambientes mais complexos.

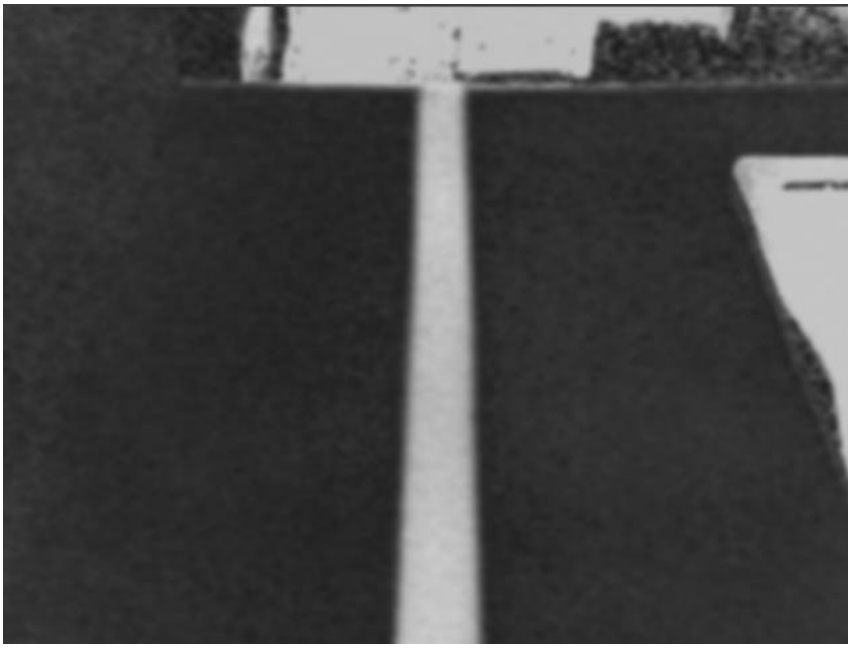


Figura 15. Imagem após a utilização do filtro gaussiano.

Uma grande vantagem da classe criada para o robô é que é possível se fazer a calibração da cor da faixa através da definição dos limites máximos e mínimos do sistema HSV, como mostrado no tópico anterior. A faixa adotada nos testes foi da cor amarela. E o método utilizado foi o *inRange* que recebe os seis parâmetros (*Hmin, Hmáx, Smin, Smáx, Vmin e Vmáx*) baseados na calibração previamente feita. A saída é a imagem binarizada com 1 para os valores dentro da faixa e 0 para os valores fora da faixa predeterminada, ilustrado na Figura 16.

Figura 16. Imagem binarizada baseada nos parâmetros da calibração.

A dilatação e a erosão são técnicas que visam reduzir o ruído como ilustrado na Figura 17, eliminando buracos e pontas nas imagens processadas, através de um processo de convolução que avalia disparidades de conjuntos de pixels em relação à sua circunvizinhança. Isso possibilita um isolamento maior da região a ser tratada. No caso deste trabalho, a região principal é a faixa no solo.

Figura 17. Imagem após o uso dos processos de dilatação e erosão.

A técnica de equalização utilizada é para aumentar o contraste entre as duas áreas trabalhadas no processamento: a faixa e o chão. O algoritmo por trás consiste em quantificar a intensidade de todos os pixels em um histograma, normalizar o brilho e realçar o contraste da cena, exibido na Figura 18.

Figura 18. Imagem após a equalização.

A transformada de perspectiva, ilustrada na Figura 19, foi utilizada para que a imagem no controle seja vista de cima e não do ponto de vista da câmera, pois desta maneira teremos mais exatidão na coleta dos pontos para a estratégia de controle. Essa técnica depende da altura da câmera e da angulação da lente em relação ao solo, sendo necessária uma calibração. Primeiramente são coletados quatro pontos da imagem capturada e depois são definidos quatro pontos que serão nosso destino de interesse. Esses conjuntos de pontos são entrada de uma função que gera uma matriz de transformação de perspectiva 3x3, pois é uma transformada linear no espaço, que é aplicada então sobre a imagem tratada.

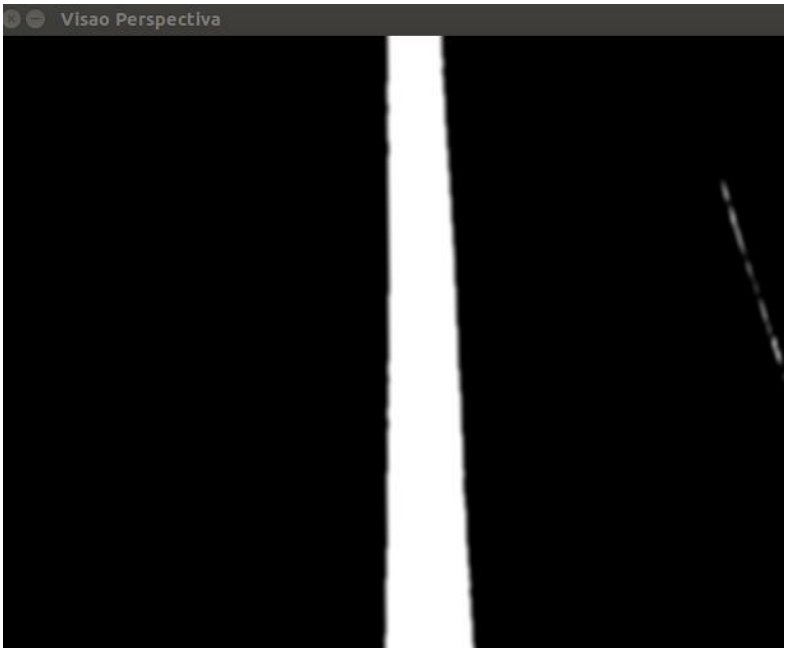


Figura 19. Aplicação da transformada de perspectiva.

Por fim, a linha de referência é o meio da tela e os valores de referência calculados são dados pela diferença entre o valor da ordenada da linha de referência e o valor do ponto mapeado. A imagem é recortada para uma área menor que é chamada de região de interesse. O resultado final é ilustrado na Figura 20 em que se pode ver a faixa destacada do restante do ambiente, a linha de referência e o ponto mapeado que servirá de referência para o controlador.

Figura 20. Imagem final que será utilizada para o controlador.

5.2.3 CALIBRAÇÃO DO SISTEMA

A calibração deve ser o primeiro procedimento adotado quando se for utilizar o sistema robótico. O usuário avalia e seleciona a cor de interesse a partir da manipulação na interface de calibração, ilustrada na Figura 21. Desta maneira, é possível haver variações na cor da faixa e na iluminação do ambiente que o sistema ainda funcionará corretamente desde que tenha sido calibrado de forma adequada.

O método desenvolvido é o *calibration* em que o usuário tem acesso a duas janelas de visualização: a imagem original no sistema RGB ilustrada na Figura 22 e a imagem após o tratamento ilustrada na Figura 23.

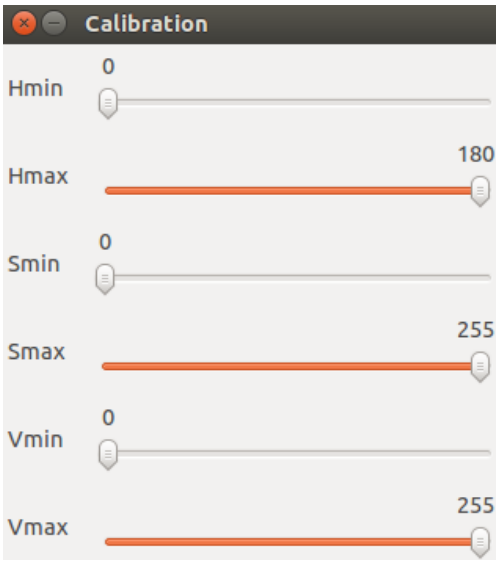


Figura 21. Barras de calibração do sistema HSV para seleção da cor da faixa.

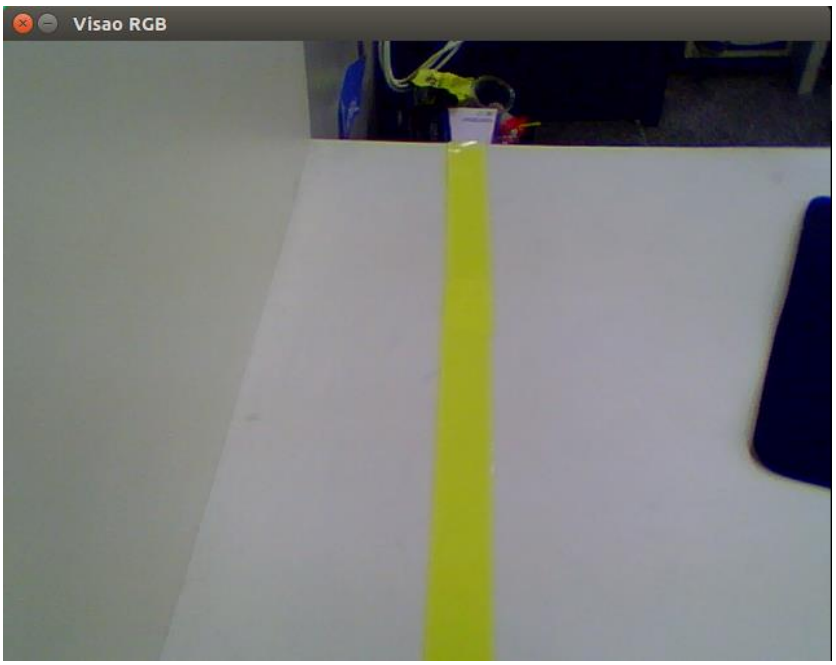


Figura 22. Imagem original em RGB.

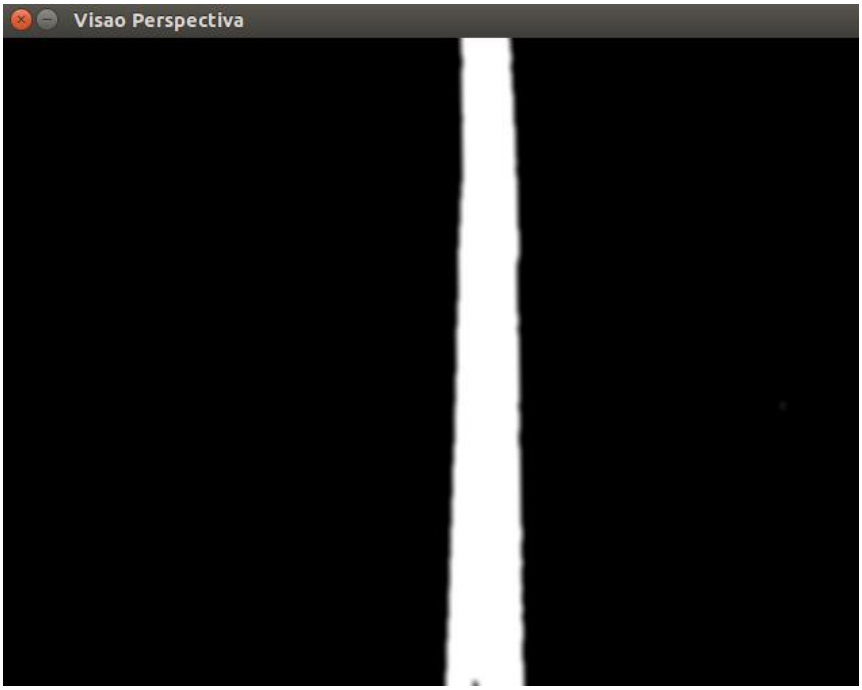


Figura 23. Imagem após o processamento.

As imagens ilustradas foram coletadas com a câmera em cima do robô a uma altura de 16cm da superfície. É interessante notar que a imagem original apresenta diversas cores, inclusive próximas da cor escolhida para a faixa, mas a calibração e o processamento foram feitos de forma adequada e desta maneira apenas a faixa é identificada.

5.2.4 CONTROLE DA DIREÇÃO PELA VISÃO COMPUTACIONAL

A linha de referência se localiza no meio da imagem de interesse e quando o ponto mapeado estiver sobre ela ou próximo a ela o robô deverá seguir em frente. Se o ponto estiver à esquerda da referência, a direção a ser tomada será a esquerda então a roda direita deverá ter uma velocidade maior que a da esquerda. Da mesma maneira que quando o ponto mapeado estiver à direita, o robô deverá girar para a direita fazendo com que a roda da esquerda tenha uma velocidade maior que a da direita.

O método *rgbProcess* retorna a diferença *d* entre a linha média e o ponto mapeado e é dada em pixels. A Tabela 2 exibe os intervalos e os valores das velocidades que foram adotados quando o processamento foi feito no computador.

Tabela 2 – Intervalos dos erros e as velocidades associadas.

Intervalo	*Direção*	*Velocidade da esquerda (em cm/s)*	*Velocidade da direita (em cm/s)*
$-10 \leq d \leq 10$	Frente	14	14
$10 < d \leq 50$	Esquerda	13	20
$50 < d \leq 150$	Esquerda	10	22
$-50 \leq d < -10$	Direita	20	13
$-150 \leq d < -50$	Direita	22	10

Os valores das velocidades foram escolhidos de acordo com o comportamento do sistema. Quando o processamento da imagem é feito no computador podem ser colocadas velocidades maiores porque as respostas são mais rápidas. Já na Raspberry a velocidade não pode ser tão alta porque o sistema não responde em tempo hábil como será analisado mais à frente. A Figura 24 ilustra as faixas na região de interesse do robô.

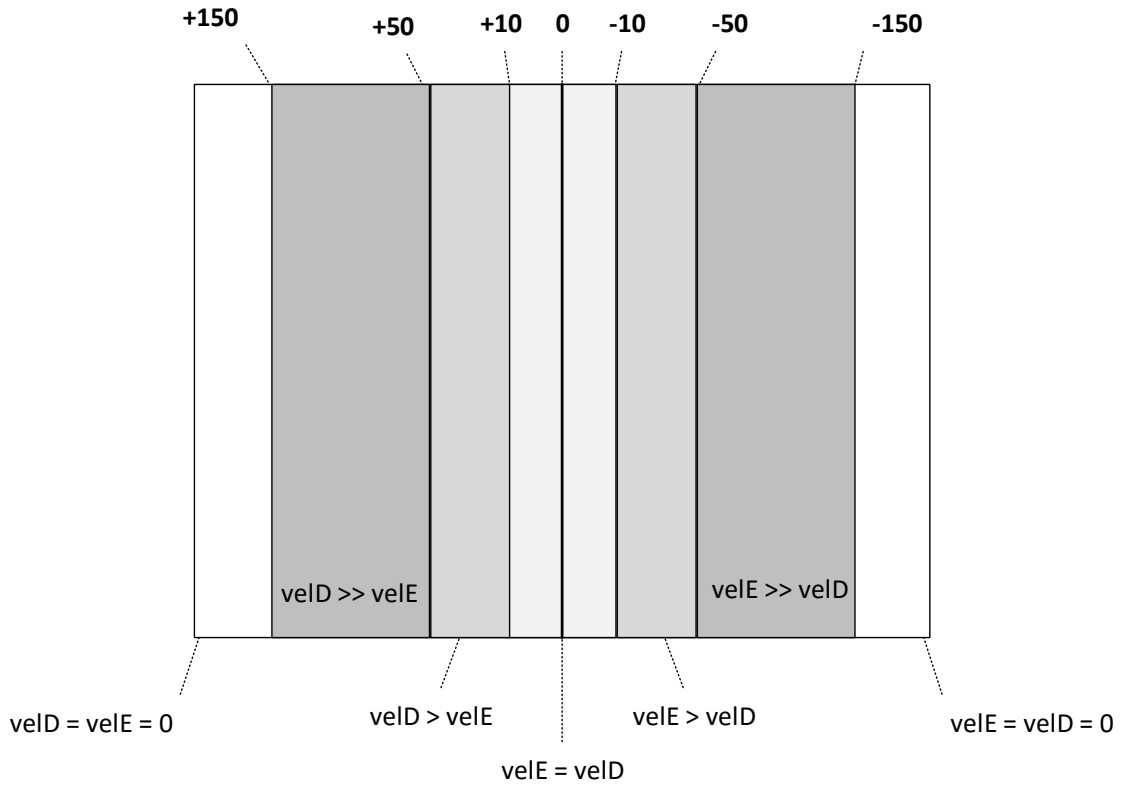


Figura 24. Faixas de operação na região de interesse do robô.

Após ser feito o processamento e identificada a faixa de operação exibe as velocidades de referência são enviadas pela serial ao Arduino para ser feito o controle.

5.2.5 MEDIÇÃO DE VELOCIDADE NO ARDUINO

A frequência do sinal do *encoder* é alta e houve uma grande dificuldade para que se pudesse fazer a medição e o cálculo da velocidade, visto que o Arduino não trabalha bem com variáveis *float*. Os cálculos com ponto flutuante são feitos por *software* o que demora um tempo considerável quando se está fazendo um controle. A maior parte dos cálculos foram feitos usando inteiro por ser mais rápido já que é feito por *hardware* no microcontrolador.

Foi usado uma biblioteca nativa do Arduino para contagem dos pulsos de um *encoder* de quadratura. Os dois canais devem ser conectados a dois pinos do Arduino. A documentação da biblioteca indica que existem três opções de conexão:

- Melhor performance: quando se usa dois pinos de interrupção.
- Boa performance: quando um dos sinais está em um pino de interrupção e outro não.
- Baixa performance: os dois sinais são conectados a pinos que não são de interrupção.

No projeto foi utilizado um Arduino Uno que só possui dois pinos de interrupção, portanto cada *encoder* terá um pino de interrupção e desta forma ambos apresentam uma boa performance, de acordo com a documentação.

Para uma performance ainda melhor do sistema de medição e controle, a taxa de transferência foi de 115200 bps. Os primeiros testes foram feitos com 9600 bps, mas esta baixa velocidade comprometia o desempenho do *encoder*, especialmente quando precisava-se imprimir valores em um terminal serial para depuração.

5.2.6 CONTROLE DE VELOCIDADE

O controle só pode ser feito quando se é possível medir a velocidade atual. A Raspberry envia via serial as velocidades de referência (vel_refDe vel_refE) e as velocidades atuais (vel_atualD e vel_atualE) são medidas pelo Arduino a partir dos *encoders*, desta forma temos o erro da direita ($erroD$) e o erro da esquerda ($erroE$) que são dados por:

$$erroD = vel\_refD - vel\_atualD \tag{8}$$

$$erroE = vel\_refE - vel\_atualE \tag{9}$$

O fluxograma ilustrado na Figura 25 ilustra os passos para o controle da velocidade. Os dois erros são as entradas do controlador *fuzzy*.

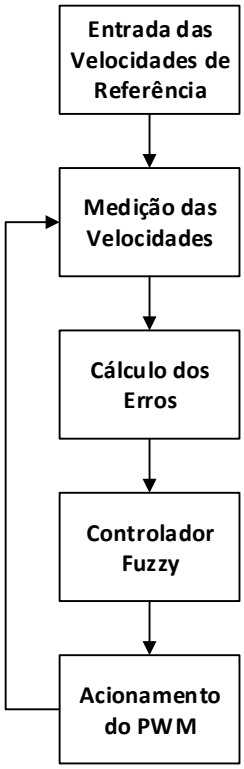


Figura 25. Fluxograma do algoritmo de controle implementado no Arduino.

Foi escolhida como estratégia de controle usar a lógica *fuzzy*, devido sua simplicidade e versatilidade, visto que pode ser ajustado para se atingir um ponto ótimo no controle. O controlador *fuzzy* foi desenvolvido no Matlab usando a caixa de ferramenta denominada *FIS Editor* e pode ser observado na Figura 26.

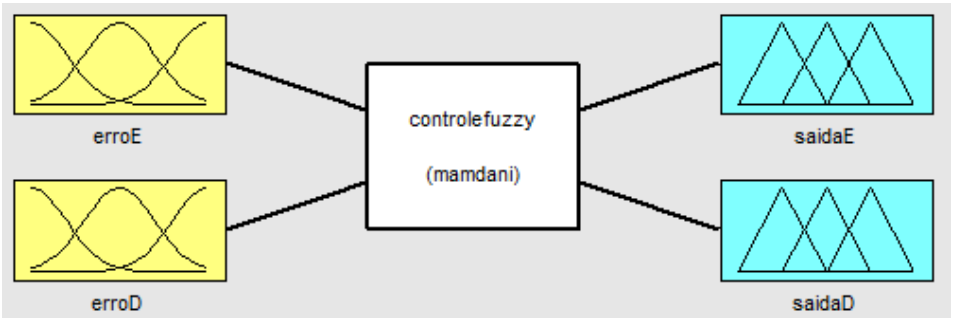


Figura 26. Esquema do controlador fuzzy desenvolvido no Matlab.

As duas entradas apresentam a mesma faixa de valores que é de -26 a 26. Isto é, são os valores extremos que os erros podem assumir considerando que a velocidade máxima é 26cm/s. Essa velocidade foi obtida experimentalmente setando o PWM em 100%. Foram feitos três conjuntos *fuzzy* que são ilustrados na Figura 27 e representam a entrada da esquerda.

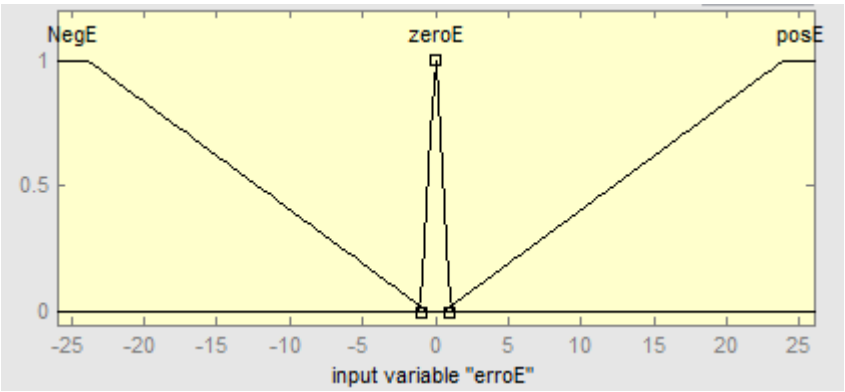


Figura 27. Conjuntos fuzzy da entrada do controlador.

Esses conjuntos foram ajustados de forma heurística, isto é, foram avaliados de forma empírica afim de encontrar os melhores valores. Na Figura 27 foi representado o conjunto da esquerda, mas o bloco da direita possui os mesmos parâmetros.

Os conjuntos de saída são ilustrados na Figura 28 e os valores de saída são somados ao valor do PWM atual para que haja a mudança de velocidade de acordo com a referência enviada pela Raspberry ou computador.

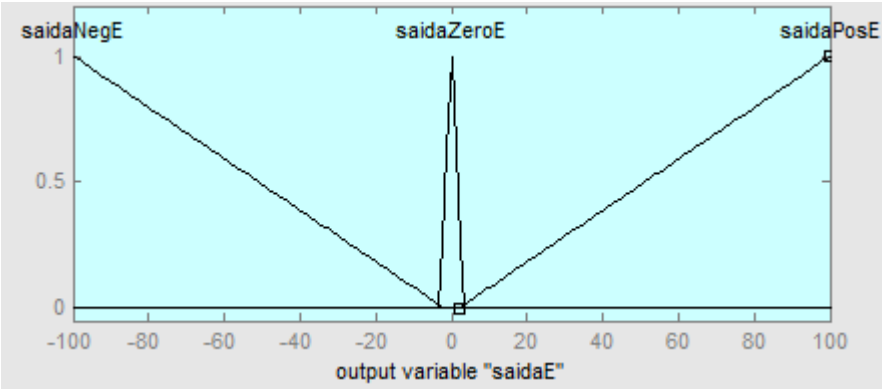


Figura 28. Conjuntos de saída do controlador *fuzzy*.

As regras definidas foram bem simples e o controlador agiu como um controlador proporcional, isto é, quanto maior o erro maior seria a saída para o PWM para que fosse corrigido. As regras foram:

Se o erro da esquerda for negativo, a saída da esquerda é negativa.

Se o erro da esquerda for zero, a saída da esquerda é zero.

Se o erro da esquerda for positivo, a saída da esquerda é positiva.
Se o erro da direita for negativo, a saída da direita é negativa.
Se o erro da direita for zero, a saída da direita é zero.
Se o erro da direita for positivo, a saída da direita é positiva.

Após o controlador ser desenvolvido foi utilizado uma ferramenta disponível *online* no site MakeProto que converte o arquivo no formato .fis da caixa de ferramentas do Matlab para um código voltado para Arduino. O código apresentado foi modificado para atender às especificações do projeto, pois o resultado proposto pela ferramenta tem como entrada um pino analógico do Arduino, mas foi substituído pelo erro de cada conjunto de rodas.

6 ANÁLISE DOS RESULTADOS

6.1 ANÁLISE E TESTES NO TRATAMENTO DE IMAGEM

No sistema há diversas fontes de ruído como as condições climáticas, a iluminação do ambiente, a câmera utilizada e a posição relativa entre a faixa de estudo e a câmera. Então se faz necessário que seja feita a calibração para cada ambiente mesmo mantendo a cor da faixa, pois a iluminação local sempre varia muito.

Uma vez feita a calibração, o sistema responde satisfatoriamente às variações que podem ocorrer como a sombra que o próprio robô, a movimentação de pessoas na frente da fonte de luz e até a diminuição da iluminação local, como será mostrado a seguir.

Foram feitos alguns testes para mostrar as respostas do algoritmo perante às mudanças. Os testes foram feitos usando o computador para fazer o processamento da imagem. O ambiente de teste contava com uma lâmpada no teto e um abajur que foi colocado em duas posições: em frente ao robô e atrás; para se analisar a influência da sombra. Também foram feitos os testes com todas as luzes apagadas e com todas as luzes apagadas e os leds da câmera acesos. Os testes foram feitos no período da noite.

6.1.1 CONDIÇÕES INICIAIS

Para o uso da plataforma robótica em um novo ambiente é sempre necessário se fazer a calibração para se obter os melhores resultados. No ambiente descrito acima os valores da calibração são exibidos na Figura 29.

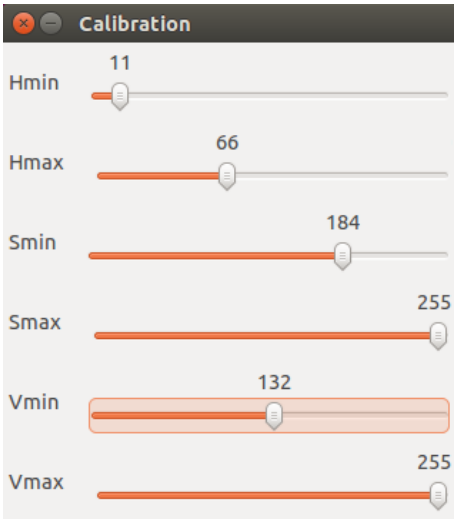


Figura 29. Interface para calibração e aquisição dos valores para definição da faixa de cor selecionada no sistema HSV.

A configuração inicial contava com a luz do teto acesa e o abajur em posição frontal ao robô, como ilustrado no esquema da Figura 30.

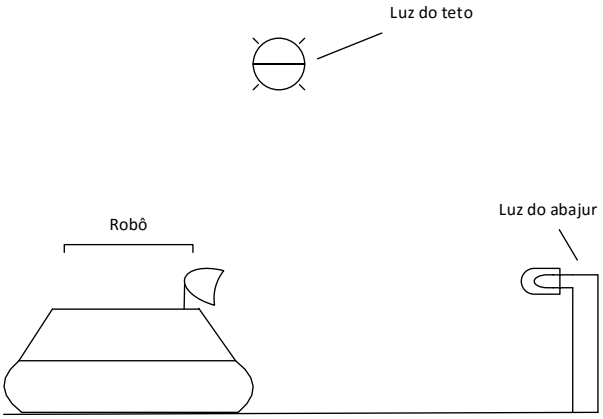


Figura 30. Esquema da condição inicial de teste.

Após feita a calibração é rodado o software do processamento e a imagem resultante após o tratamento é ilustrada na Figura 31.

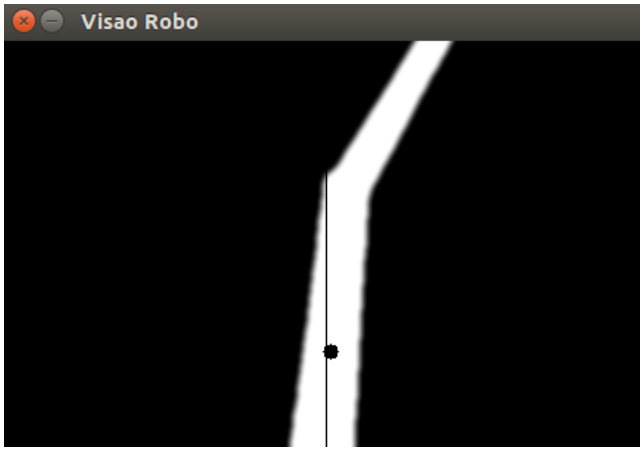


Figura 31. Imagem de saída após calibração.

6.1.2 TESTE 1: LUZ DO TETO ACESA E ABAJUR NA FRENTE APAGADO

Nesta configuração a luz do abajur que estava à frente do robô foi desligada e o resultado é ilustrado na Figura 32. Pode ser notada uma pequena diferença, mas o algoritmo de processamento atua e se torna quase imperceptível a mudança significativa na iluminação. A atuação do sistema de tratamento consiste nos algoritmos utilizados como o de dilatação e erosão que diminuem o ruído existente e o algoritmo da equalização que ressalta o contraste entre a faixa e o chão.

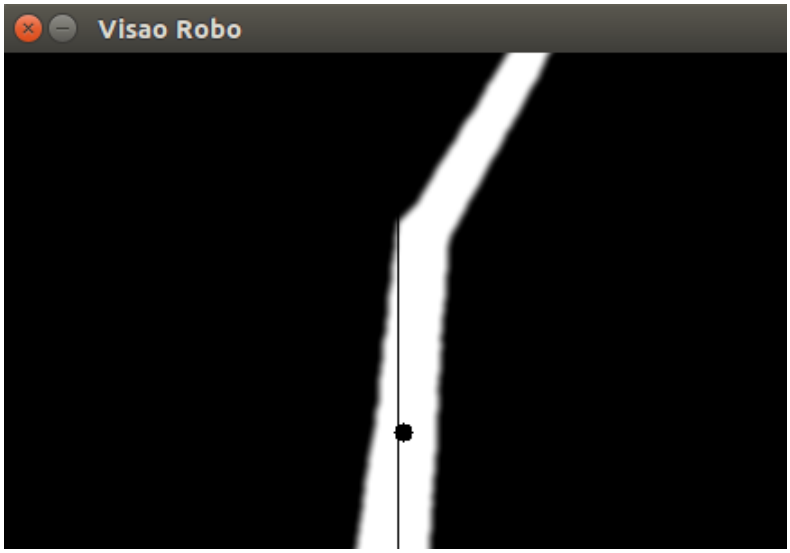


Figura 32. Teste com a luz do teto acesa e luz em posição frontal apagada.

6.1.3 TESTE2: LUZ DO TETO E LUZ FRONTAL APAGADAS E LEDS DA CÂMERA LIGADOS

A resposta desse teste foi muito satisfatória, pois o sistema de tratamento da imagem atuou de forma muito positiva e manteve um resultado eficaz, principalmente

na região do ponto de referência. Pode-se ver pela Figura 33 que a região mais afetada foi a superior que é onde a luz da câmera não alcança da mesma forma.

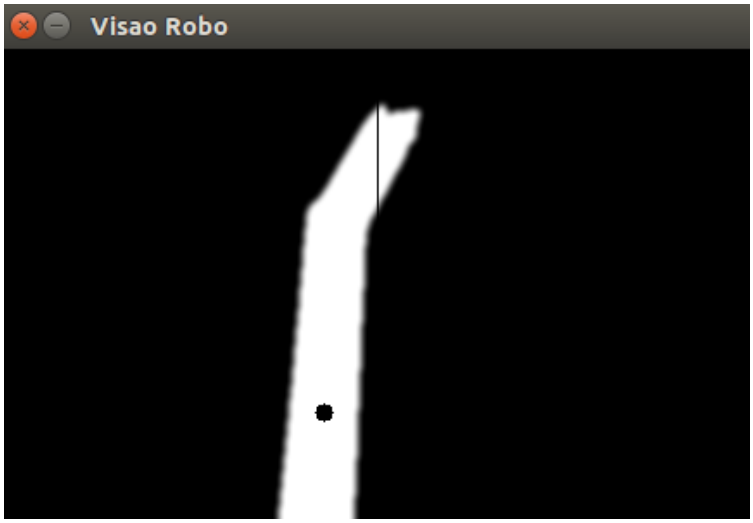


Figura 33. Teste com as duas luzes apagadas e leds da câmera acesos.

6.1.4 TESTE 3: LUZ DO TETO ACESA E LUZ DO ABAJUR EM POSIÇÃO ANTERIOR AO ROBÔ

O principal motivo para esse teste foi avaliar se a sombra do robô afetaria o sistema. A luz secundária ficou em posição anterior ao robô, ilustrado na Figura 34.

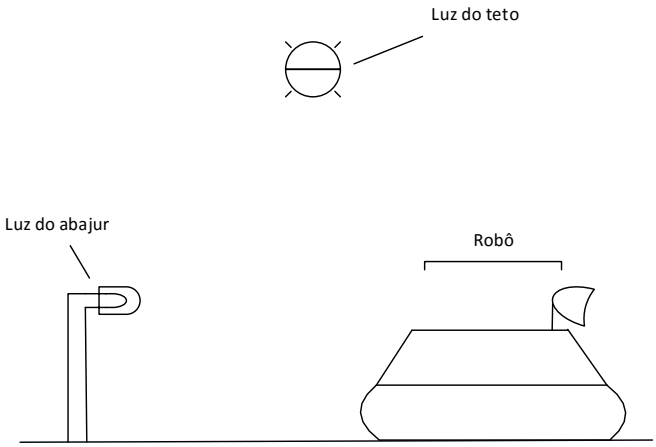


Figura 34. Esquema do teste com a luz secundária anterior ao robô.

Nota-se que não houve mudança significativa na imagem resultante exibida na Figura 35.

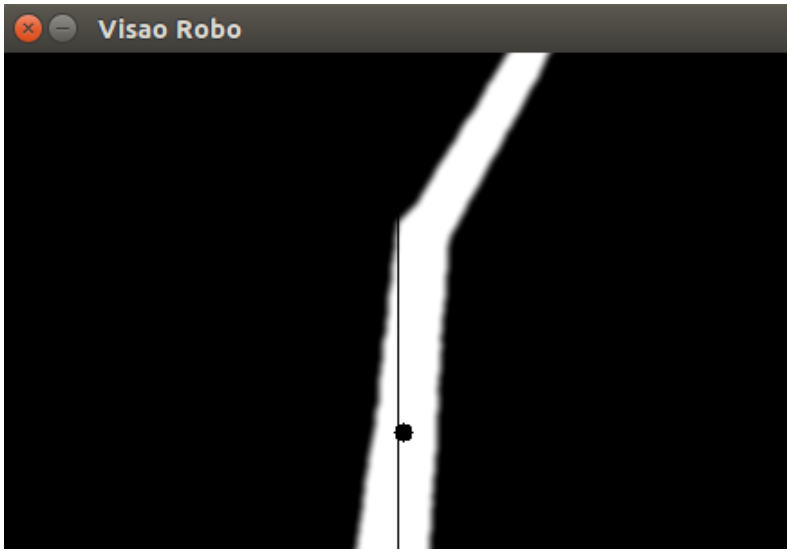


Figura 35. Teste com a luz do teto acesa e luz em posição anterior apagada.

Os testes com uma luz antes e depois do robô foram feitos porque foi uma dificuldade encontrada durante a implementação do projeto, pois a sombra do robô alterava a iluminação na região e gerava ruído principalmente na zona inferior da imagem. Mas com o aperfeiçoamento do algoritmo de tratamento pôde-se obter bons resultados.

Os testes mostraram que o sistema foi eficaz mesmo para variações significativas na iluminação do ambiente, desde que seja feita uma calibração adequada. Sendo então necessário que o robô percorra a trajetória dentro dos limites de operação estabelecidos pelos parâmetros-limites HSV (*Hmin, Hmáx, Smin, Smáx, Vmin e Vmáx*).

6.2 DIFERENÇA DA RASPBERRY PARA O COMPUTADOR

A Raspberry utilizada foi a da primeira geração Modelo B e não correspondeu bem, pois a imagem ficava lenta. E ocasionalmente a imagem ficava escura ou entrecortada fazendo com que fossem enviadas velocidades de referência incoerentes para o controle dinâmico implementado no Arduino.

Quando o processamento da imagem é feito no computador as mudanças de direção e velocidade ocorrem de maneira mais rápida e o funcionamento do robô é bem mais eficaz.

As fotos coletadas foram do processamento no computador, mas os resultados obtidos na Raspberry para a calibração e o tratamento da imagem foram os mesmos. A

diferença era no controle dinâmico, visto que o sistema não respondia rápido às mudanças de posição e velocidade.

7 CONCLUSÃO

Este trabalho teve como uma das principais finalidades um conhecimento maior sobre o campo da visão computacional e processamento de imagens, sendo o primeiro passo de um projeto que ainda tem muito a crescer e se desenvolver em trabalhos futuros.

O grande fator enriquecedor deste trabalho foi a multidisciplinaridade, em que um conjunto de disciplinas foram trabalhadas em conjunto e em alguns momentos simultaneamente para atingir um objetivo. O trabalho contou com sistemas de aquisição de dados, técnicas de controle inteligente, programação voltada a objetos, mecânica e cinemática, eletrônica de potência para o acionamento e controle dos motores, programação em Linux para o sistema embarcado e o estudo de visão computacional que não é visto no durante o curso de Engenharia Elétrica.

Todo o projeto será colocado em um site, ainda em desenvolvimento, com todo o material necessário para confecção. Os códigos serão disponibilizados para que se possa haver um aperfeiçoamento e desenvolvimento dos algoritmos utilizados visando deixar o projeto cada vez mais robusto.

Como visto na análise dos resultados a iluminação ambiente influencia na imagem se não for feita uma boa calibração. Mas nos testes foram feitas variações significativas e o sistema ainda se mostrou eficiente. A dependência de uma calibração sempre que se mudar o ambiente é a grande limitação do sistema, pois na prática o robô poderia percorrer locais variados e sair da zona de operação estabelecida.

A Raspberry Pi 1 Modelo B não se mostrou uma boa solução para tal projeto devido à sua baixa capacidade de processamento que o sistema requeria. E desta forma o melhor desempenho da plataforma robótica foi quando o processamento da imagem e envio das velocidades de referência foram feitos no computador.

Os trabalhos futuros serão voltados para implementar uma visão estéreo, em que o sistema é alimentado por duas fontes de captura com posições distintas que geram conteúdos ligeiramente diferentes. Sendo assim possível fazer uma reconstrução em 3D do ambiente e evitar obstáculos.

BIBLIOGRAFIA

BRADSKI, G.; KAEHLER, A. *Learning OpenCV*, O'Reilly Media, Inc., 2008.

DUDEK, G.; JENKIN, M. *Computacional principlesof mobile robots*. Cambridge University Press, 2005.

FIGUEREIDO, L. C.; JOTA, F. Introdução ao controle de sistemas não-holonômicos. Revista Controle e Automação, vol. 15, no. 3, pp. 243-268, 2004.

JACOBINA, C. B. Sistemas de Acionamento Estático de Máquina Elétrica. Campina Grande, Paraíba, 2005.

JUNG, C.; OSÓRIO, F.; KELBER, C.; HEINEN, F. Jornada de Atualização em Informática JAI2005. SBC, ch. Computação Embarcada: Projeto e Implementação de Veículos Autônomos Inteligentes, p. 1358-1406, 2005.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. *Digital ImageProcessingusing MATLAB*, Pearson, 2006.

MakeProto. Disponível em <http://www.makeproto.com/projects/fuzzy/matlab_arduino_FIST/index.php> Acesso em 07/05/2016.

MARENGONI, M.; STRINGHINI, D.; *Introdução à Visão Computacional usando OpenCV*. RITA. Vol. XVI, Número 1, 2009.

MURPHY, R. R. *Introduction to AI robotics*. Cambridge: MIT Press, 2000.

NGUYEN, H. G.; MORRELL, J.; *Segway Robotic Mobility Plataform*. Mobile Robots XVII, pp. 27-28.

OLLERO, A. *Robótica – Manipuladores y robotsmóviles*. Barcelona: Marcombo, 2001.

OLLERO, A.; AMIDI, O. "Predictive Path Trackingpf Mobile Robots: applicationtothe CMU Navlab", no *Proc. ofthe IEEE InternationalConferenceonAdvancedRobotics*, Pisa, Itália, p. 1081-1086. 1991.

ROMERO, R.; MEDEIROS, Adelardo. *Robótica Móvel*. 1. Ed. – Rio de Janeiro: LTC, 2014.

SANTOS, Alexsandro José Virgínio dos. *Análise e controle de um veículo robótico tracionado por esteiras.* João Pessoa, 2015.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilisticrobotics.*Cambridge: MIT Press, 2005.

YOUN, I.; TCHAMNA, R.; LEE, S. "Preview Suspension Control For A Full Tracked Vehicle". International Journal of Automotive Tecnology, 15 (3), 399-410; 2014.