Sistema de Medição de Distância Baseado em Visão Computacional Utilizando Laser de Linha

Walber C. Jesus Rocha,
Samuel R. Jesus, João Carlos N. Bittencourt
Universidade Federal do Recôncavo da Bahia Rua Rui Barbosa,
710 - Cruz das Almas, Bahia - Brasil
walber_jesus@hotmail.com, samuelreboucas07@hotmail.com, joaocarlos@ufrb.edu.br

RESUMO

A robótica é uma área de estudo em crescente desenvolvimento, propondo soluções automatizadas para diversas tarefas do cotidiano. Com o advento da robótica móvel, plataformas robóticas passaram a executar movimentos dinâmicos, de forma a possuírem autonomia na realização de determinadas tarefas. Aplicações de visão computacional tem proporcionado à plataformas robóticas a capacidade de navegação e mapeamento, a partir da captura e processamento de imagens. Este trabalho apresenta um sistema de medição de distância até os obstáculos, baseado na detecção de um laser de linha e analisar o desempenho da aplicação. A faixa de medição utilizada durante os testes variou entre 20 cm e 100 cm, apresentando erro médio de 0,68 cm e tempo de resposta aproximadamente 153 ms. O sistema foi implementado na plataforma computacional Raspberry Pi, modelo B+, onde a capacidade de processamento foi aliada ao baixo custo de implementação. Os resultados obtidos enfatizam a acurácia dos métodos analisados, com taxas de erro de até 1,36 % e alto desempenho quando aplicada a plataformas computacionais com baixo poder de processamento.

CCS Concepts

•**Computing methodologies → Artificial intelligence; Computer vision; Computer vision tasks; Vision for robotics;**

Palavras-chave

Telêmetro a laser. Detecção de linhas de laser. Agregação de Pixels. Gradiente de Sobel. Transformada de Hough. Raspberry Pi. Navegação de robôs moveis. Distância através de laser de linha.

ABSTRACT

Robotics is a growing field of study, proposing automated solutions for various os humanity daily tasks. With the advent of mobile robotics, robotic platforms began to perform dynamic movements in order to have autonomy in execution of tasks. Computer vision applications have provided robotic platforms with navigation and mapping capabilities through image processing. This paper presents a distance measurement system, based on the detection of a line laser, using image processing techniques to determine the distance to multiple obstacles and analyze the system performance. The measurement range used during the tests ranged from 20 cm to 100 cm, with an average error of 0.68 cm and response time approximately 153 ms. The system was implemented on the Raspberry Pi, model B+ computer platform, where the processing capacity was combined with the low implementation cost. The results emphasize the accuracy of the analyzed methods, with error rates of up to 1.36 % and high performance when applied to computational platforms with low processing power.

Keywords

Line laser detection. Pixel Aggregation. Gradient of Sobel. Hough Transform. Raspberry Pi. Robot mobile navigation. Distance through line laser.

1. INTRODUÇÃO

A robótica é uma área de estudo em crescente desenvolvimento que, dentre outras coisas, busca proporcionar ao ser humano uma vida mais confortável, propondo soluções para tarefas do seu cotidiano [16]. Com o advento dos robôs móveis, as plataformas robóticas passaram a executar movimentos dinâmicos, de forma a apresentarem autonomia na realização de suas tarefas [12]. Um robô móvel pode ser definido como um autômato mecânico, estruturado sobre uma base não estática, controlada por um sistema computacional capaz de gerir sensores e atuadores, permitindo que o mesmo possa interagir com o ambiente [2].

Para que uma plataforma robótica seja capaz de navegar e realizar operações de mapeamento de forma dinâmica, obter informações sobre o ambiente no qual está inserido é de extrema importância para execução de suas ações. Nesse contexto, as soluções baseadas em Visão Computacional têm buscado proporcionar a uma plataforma robótica a capacidade de perceber o ambiente, compreender o que está sendo observado, realizar ações apropriadas e aprender com a

experiência a fim de melhorar o seu desempenho no futuro [18].

Em geral, sistemas robóticos são constituídos a partir de plataformas computacionais dotadas de baixa capacidade de processamento, tal qual os microcontroladores e/ou microprocessadores de baixo custo. Com o advento dos computadores de placa, a exemplo da Raspberry Pi e Beagleboard, aplicações robóticas ganharam uma nova roupagem, na qual a capacidade de processamento foi aliada ao baixo custo de implementação. Todavia, aplicações de visão computacional, tipicamente, se caracterizam por apresentar alta demanda computacional, uma vez que esse tipo de aplicação costuma realizar operações sobre um grande volume de dados em espaços de tempo com restrições de tempo real.

Nesse contexto, este trabalho apresenta um sistema de medição de distância, baseado na detecção de um feixe de laser de linha a partir da utilização de técnicas de processamento de imagem. Além disso, o presente artigo apresenta uma análise desempenho, baseada no tempo de execução e consumo de memória, de dois métodos computacionais de medição, quando executados na plataforma computacional Raspberry Pi, versão B+.

2. VISÃO COMPUTACIONAL PARA MEDIÇÃO DE DISTÂNCIAS

Nas últimas décadas, a pesquisa em visão computacional evoluiu, abrindo espaço para o surgimento de aplicações de aprendizagem de máquina, física (ótica), matemática (descrição geométrica, estatística e otimização), inteligência artificial, neurobiologia, técnicas de processamento de imagens e robótica (visão do robô) [3]. São exemplos dessas aplicações equipamentos para processamento de imagens médicas, aplicações industriais para controle de qualidade e automação de veículos [11]. No âmbito da robótica, aplicações de visão computacional tem proporcionando a capacidade de navegação e mapeamento dinâmico, a partir da captura e processamento de imagens, disponibilizando informações sobre o ambiente no qual o robô está inserido.

A visão computacional é uma importante ferramenta aplicada a sistemas robóticos. Suas aplicações consistem em receber imagens do ambiente de operação do robô através de uma câmera e processá-las, a fim de extrair parâmetros como identificação de linhas, determinação de posição e orientação [15].

Em geral, métodos de extração de parâmetros em imagens utilizam de reconhecimento de bordas e detecção de linhas. Dentre esses métodos, destacam-se técnicas como Gradiente de Sobel, Transformada de Hough e Threshold [5]. Outros métodos são baseados na detecção dos padrões apresentados nos pixels, como o método de Agregação de Pixels. As seções a seguir destacam os conceitos fundamentais associados a cada um dos métodos empregados no desenvolvimento do sistema de medição.

2.1 Crescimento de Região por Agregação de Pixels

O método de crescimento de região por Agregação de Pixels tem por finalidade associar novos pixels a um dado conjunto de pixels pré-selecionados, de acordo com suas características predominantes [9]. Para tanto, um pixel precisa ser designado como semente. A partir desta semente, de forma iterativa, é realizado o crescimento das regiões, agregando cada pixel vizinho que apresente atributos similares. Tais atributos podem ser classificados como intensidade, textura ou cor [17]. Este processo é ilustrado na Figura 1.

A Figura 1a apresenta uma imagem contendo uma reta vermelha. A partir da escolha do pixel sementem indicado na Figura 1b, os pixels vizinhos que possuem características semelhantes são agregados, dando início à construção da região, conforme observado na Figura 1c. Essa região se expande por completo, evidenciando a tonalidade das bordas, de forma que a diferencie das regiões vizinhas.

2.2 Gradiente de Sobel

O Gradiente de Sobel é um operador de diferenciação capaz de calcular diferenças horizontais e verticais na intensidade do gradiente entre os pixels de uma imagem [19]. Para tanto, multiplicam-se duas matrizes de dimensão 3 × 3 com a imagem original *Im*. Tais matrizes são chamadas de máscaras de convolução, as quais são multiplicadas a cada pixel da imagem, com o intuito de descobrir a sua intensidade vertical e/ou horizontal [5].
Mudanças horizontais são obtidas ao multiplicar a imagem *Im* com a máscara (G_x), definida como:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * Im \qquad (1)$$

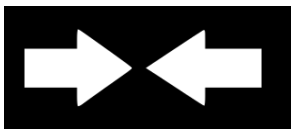
Mudanças verticais são obtidas ao multiplicar a imagem *Im* com a máscara (G_y), definida a partir da Equação 2:

$$G_y = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * Im \qquad (2)$$

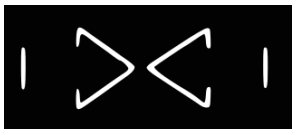
O valor absoluto do operador Sobel para cada pixel é calculado a partir da Equação 3.

$$[Operador\ Sobel] = \sqrt{{G_x}^2 + {G_y}^2} \qquad (3)$$

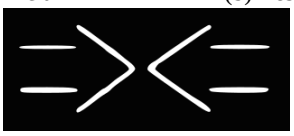
Ao multiplicar a máscara de convolução G_x pela Figura 2a, obtém-se como resultado segmentos de reta onde os pixels apresentam variações verticais, conforme representação da Figura 2b.



(a) Figura inicial



(b) Resultado G_x



(c) Resultado G_y

Figura 2: Aplicação do Operador Sobel. Adaptado de [13].

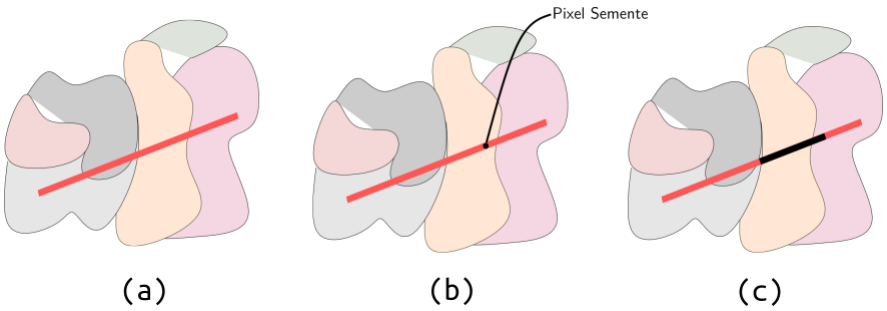


Figura 1: Aplicação do crescimento de região por agregação de pixels.

De maneira similar, ao multiplicar pela máscara de convolução G_y, os segmentos de retas onde os pixels apresentam variações horizontais são evidenciados, como na Figura 2c.

Todavia, segmentos de retas inclinadas podem ser visíveis nas Figuras 2b e 2c visto que, tais segmentos, possuem variações horizontais e verticais simultaneamente. Desta forma o uso do Gradiente de Sobel para tais aplicações deve ser agregado a técnicas de filtragem de retas, como por exemplo, a Transformada de Hough.

2.3 Transformada de Hough Padrão

A transformada Hough é uma técnica de processamento de imagens que tem por objetivo a extração de características presentes em imagens digitais, sem nenhum conhecimento prévio sobre tais características. A transformada de Hough foi desenvolvida por Paul Hough em 1962 com o intuito de detectar linhas retas e curvas em imagens digitais [20]. No entanto, esse método vem sendo expandido para a detecção de qualquer forma geométrica (círculos, elipses, entre outras) [4], desde que possa ser representada matematicamente na forma cartesiana ou parametrizada.

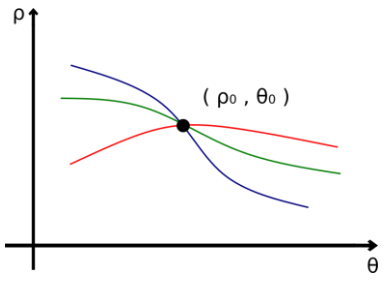


Figura 3: Intersecção dos pixels no plano (ρ,θ).

A ideia central da transformada, conforme observada na Figura 3, é representar um pixel de borda da imagem em uma curva no espaço de parâmetros, de acordo com a forma geométrica que se deseja identificar. Cada pixel na imagem é utilizado para traçar uma curva senoidal no plano (ρ,θ), e cada uma das curvas possui um acumulador associado. Deste modo, se dois pontos distintos, isto é, se dois pixels distintos se cruzam, conforme ilustrado na Figura 3, significa que ambos os pontos pertencem à uma mesma linha. Esta característica implica em um incremento no acumulador da reta [6].

A Transformada Probabilística de Hough utiliza o mesmo algoritmo da transformada padrão, com o objetivo de reduzir a quantidade de ruído presente no processamento da transformada padrão. Além disso, esse método é capaz de reduzir significativamente o tempo de execução em comparação à transformada clássica [10]. Esse algoritmo introduz melhorias nos estágios de contagem e acumulação durante o processamento, uma vez que, ao invés de utilizar todos os pixels de borda, considera apenas uma fração deles.

2.4 Limiarização

A técnica de limiarização, também conhecida como Threshold, é um método aplicado como critério de seleção dos pixels que compõe uma imagem digital. O processo consiste na utilização de uma constante de brilho T, considerada como critério de seleção. Todos os pixels da imagem são comparados a este critério e então alterados conforme a necessidade [5].

Seja $F(x,y)$ uma imagem em escala de cinza e cada pixel da imagem denotado por $F(x_i,y_i)$. Dessa forma:

$$F(x_i,y_i) = \begin{cases} 255, & se\ F(x_i,y_i) > 0 \\ 0, & caso\ contrário \end{cases}$$

De acordo com a Equação 4, quando o valor da intensidade do pixel é maior ou igual ao critério de seleção estabelecido, o pixel correspondente passa a apresentar intensidade máxima (branco). Caso contrário, a intensidade do pixel torna-se mínima (preto). Como resultado, obtém-se uma imagem binária, realçando a região da imagem onde encontra-se o elemento que se deseja destacar.

2.5 Considerações para Implementação

A escolha das técnicas apresentadas fundamentou-se nas características e particularidades apresentadas por cada método. No método do Gradiente de Sobel, busca-se uma reta a partir da intensidade que os pixels que compõe uma imagem apresentam, quando convertida para escala de cinza. Dessa forma, torna-se possível a extração e identificação de retas horizontais ou verticais analisando a intensidade dos pixels. Por outro lado, no método de Agregação de Pixels, busca-se identificar a reta a partir do sistema de cor presente em cada pixel. A extração dos pixels da linha do laser consiste na aplicação da limiarização simples, uma vez que o espectro da câmera e do feixe de laser são conhecidos. Por fim, a

Transformada Probabilística de Hough foi utilizada com vistas a filtragem dos ruídos oriundos da extração.

3. ARQUITETURA DO SISTEMA

Esta seção apresenta a metodologia empregado no desenvolvimento do sistema de medição de distância utilizando métodos de processamento de imagens. O sistema proposto baseia-se em uma plataforma laser/câmera, dotada de uma unidade de processamento dedicada. O objetivo principal dos métodos empregados aqui está na direção da captura do feixe de laser, haja visto aperfeiçoar o sistema ao reduzir o erro de medição.

A plataforma de hardware do sistema proposto é apresentada na Figura 4. Tal plataforma é dotada de uma placa Raspberry Pi B+, uma câmera de 5 MP, um emissor laser de linha vermelho de 5 W e um sensor de luminosidade. O sistema foi desenvolvido em linguagem Python (versão 2.7.15), utilizando como base para as aplicações de processamento de imagem a biblioteca OpenCV. A plataforma de software foi analisada no sistema operacional GNU/Linux Raspibian (kernel versão 4.14.62).

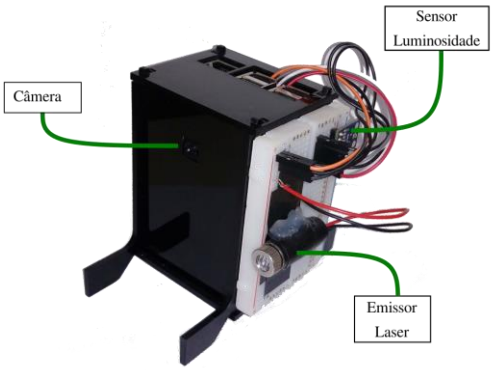


Figura 4: Arquitetura de hardware.

O funcionamento geral do sistema, assim como a sequência em que são executados é apresentado no fluxograma da Figura 5. Inicialmente, verifica-se o funcionamento do emissor laser de linha.

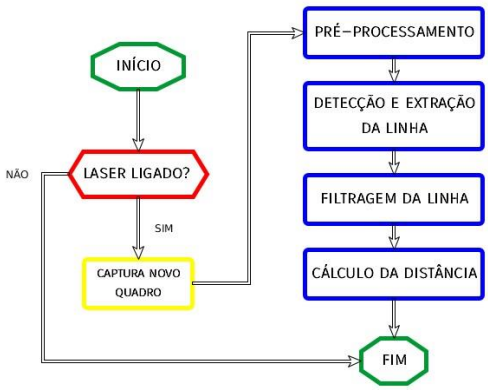


Figura 5: Fluxograma geral do sistema.

Uma vez satisfeita, um novo quadro é capturado, seguido da execução das etapas: (i) pré-processamento, composto por operações de otimização na imagem capturada; (ii) detecção e extração da linha, onde são aplicados os métodos Gradiente de Sobel e Agregação de Pixels; (iii) filtragem da linha, utilizando a Transformada Probabilística de Hough; e (iv) cálculo e determinação da distância.

3.1 Calibração da câmera

O processo de calibração da câmera foi desenvolvido com base no padrão coplanar do tabuleiro de xadrez, ilustrado na Figura 6. A escolha de tal padrão se deu em função da presença de cantos agudos e bordas distintas, bem como a quantidade e dimensões dos quadrados conhecidos. Para realização da calibração os seguintes passos são realizados: (i) obtenção de um conjunto de imagens do padrão de xadrez segundo diferentes orientações e posições no espaço; (ii) localização dos cantos internos do padrão de calibração em todas as imagens adquiridas; e (iii) cálculo dos parâmetros intrínsecos, extrínsecos e de distorção da câmara. Através dos parâmetros intrínsecos e extrínsecos obtidos com o processo de calibração é possível realizar transformações geométricas que mapeiam um ponto 3D no espaço em seu respectivo ponto 2D na imagem.

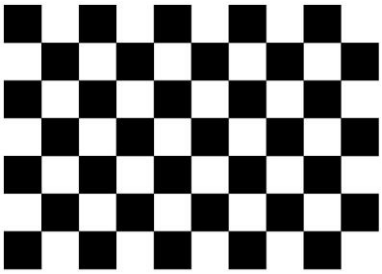
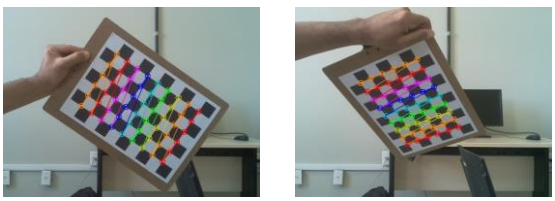


Figura 6: Padrão utilizado na calibração da câmera.

O primeiro passo do processo de calibração consistiu em realizar a captura de 15 imagens, cada uma contendo o padrão escolhido exposto sob diferentes ângulos e orientações à câmera. As localizações dos cantos internos do padrão de calibração foram adquiridas a partir da função findChessboardCorners, disponível na biblioteca OpenCV. Inicialmente, a função determina se, para cada imagem de entrada, é possível obter uma visão do padrão do tabuleiro de xadrez.



(a) (b)

Figura 7: Mapeamento dos cantos internos do tabuleiro de xadrez.

Em seguida, tal rotina identifica e mapeia os cantos internos do tabuleiro de xadrez, tal qual indicado na Figura 7. Como consequência, a função retorna um valor diferente de zero se

todos os cantos foram encontrados. Caso contrário, na situação em que a função não encontrar todos os cantos, ela retorna zero.

Com base nesse mapeamento, a função CalibrateCamera, presente na biblioteca OpenCV, estima os parâmetros intrínsecos da câmera e sua respectiva matriz de distorção. Este processo é realizado apenas uma vez, visto que a câmera deve permanecer fixa ao longo de sua utilização. Os parâmetros resultantes para o módulo de câmera Raspberry Pi são apresentados na Equação 5, para a Matriz de Parâmetros Intrínsecos).

$$\begin{bmatrix} 6{,}77 * 10^2 & 0 & 3{,}74 * 10^2 \\ 0 & 6{,}78 * 10^2 & 2{,}43 * 10^2 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Do mesmo modo, a Equação 6 apresenta a representação aritmética da Matriz de Distorção.

$$\begin{bmatrix} 7{,}08 * 10^{-2} & 3{,}45 & 1{,}58 * 10^{-3} \\ 5{,}09 * 10^{-4} & -4{,}26 * 10 & 0 \end{bmatrix} \quad (6)$$

De posse da matriz de parâmetros intrínsecos e de distorção, a função undistort, também presente na biblioteca OpenCV, realiza a correção da distorção nas imagens capturadas através câmera. Este tipo de distorção produz variações no tamanho de um objeto de acordo com a região que se encontra na cena, assim como alternâncias na espessura da linha do laser.

3.2 Pré-processamento

A finalidade das operações de pré-processamento é obter uma imagem com dimensões reduzidas e com níveis de cores balanceados. Uma vez que o laser apresenta altura fixa em relação ao plano focal da câmera, realiza-se um corte na altura da imagem capturada, a fim de reduzir sua dimensão sem perder a informação do laser. Tal etapa é caracterizada pelo Algoritmo 1.

Algorithm 1 ETAPA DE PRÉ-PROCESSAMENTO DA IMAGEM.

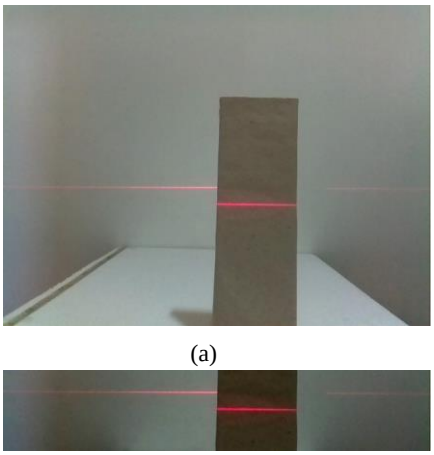
```
Entrada: ImgEntrada, alturaCorte
início
    imgCorte = corteImagem(ImgEntrada, alturaCorte)
    imgNorm = normalizar(imgCorte)
    imgFim = filtroGaussiano(imgNorm)
fim
Saída: imgFim
```

Tendo em vista que as imagens podem ser adquiridas sob diferentes condições de iluminação, sua paleta de cores pode se afetada, dificultando a localização da linha do laser. Sendo assim, o sistema possui um mecanismo de normalização da imagem capturada. Em seguida, um filtro de desfocagem gaussiana é aplicado, de forma que os pixels apresentem cores uniformes. Este processo facilita a determinação do feixe de laser na imagem. O resultado desta etapa de pré-processamento é uma imagem com dimensões reduzidas e paleta de cores ajustada, conforme as Figuras 8a e 8b.

3.3 Detecção e Extração da Linha

Dois mecanismos foram utilizados para detecção e extração da linha: (i) Gradiente de Sobel; e (ii) Agregação de Pixels. Ambos os métodos foram desenvolvidos, com base nas especificações aqui apresentadas.

O processamento baseado na técnica de Gradiente de Sobel é explicitado a partir do Algoritmo 2. Inicialmente é realizada uma transformação na escala de cores da imagem, transformando-a para tons de cinza. Em seguida, a diferença na intensidade horizontal dos pixels é calculada, resultando no realce da linha do laser. A partir do resultado oriundo do Gradiente de Sobel, a função de limiarização realiza a análise, pixel a pixel da imagem, a medida em que determina se o mesmo pertence ou não ao feixe presente na imagem, caracterizando-a conforme o limiar de brilho pré-estabelecido.



(a)

(b)

Figura 8: Pré-processamento da imagem de entrada.

Algorithm 2 GRADIENTE SOBEL

```
Entrada: imgFim
início
    cinza = escalaCinza(imgFim)
    gy = gradienteSobel(cinza)
    imgLimiar = limiarizacao(gy)
fim
Saída: imgLimiarizada
```

Os resultados desta etapa de processamento são ilustrados na Figura 9.

Figura 9: Resultado do Gradiente de Sobel sobre uma imagem de teste.

A especificação do método de Agregação de Pixels é descrita a partir do Algoritmo 3. A primeira etapa de processamento consiste na transformação da imagem para o sistema de cores Hue, Saturation e Value (HSV), uma vez que este sistema apresenta maior abrangência sobre as cores do espectro emitido

pelo laser. Para realizar a agregação dos pixels pertencentes ao feixe de laser, a função inRange foi utilizada. Tal função está disponível na biblioteca OpenCV, e é responsável por localizar e agregar os pixels pertencentes ao intervalo de cores especificado.

A região agregada passa pelo processo de abertura morfológica. Tal técnica combina a erosão, removendo os pixels em um dado raio, juntamente com a dilatação, responsável pelo acréscimo de segmentos de linha remanescentes do processo de erosão. Por fim, o feixe de laser é destacado através da função bitwiseAnd, disponível na biblioteca OpenCV. Esta função recebe como parâmetro uma imagem de entrada e a região na qual se encontra o feixe de laser. Como resultado, todos os pixels que pertencem à região são realçados, conforme apresentado na Figura 10.

Assim como no método do Gradiente de Sobel, a caracterização dos pixels ocorre a partir da função Threshold.

Algorithm 3 AGREGAÇÃO DE PIXELS.

```
Entrada: imgFim
início
    imgHSV = convertToHSV(imgFim)
    regiao = inRange(imgHSV, faixaCor)

    regiaoFinal = morfologia(regiao)
    pxAgregados = bitwiseAnd(imgFim, regiaoFinal)

    cinza = escalaCinza(pxAgregados)
    imgThershold = thershold(cinza)
fim
Saída: imgLimiarizada
```

Figura 10: Resultado do processo de Agregação de Pixels.

3.4 Filtragem da Linha

Frequentemente, pixels não pertencentes a linha do laser estão presentes nos resultados oriundos do Gradiente de Sobel e Agregação de Pixels. O surgimento de tais ruídos está relacionado às superfícies de reflexão dos objetos em questão. A filtragem da linha é realizada a partir da função HougLinesP, disponível na biblioteca OpenCV. O Algoritmo 4 foi utilizado no sentido de filtrar a linha do laser, eliminando os ruídos ou segmentos de linha irrelevantes para o cálculo da distância.

Essa rotina é responsável por encontrar os segmentos de linha utilizando o método da Transformada Probabilística de Hough. Em seguida, os pixels referentes à altura da linha do laser são armazenados em um vetor. Uma vez que a distância até o objeto é calculada a partir da altura máxima em que a projeção da linha do laser se apresenta na imagem, tal vetor é percorrido a fim de encontrar a altura máxima, a qual corresponde ao ponto do feixe de laser refletido no objeto mais próximo à câmera.

Algorithm 4 FILTRAGEM DA LINHA

```
Entrada: imgThershold
início
    alturas = [ ]
    linhas = houghLinesP(imgThershold)
    if linhas != Vazio then
        repita
            alturas.adicionar(linhas(y_i))
        até linhas(fim);
        repita
            if alturas[i] ≥ yMax then
                yMax = alturas[i]
        até alturas(fim);
    else
        yMax = Vazio
    end
fim
Saída: yMax
```

3.5 Cálculo da Distância

O cálculo da medida de distância até um objeto que se encontra no campo de visão da câmera é determinado com base no modelo apresentada na Figura 11. Em função da estrutura da plataforma, apresentada na Figura 4, o emissor laser de linha foi posicionado ao lado da câmera, a uma distância vertical de 2,8 cm, na direção inferior. Na Figura 11, a distância do centro da imagem até a posição em que a linha do laser é projetada no objeto é definida como a distância ao Plano Focal (DPF), e θ representa a medida do ângulo de visão (em radianos) formado pela projeção da linha do laser na imagem.

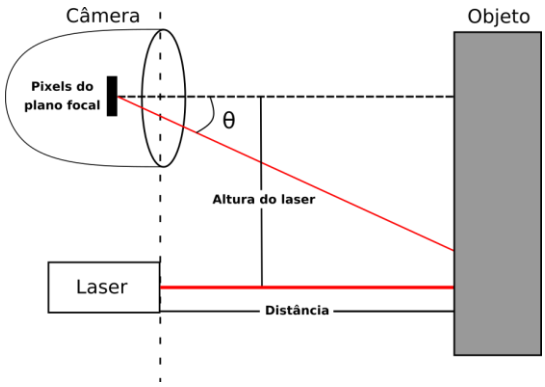


Figura 11: Modelagem para obtenção da distância ao objeto.

A distância até o objeto é determinada a partir da altura com que a linha do laser é projetada na imagem. A relação para o cálculo de determinação da distância é descrita a partir da análise trigonométrica do sistema, tal como apresentado na Equação 7, na qual *h* representa a altura do laser e *d* a distância medida.

$$d = \frac{h}{\tan\theta};\ \theta = (DPF * RPP) + DR \qquad (7)$$

A expansão da relação da apresentada na Equação 7 é utilizada para converter cada pixel da imagem, em seu valor correspondente medido em centímetros. Para tanto, determina-se a expressão que relaciona a quantidade de ângulo por pixels da imagem, também

chamada de Radianos por Pixel (RPP), e a DPF, equivalente a altura máxima da linha projetada sobre o objeto. Em função da existência de deformações na lente da câmera, considera-se o parâmetro de correção angular, denotado como Deslocamento Radiano (DR).

Como resultado, as distâncias para os objetos que estejam posicionados frente à câmera são obtidas, conforme ilustrado na Figura 12.

3.6 Integração com a API do ROS

O Robot Operating System (ROS) é uma estrutura voltada para o desenvolvimento softwares para aplicações de robótica. Em sua arquitetura, o ROS disponibiliza uma coleção de ferramentas, bibliotecas e padrões que visam simplificar comportamentos complexos e robustos associados aos sistemas robóticos.

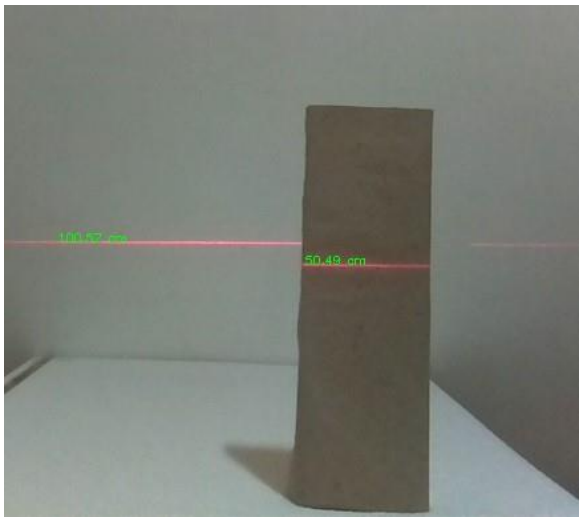


Figura 12: Resultado final para múltiplos objetos frente a câmera.

Da perspectiva do robô, muitos problemas que parecem triviais para os seres humanos, como por exemplo buscar um item em um ambiente, apresentam variações entre instâncias de tarefas e ambientes. Com base nisso, o ROS foi concebido para incentivar o desenvolvimento de softwares de robótica, transformando comportamentos de modelagem complexa em tarefas de baixa complexidade. A partir desta premissa, o ROS proporciona mecanismo para estabelecer a comunicação entre os componentes estruturais do sistema, oferecendo suporte a plataformas robóticas e dispositivos periféricos [14]. Conforme a Figura 13, os componentes básicos da estrutura de comunicação estabelecida pelo ROS são: (i) o servidor (mestre), (ii) os tópicos, (iii) nós, (iv) mensagens e (v) serviços.

O servidor é o responsável por inicializar as dependências, bibliotecas do ROS e o sistema de troca de mensagens. Os nós são arquivos executáveis, ou rotinas desenvolvidas para realizar determinadas tarefas, que utilizam as bibliotecas do ROS para estabelecer a comunicação com outros nós inscritos no mesmo tópico, publicar ou assinar tópicos, assim como prover ou usar serviços.

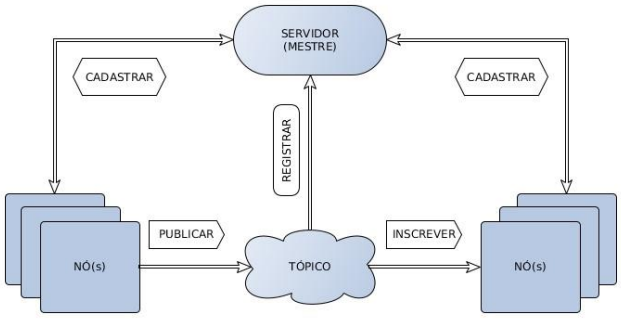


Figura 13: Estrutura de comunicação do ROS. Adaptado de [1].

Os tópicos são estruturas que armazenam tipos específicos de mensagens publicadas pelos nós, as quais podem ser visualizadas por toda a aplicação. Dessa forma, sempre que o nó publicar novas mensagens em um tópico, os nós inscritos receberão a informação referente àquela mensagem. Os serviços podem ser vistos como outra forma a partir da qual os nós podem se comunicar, porquanto permitem que os nós enviem e recebam solicitações de mensagens e serviços.

A estrutura de nós empregada no modelo de software proposto é apresentada na Figura 14. O nó FluxoCâmera é responsável por realizar a captura constante, na forma de fluxo de vídeo. Por outro lado, o no´ Imagem é responsável por capturar somente uma imagem a partir da câmera conectada ao Raspberry Pi.

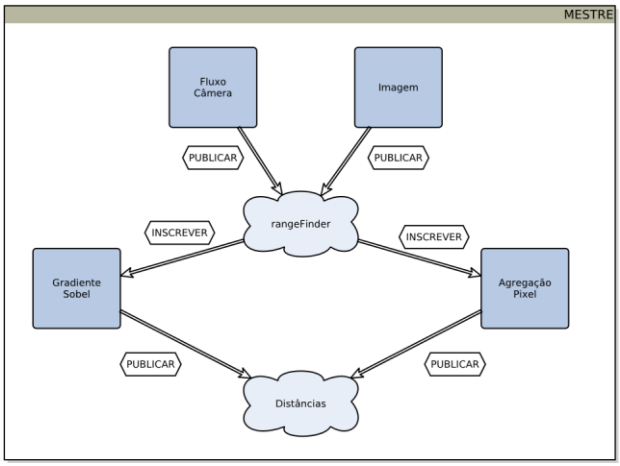


Figura 14: Estrutura de comunicação do ROS para o modelo de software proposto.

Ambos os nós, são encarregados de publicar as imagens capturadas no tópico rangeFinder. Uma vez que a imagem é publicada, os nós definidos como GradienteSobel e AgregaçãoPixel iniciam o processamento das imagens, realizando a detecção e extração da linha de laser de acordo com o mecanismo estabelecido, efetuam o cálculo da distância e publicam o resultado da operação no tópico Distâncias.

4. RESULTADOS

Para validar a aplicação proposta foram realizados testes de precisão, cujo intuito foi estimar um intervalo de confiança para as medições realizadas a partir do sistema. Além disso, uma análise sobre o tempo de execução e consumo de memória que cada técnica apresenta foi realizada. O objetivo de tal análise foi avaliar o impacto da implementação do sistema em plataformas computacionais de baixo custo e bom baixo poder de processamento. Tais testes de desempenho foram realizados sobre uma placa Raspberry Pi B+.

A fim de determinar a confiabilidade das medições de distância, os testes foram realizados em ambiente com nível de luminosidade controlada, permitindo assim uma melhor identificação da linha de laser projetada sobre os objetos. As medições foram realizadas sob a perspectiva de duas técnicas de detecção do laser de linha: (i) método do Gradiente de Sobel; e (ii) Agregação de Pixels.

A distância para o objeto foi aferida usando uma fita métrica e então comparada com o resultado obtido através do sistema. Durante os testes, verificou-se que, para distâncias inferiores à 20 cm, o foco do feixe de laser é perdido pela câmera. Por outro lado, para distâncias superiores à 100 cm, a projeção da linha do laser não é capturada pela a câmera devido à baixa intensidade da luminosidade laser empregado nos testes. Deste modo, em razão das restrições impostas pelo sistema câmera/laser, a faixa de variação das medições foi estabelecida entre as medidas 20 cm e 100 cm. Os testes empregados aqui consistiram em medições sucessivas praticando variações nas distâncias em passos de 10 cm. Ao utilizar o método de detecção da linha de laser a partir do Gradiente de Sobel, foram capturadas cinco imagens para cada medida de distância. A Tabela 1 sintetiza a média das distâncias obtidas para as cinco imagens sob diferentes medidas de distância.

Tabela 1: Resultados de medida de distância utilizando o Gradiante de Sobel.

Distância	**Gradiente de Sobel**		
	Média		σ
20 cm	20,216	**0,216**	0,679
30 cm	30,428	0,428	0,893
40 cm	40,494	0,494	0,973
50 cm	50,506	0,506	0,819
60 cm	60,52	0,520	1,284
70 cm	69,362	0,638	1,306
80 cm	80,676	0,676	2,140
90 cm	91,258	1,258	1,212
100 cm	101,38	1,380	1,658

A partir da análise da Tabela 1, observa-se que o erro médio das medidas apuradas pelo sistema apresenta um comportamento crescente. Dessa forma, ao aumentar a distância do objeto à câmera, o erro apresentado pelo sistema é potencializado.

No intervalo entre 20 cm e 40 cm verifica-se que o erro médio obtido foi inferior a 0,5 cm, refletindo no desvio padrão inferior a 1 cm. Com o desvio obtido, é possível identificar a homogeneidade das medições. O menor erro obtido 0,216 cm aconteceu para a distância de 20 cm, a maior dispersão das estimativas de distância ocorreu para medições onde o objeto se encontrava a 80 cm de distância. A causa está associada à baixa qualidade na captura das imagens, resultando na imprecisão dos parâmetros RPP e DR. Ainda é possível identificar a pior medida de distância, ocorrida para a medida de 100 cm. Esse resultado corrobora com a afirmação que a medida em que os objetos se distanciam da câmera, há um aumento no erro médio associado a tais medidas. E possível considerar, portanto, que tais desvios´ não possuem uma influência abrangente, evidenciando a precisão do método dentro das limitações físicas do hardware especificadas.

Ao utilizar o método de Agregação de Pixels para detecção da linha do laser, também foram obtidas cinco imagens para cada uma das distâncias pertencentes ao intervalo de 20 cm a 100 cm com passos de 10 cm. Os resultados oriundos do processamento são apresentados na Tabela 2.

Com base na análise da Tabela 2, constata-se que o maior erro médio obtido se apresentou na distância de 20 cm. Este fato é devido às restrições físicas do sensor de foco da câmera, o qual, para distâncias muito curtas, não é capaz de estabilizar o foco da imagem, resultando na variação na espessura do feixe de laser. Para distâncias superiores a 20 cm, o erro obtido nas medições apresenta comportamento crescente, de modo que a medida em que o objeto se distancia da câmera há o aumento do erro. O comportamento crescente do erro pode ser observado até aproximadamente 70 cm. A partir desta medida, as distâncias mensuradas apresentam erros não sistemáticos. Essa disparidade nos erros é oriunda das características de hardware do conjunto câmera/laser, tal que, a baixa percepção do feixe de laser interfere no processamento do sistema.

Tabela 2: Resultados de medida de distância utilizando a Agregação de Pixels.

Distância	**Agregação de Pixels**		
	Média		σ
20 cm	20,992	0,992	1,562
30 cm	30,460	**0,460**	1,745
40 cm	40,678	0,678	1,092
50 cm	51,828	1,828	2,297
60 cm	62,074	2,074	1,208
70 cm	68,686	1,314	0,877
80 cm	80,726	0,726	2,224
90 cm	90,676	0,676	1,681
100 cm	99,112	0,888	3,381

Observa-se que os erros obtidos demonstram um aspecto não homogêneo. Sendo assim é possível considerar que, em comparação ao método do Gradiente de Sobel, a acurácia deste método não é eficaz, visto que, as limitações impostas pelo

conjunto câmera/laser interferem severamente nas medidas de distâncias através deste método.

Com a finalidade de estipular a taxa de precisão dos métodos propostos, uma análise comparativa com os trabalhos presentes na literatura, orientados à medição de distância utilizando emissores lasers, foi realizada. Os resultados oriundos desta análise são apresentados na Tabela 3.

No trabalho apresentado em [21], os autores propõem a utilização de dois métodos para determinação de distância utilizando emissor laser de ponto: (i) série polinomial de Fourier; e (ii) filtro Chebyshev. Estes métodos são aplicados como ajustes da curva de detecção, com a finalidade de compensar o erro computacional induzido pela distorção de lentes monocular. O erro médio foi mensurado para a faixa de 29 cm à 90 cm, alcançando erro médio de 0,143 cm e 0,411 cm, respectivamente.

O método proposto por [8] utiliza um emissor laser de ponto para determinar a distância para um objeto. A estrutura apresentada consiste na captura de imagem contendo o ponteiro laser. Em seguida, a imagem é binarizada, destacando o pixel onde se encontra o ponteiro laser. A faixa de detecção do erro médio deu-se no intervalo de 20 cm à 70 cm, atingindo erro médio de 0,747 cm.

Tabela 3: Comparação do nosso trabalho com métodos presentes na literatura.

Método	**Erro médio (cm)**
Filtro Chebyshev [21]	0,143
Série polinomial de Fourier [21]	0,411
Binarização da imagem [8]	0,747
Segmentação por cores [7]	2,642
Gradiente de Sobel (Nosso)	0,681
Agregação de Pixels (Nosso)	1,071

O trabalho apresentado em [7] introduz a metodologia de busca do emissor laser de ponto através do espectro de cor emitido pelo mesmo. A faixa de detecção avaliada consistiu no intervalo de 29 cm até 100 cm, alcançando erro médio de 2,642 cm.

Com o intuito de estabelecer um perfil de luminosidade dentro do intervalo no qual o sistema é capaz de atuar, foram realizados testes variando a iluminação do ambiente. Tais testes consistiram na captura de imagens a 20 cm, 50 cm e 100 cm, representando a distância mínima, distância intermediária e distância máxima do intervalo definido nas análises preliminares. Em cada distância, a luminosidade foi diferenciada sob 4 níveis diferentes. Os resultados são descritos na Tabela 4.

A partir dos dados na Tabela 4 observam-se que os quatro níveis de luminosidade para medida de 20 cm diferiram de forma não sistemática. Esse fato ocorre devido às características do hardware da câmera, visto que, para distâncias pequenas e baixa luminosidade, o sensor de foco da câmera se apresenta instável. Durante os experimentos, observou-se que a linha do laser se tornava espessa e tênue de forma intermitente. Em contrapartida, pode-se destacar o menor erro obtido para este teste com 150 lux, devido a capacidade de o foco automático estabilizar a imagem dada a maior incidência de luz no sensor da lente. Por esta razão, o impacto da luminosidade sobre tal distância acarreta em uma maior acurácia.

Tabela 4: Distâncias para diferentes níveis de luminosidade.

Distância	**Luminosidade**			
	0 lux	**30 lux**	**80 lux**	**150 lux**
20 cm	22,7	23,83	22,7	20,64
50 cm	50,49	50,49	50,49	49,05
100 cm	100,52	100,52	100,52	94,99

A análise dos níveis de luminosidade para a medida de 50 cm, apontam que a distância medida se manteve invariável para os três primeiros níveis de luminosidade, uma vez que o sensor de foco da câmera permanece estável durante toda a medição. A partir do aumento da incidência de luz sobre o feixe de laser, o espectro de cor dos seus pixels começa a sofrer variações, dificultado assim a percepção da linha pelo sistema.

O valor máximo do intervalo apresentou um comportamento semelhante ao anterior, visto que, o erro para o nível máximo de luminosidade cresceu cerca de 5 %. Com base nesta análise, observa-se que para o nível de luminosidade até 80 lux o sistema se mantém constante. A medida em que o objeto se distancia da câmera, e a incidência de luz no ambiente aumenta, a taxa de erro associada também aumenta.

Os testes utilizados para realizar a análise do desempenho do sistema consistiram de duas etapas: (i) testes de tempo de execução; e (ii) consumo de memória. O teste de tempo de execução foi realizado a fim de estabelecer o tempo médio na execução dos métodos Gradiente de Sobel e Agregação de Pixels, aplicando as rotinas disponíveis no módulo Line Profiler, presente na biblioteca do Python, 10 vezes para cada imagem pertencente ao intervalo detectado pelo sistema. De forma análoga, com o uso das rotinas disponíveis no módulo Python Memory Profile determinou-se o consumo médio de memória para cada um dos métodos.

Para cada execução da rotina, foram produzidos registros contendo o perfil do tempo de execução, linha a linha, para cada um dos métodos, totalizando assim 180 perfis. Posteriormente, a média do tempo total de execução para cada medida de distância foi apurada. Com base nos dados presentes na Tabela 5, observa-se que o método do Gradiente de Sobel possui um tempo de execução menor devido a sua característica de execução. Em tal método multiplica-se a imagem pela máscara G_y, obtendo como resultado as linhas horizontais presentes na imagem. Por outro lado, no método de Agregação de Pixels, para cada pixel da imagem, é necessário verificar se o mesmo pertence ou não à uma região previamente especificada.

Para o método de Agregação de pixels, o maior tempo de execução ocorreu em 30 cm. Constatou-se que esta medida foi devido à distorção do feixe de laser conforme aproximação do objeto à câmera. Nestes casos, conforme a distância do objeto à câmera aumenta, a espessura do feixe de laser diminui. Um comportamento atípico é notório para as medidas 20 cm e 100 cm,

visto que devido as características do conjunto câmera/laser, existe a perda de foco ocasionando à baixa visualização do feixe de laser.

Tabela 5: Tempo de execução.

Distância	Tempo médio de execução (ms)	
	Gradiente de Sobel	**Agregação de Pixels**
20 cm	**152,84**	237,44
30 cm	155,34	241,55
40 cm	157,20	239,03
50 cm	154,43	234,98
60 cm	155,80	234,53
70 cm	155,00	233,25
80 cm	157,53	231,40
90 cm	157,06	231,09
100 cm	154,50	228,35

Outro fator que afetou o tempo de execução do método de Agregação de Pixels foi a presença de falsos-positivos devido à escolha do pixel semente. Sendo assim, a aplicação do filtro morfológico faz-se necessária, dado que o mesmo reduz significantemente a presença desses ruídos. Posteriormente, ainda se faz necessário a aplicação dos filtros de Limiarização e Hough Lines, a fim de restringir os resultados obtidos para a reta correspondente ao feixe de laser.

Para uma plataforma robótica móvel, os dados oriundos do processamento de uma dada aplicação de sensoriamento devem estar disponíveis o mais breve possível, de modo que o mesmo seja capaz de realizar as tomadas de decisão necessárias e em tempo hábil. Sendo assim, no que se refere ao tempo de execução, os métodos analisados neste trabalho apresentam resultados favoráveis ao uso em plataformas robóticas com baixo poder computacional.

Tabela 6: Consumo de memória.

Distância	Consumo médio de memória (MB)	
	Gradiente de Sobel	**Agregação de Pixels**
20 cm	0,39	0,25
30 cm	0,40	0,25
40 cm	0,41	**0,24**
50 cm	0,40	0,25
60 cm	0,40	0,25
70 cm	0,39	**0,24**
80 cm	0,39	**0,24**
90 cm	0,40	**0,24**
100 cm	0,40	0,25

O teste de consumo de memória visa avaliar se os métodos propostos podem ser aplicados em computadores de placa móvel, visto que os mesmos, em geral, apresentam memória limitada. Esse teste consistiu em traçar um perfil do consumo de memória para ambos os métodos desenvolvidos no presente trabalho, cujo resultado é apresentado na Tabela 6. O processo de análise de perfil de memória foi desenvolvido a partir de rotinas presentes no módulo Memory Profiler. Para cada imagem, o perfil foi analisado 10 vezes, resultando em um total de 180 perfis de execução.

Conforme descrito na Tabela 6, percebe-se que não há uma distinção evidente entre o consumo de memória por parte de ambos os métodos, com variação máxima de até 160 KB. Sob o ponto de vista de memória, a utilização de ambos os métodos é viável para plataformas computacionais com baixa capacidade de armazenamento.

5. CONCLUSÃO

Sensores visuais fazem uso de imagens e vídeos para realizar um vasto conjunto de tarefas. O uso dos sensores visuais introduziu um novo aspecto ao desenvolvimento de plataformas de robótica móvel, que até então necessitavam de múltiplos sensores para realizar diferentes tarefas. A partir de uma única câmera, um sensor visual é capaz de fornecer dados para suas respectivas tarefas. O presente trabalho apresentou um sensor visual de distância utilizando laser de linha implementado em uma plataforma computacional Raspberry Pi, modelo B+. Tal sensor utiliza técnicas de visão computacional para processar as imagens capturadas a partir de uma câmera. O processamento envolve detectar o laser e determinar a distância para os múltiplos objetos presentes no campo de visão da câmera.

Com base nos resultados obtidos, observa-se que ambos os métodos apresentam uma precisão notável na faixa de operação do sistema. O uso do emissor laser de linha introduz um aspecto positivo à detecção, tal que, diferente do emissor laser de ponto, não necessita que os objetos estejam dispostos na sua linha de ação. O contraste da acurácia entre os resultados obtidos e os trabalhos presentes na literatura é aproximadamente 0,58 cm, ratificando a precisão dos métodos propostos.

O sistema ainda é capaz de fornecer as informações com tempo de resposta aproximadamente de 153 ms. Em condições normais, o método do Gradiente de Sobel apresenta tempo de resposta cerca de 68 % mais rápido quando comparado ao método de Agregação de Pixels. Destaca-se ainda que há uma perda considerável na precisão do sistema para ambientes com altas taxas de luminosidade, como por exemplo, ambientes externos, em que há incidência direta de luz solar.

O hardware nativo da Raspberry Pi B+ se mostrou adequado para o processamento das imagens capturadas utilizando os módulos disponíveis na biblioteca de visão computacional OpenCV. Com base nas análises de desempenho efetuadas, este sensor se mostra adequado para aplicações em sistemas embarcados, devido a sua simplicidade, velocidade e baixo consumo.

Como trabalhos futuros, é desejável integrar o sensor a uma plataforma robótica, a fim de investigar a execução do sensor em um ambiente de aplicação. Além disso, é possível expandir e aprimorar a faixa de percepção do laser para distâncias superiores a 100 cm, tendo em vista a detecção mediante ambientes com incidência direta de luz solar. Finalmente, ainda é desejável

investigar a determinação da pose dos obstáculos em relação ao plano da câmera, a partir da reflexão da linha do laser.

6. REFERÊNCIAS

[1] M. H. Achmad, G. Priyandoko, R. Rosli, and M. R. Daud. Tele-Operated Mobile Robot for 3D Visual Inspection Utilizing Distributed Operating System Platform. *International Journal of Vehicle Structures & Systems*, 9(3):190–194, 2017.

[2] E. R. DE PIERI. Curso de robótica móvel. *Santa Catarina, Florianópolis*, 2002.

[3] G. FERRAREZI et al. Sistema de aquisição de imagem do olho humano para avaliação da resposta da pupila submetida a estímulos luminosos. *Trabalho de Conclusão de Curso, EESCUSP*, 2010.

[4] I. S. Gonçalves. Desenvolvimento de um sistema reativo para desvio de obstáculo. 2011.

[5] R. C. Gonzalez and R. E. Woods. Digital image processing second edition. *Beijing: Publishing House of Electronics Industry*, 455, 2002.

[6] L. S. Guedes. Estudo e aplicação de métodos de segmentação de imagens para o diagnóstico de falhas na fabricação de equipos. 2014.

[7] A. A. Hamad, D. A. Dhahir, B. R. Mhdi, and W. H. Salim. Laser distance measurement by using web camera. *Laser*, 4(04), 2014.

[8] F. B. Jesus, P. Joel Filho, J. C. Bittencourt, and T. C. Jesus. Modeling and simulation of laser rangefinder architecture. *system*, 2015.

[9] R. H. Junior. *Segmentação de imagens por morfologia matemática*. PhD thesis, Instituto de Matemática e Estatística da Universidade de São Paulo, 07.03. 97., 1997.

[10] V. Leavers. Which hough transform? *CVGIP: Image understanding*, 58(2):250–264, 1993.

[11] F. A. Manzi et al. *Aplicação de visão computacional para extração de características em imagens do olho humano*. PhD thesis, Universidade de São Paulo, 2007.

[12] J. Marchi et al. Navegação de robôs moveis autônomos: estudo implementação de abordagens. 2001.

[13] C. Meccanismo. Opencv python, edge detection and image gradient analysis, 2019.

[14] M. Quigley, B. Gerkey, and W. D. Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. "O'Reilly Media, Inc.", 2015.

[15] M. L. Rios et al. Visão computacional aplicada ao monitoramento de robôs moveis em cenários de robótica educacional. 2017.

[16] V. ROMANO and M. DUTRA. Introdução a robótica industrial. *Robótica Industrial: Aplicação na Indústria de Manufatura e de Processo, São Paulo: Edgard Blucher*, pages 1–19, 2002.

[17] M. F. Saldanha and C. FREITAS. Segmentação de imagens digitais: Uma revisão. *Divisão de Processamento de Imagens-Instituto Nacional de Pesquisas Espaciais (INPE), São Paulo*, 2009.

[18] N. Sebe, I. Cohen, A. Garg, and T. S. Huang. *Machine Learning in Computer Vision*, volume 29. Springer Science Business Media, 2005.

[19] S. Selhorst. Utilização da visão computacional e detecção de características para auxiliar na navegação de pessoas com deficiência visual em ambientes internos. 2014.

[20] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science Business Media, 2010.

[21] X. Zhang, Y. Yang, Z. Liu, and J. Zhang. An improved sensor framework of mono-cam based laser rangefinder. *Sensors and Actuators A: Physical*, 201:114–126, 2013.