UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SISTEMA DE AUXÍLIO À NAVEGAÇÃO PARA PORTADORES DE DEFICIÊNCIA VISUAL

FERNANDO CIRINO SATO

ORIENTADOR: PROF. DR. EMERSON CARLOS PEDRINO

São Carlos – SP

Novembro/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SISTEMA DE AUXÍLIO À NAVEGAÇÃO PARA PORTADORES DE DEFICIÊNCIA VISUAL

FERNANDO CIRINO SATO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Processamento de Imagens e Sinais
Orientador: Prof. Dr. Emerson Carlos Pedrino

São Carlos – SP

Novembro/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S253sa

Sato, Fernando Cirino.
Sistema de auxílio à navegação para portadores de deficiência visual / Fernando Cirino Sato. -- São Carlos : UFSCar, 2015.
102 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Processamento de imagens. 2. Visão estereoscópica. 3. Navegação. 4. Deficientes visuais. I. Título.

CDD: 006.42 (20ª)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Fernando Cirino Sato, realizada em 16/12/2014:

Prof. Dr. Emerson Carlos Pedrino
UFSCar

Prof. Dr. Michel Andre Maurice Hospital
UFSCar

Prof. Dr. Valentin Obac Roda
UFRN

Dedico ao meus pais que sempre me apoiaram e me ajudaram sempre que eu precisei, minha mãe Olívia que sempre me apoiou e meu pai Osvaldo que sempre estará no meu coração e com certeza está me guiando lá de cima.

AGRADECIMENTOS

Agradeço a Deus por me proporcionar todas as oportunidades que eu tive até agora e que terei no futuro. Agradeço aos meus pais Osvaldo e Olívia que sempre me deram suporte e amor e com certeza as conquistas da minha vida são resultado da dedicação em me criar da melhor maneira possível. Agradeço meu irmão Henrique por sempre estar ao lado para qualquer situação. Agradeço a toda minha família, sem citar todos para eu não esquecer alguém, porém uma menção especial para minha madrinha Terezinha e para minha tia Enih que com certeza ajudaram muito neste período do mestrado, dando condições para eu me dedicar exclusivamente ao estudo. Agradeço aos meus amigos de república Davi, Fabinho e Honda(ordem alfabética) que por dividirmos o mesmo apê ,além do quase membro da república Guilherme, também conhecido como Giu. Todos nós compartilhamos o mesmo objetivo de ser mestre e sempre com muita harmonia, sem uma única briga(apenas algumas histórias engraçadas). Agradeço aos meus amigos do grupo semdiquete Hélder e Richard que com certeza foram amizades que eu valorizo muito por termos muitos gostos parecidos e por isso sempre temos algo bacana para conversarmos(menos coisas alfa, beta ou features novas). Agradeço aos camaradas do laboratório GAPIS: Guilherme, Lucas Goiás, Paulinho, Pillon, Ettore e Andrés! E aos amigos do DC: Cláudia, Vitinho, Odair, Carol, Diego Saqui, Thiago,André,Augusto entre outros! Gostaria de agradecer também ao sr.Nelson e ao sr.Miguel, da Associação de Deficientes Visuais de Marília pela grande ajuda em proporcionar o encontro com deficientes visuais para realizar o questionário deste trabalho e compartilhar as experiências da vida de uma pessoa deficiente visual. Agradeço ao meu ex-orientador de graduação Leonardo Botega por mostrar os caminhos a serem percorridos para chegar ao mestrado. Por fim agradeço ao meu orientador Emerson Pedrino que sempre me ajudou, me guiou e me aconselhou não só no mestrado mas em outros aspectos da vida, sempre pensando o melhor para mim, com as várias caminhadas que fizemos e sempre trocamos idéias caminhando, com certeza meu sucesso neste trabalho é em grande parte a sua contribuição com ótimas idéias e maneiras de fazer pesquisa.

”Não existe um caminho para a felicidade. A felicidade é o caminho.”

Mahatma Gandhi

RESUMO

Os sistemas de auxílio à navegação podem ser utilizados para auxiliar deficientes visuais a realizar tarefas do cotidiano, como por exemplo sua locomoção e orientação. O objetivo deste projeto é desenvolver um protótipo de um sistema de auxílio à navegação para deficientes visuais em ambientes internos. O sistema possui sensores de vídeo que detectam obstáculos à frente, e marcadores contendo pontos de interesse que possam ser de utilidade ao usuário como portas e escadas. A metodologia deste trabalho implica em selecionar ferramentas e técnicas de visão computacional e processamento digital de imagens que possam ser de utilidade à implementação de um protótipo que valide o trabalho proposto. Como resultado, foi criado um protótipo que detecta obstáculos em tempo real, assim como pontos de interesse.

Palavras-chave: Visão Estereoscópica, Processamento de Imagens, Navegação, Deficientes Visuais

ABSTRACT

Aid to navigation systems can be used to help the visually impaired to perform daily tasks, such as their mobility and orientation. The objective of this project is to develop a prototype of an aid to navigation system for visually impaired people in indoors environments. The system has video sensors that detect obstacles ahead and markers containing points of interest that may be useful to the user, as doors and stairs. The methodology of this work involves selecting tools, computer vision techniques and digital image processing that may be useful to implement a prototype to validate the proposed work. As a result, a prototype was created that detects obstacles in real-time as well as points of interest.

Keywords: Stereoscopic Vision, Image Processing, Navigation, Visually Impaired

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A locomoção, sem dúvidas é um dos recursos mais importantes e essenciais para todas as espécies, inclusive a humana, sejam para ações simples como se mover poucos metros de distancia ou milhares de quilômetros para diversas finalidades como trabalho, esporte, lazer e outras finalidades. Neste contexto, o sentido da visão pode ser considerado importantíssimo, pois com ele, sabemos o caminho mais seguro que devemos nos locomover e também sabemos a localização de pontos de interesse como portas, janelas e escadas; logo, sem o sentido da visão a navegação é seriamente comprometida.

Os Sistemas de Auxílio à Navegação foram criados para dar suporte a pessoas com algum tipo de deficiência que possa atrapalhar sua locomoção, como deficientes visuais. Estes sistemas podem possuir diferentes abordagens de auxílio como o uso de GPS, pisos inteligentes, câmeras e outros diversos sensores que de um modo geral são utilizados para captar as características do ambiente, interpretá-lo e repassar ao usuário um comando que o guie para um caminho sem obstáculos e informe o que esta em sua frente.

1.1 Objetivos

O principal objetivo deste trabalho é a implementação de um sistema de auxílio a navegação para deficientes visuais que detecte obstáculos e indique pontos de interesse presentes em um ambiente interno mapeado. A detecção de obstáculos será feita utilizando técnicas e ferramentas relacionadas a visão estereoscópica enquanto que o reconhecimento de pontos de interesse será feito utilizando técnicas de processamento digital de imagens. Os pontos de interesse de um cenário são lugares ou objetos que um usuário não familiarizado com o ambiente possa obter informações sobre sua orientação, como banheiros, portas, escadas, bancos, bebedouro, entre outros. Foram comparadas duas ferramentas de visão estereoscópica: o *Kinect* e um sistema de-

senvolvido para este projeto envolvendo duas webcams. Um protótipo foi implementado para a validação das técnicas e ferramentas aplicadas, onde foram realizados testes em ambientes mapeados com marcadores indicando pontos de interesse, verificando a resposta do algoritmo de detecção de obstáculos e também a acurácia do reconhecimento dos marcadores de pontos de interesse. Este protótipo não foi testado em pessoas pois o *Kinect* possui limitações de movimentação e portanto todos os testes foram realizados em ambientes determinados que possibilitaram o teste utilizando o Kinect; para atingir a mobilidade desejada, outros *hardwares* serão utilizados em trabalhos futuros.

1.2 Motivação

O deficiente visual, muitas vezes, é extremamente dependente de outras pessoas para realizar tarefas comuns do cotidiano, como se locomover, encontrar e interagir com determinados objetos. Este problema se acentua mais quando estão em ambientes desconhecidos como restaurantes, bibliotecas, lojas, entre outros. Uma solução para auxiliar o deficiente visual a se orientar melhor seria o mapeamento e implantação de marcadores contendo pontos de interesse.

Assim, justifica-se criar um protótipo de um sistema de auxílio à navegação que consiga suprir a ausência do sentido da visão do usuário com deficiência visual através de um dispositivo que detecte obstáculos e indique pontos de interesse.

Este sistema ajudará o usuário a ter mais independência e segurança, uma vez que buscará informar eventuais pontos de interesse e ao mesmo tempo buscará caminhos com o menor número de obstáculos e poderá captar e avisar em tempo real, eventuais mudanças bruscas que possam ocorrer durante o trajeto, como por exemplo, uma pessoa ou animal cruzando seu caminho.

1.3 Organização do documento

No capítulo 2 deste trabalho, serão abordados tópicos relacionados à visão computacional, especificamente na visão estereoscópica, mostrando como funciona a visão humana e sua inspiração para a aplicação desta característica em visão computacional.

No capítulo 3 serão mostradas algumas técnicas envolvendo processamento digital de imagens que são de grande importância para o desenvolvimento deste projeto.

O capítulo 4 envolverá a apresentação de trabalhos relacionados envolvendo um ou mais

dos seguintes tópicos: visão computacional, processamento de imagens, sistema de auxílio a navegação.

No capítulo 5 é apresentada a metodologia proposta neste trabalho, mostrando os objetivos principais do protótipo, o detalhamento das técnicas e ferramentas estudadas, assim como o detalhamento dos algoritmos de navegação e de detecção de pontos de interesse; por fim serão exibidos os resultados obtidos.

Por fim, o capítulo 6 trará as conclusões obtidas neste trabalho, assim como possíveis trabalhos futuros que envolvem o aprimoramento do protótipo.

Capítulo 2

VISÃO ESTÉREO

2.1 Considerações iniciais

A visão é um importante meio de vários seres vivos de captar o que acontece ao redor, e nos últimos anos, este tipo de recurso vem sendo explorado e utilizado em computadores. Segundo Szeliski (2011), a visão computacional, envolve transformar imagens em descrições estruturais de uma cena. Para conseguir se aproximar da visão da maioria dos animais é necessário possuir um sistema que explora técnicas que utilizam duas imagens como fonte de informação, tais sistemas podem ser chamados de estereoscópicos.

Este capítulo é organizado da seguinte forma: na seção 2.1, são explicados conceitos sobre a visão humana e comparações com a de outros animais, a seção 2.2 é voltada a explicar de forma mais detalhada a visão binocular. A seção 2.3 é voltada a explicar conceitos de geometria epipolar, enquanto na seção 2.4 é explicado a importância de um sistema estereoscópico bem calibrado. A seção 2.5 aborda conceitos de retificação, método que facilita a busca de correspondências, limitando a busca em apenas uma coordenada. Na seção 2.6, são apresentados métodos que auxiliam a estimação de distância. A seção 2.7 mostra o estado da arte de sistemas estereoscópicos, apontando várias situações em que este tipo de técnica pode ser utilizado. Já na seção 2.8, são apresentadas as considerações finais sobre o capítulo.

2.2 Visão humana

A visão é um dos sentidos mais importantes para a percepção da maioria dos animais, e com o ser humano não é diferente. O olho humano é, essencialmente, um órgão esférico, contendo uma abertura circular frontal, a pupila, por onde entram os raios de luz (TOMMASELLI,

2009). Os olhos humanos recebem um número gigantesco de informações contínuas vindas do ambiente, adaptando-se a mudanças de iluminação, foco e outras possíveis transições que possam ocorrer ao seu redor. Embora o número de situações de adaptações dos olhos humanos seja grande, o tempo de resposta a determinadas mudanças bruscas pode ser maior como, por exemplo, quando uma pessoa fica muito tempo em um ambiente escuro e uma exposição brusca à luz faz com que os seus olhos fiquem ”cegos”por um rápido momento. Segundo Tommaselli (2009), isso acontece porque a retina do olho humano retém uma imagem por um tempo que varia entre $1/8$ e $1/20$ segundos, baseado nesta propriedade, filmes e animações utilizam imagens estáticas que são enviadas a uma velocidade superior a 20 quadros por segundo. Na figura 2.1 é mostrado como é o olho humano.

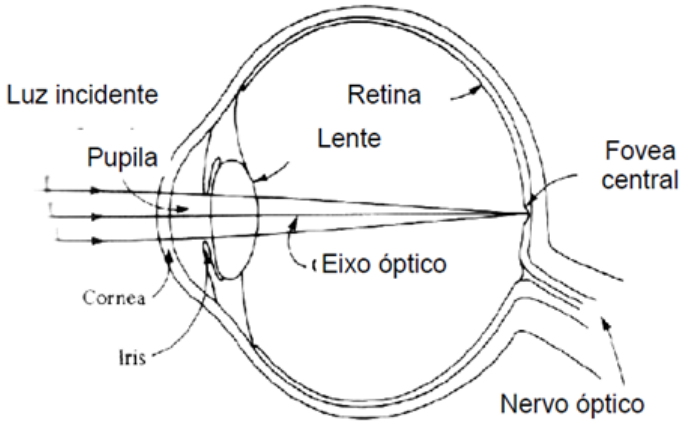

Figura 2.1: Esquema do olho humano (TOMMASELLI, 2009).

Como pode ser observado, a pupila recebe raios de luz passando pelas córneas, atingindo o cristalino, que é uma lente biconvexa, passando pela retina e chegando ao nervo óptico. A íris é uma membrana que define a coloração do olho, e sua contração e expansão controlam a quantidade de luz que entra no olho (TOMMASELLI, 2009).

Outra característica da visão humana é o poder de interpretação dos objetos que estão ao alcance de sua visão, como, por exemplo, conseguir distinguir qual objeto está mais próximo ou distante sem a necessidade de um esforço significante ou de um grande tempo de resposta. Este tipo de interpretação pode ser encontrado também em outros animais, onde o campo de visão pode ser maior ou menor comparado ao sistema de visão humano, esta variação é uma questão evolutiva, onde, por exemplo, certos animais têm os olhos posicionados na lateral da cabeça, proporcionando um campo de visão maior para conseguir enxergar predadores por perto, enquanto outros têm um campo de visão menor, já que seus olhos estão posicionados na frente da cabeça, porém possuindo a vantagem de estimar a distância e focar suas presas com maior facilidade. A figura 2.2 mostra os diferentes campos de visão encontrados na natureza onde é

possível observar que o campo de visão de um coelho é maior que a de um ser humano, pois seus olhos estão posicionados em locais diferentes.

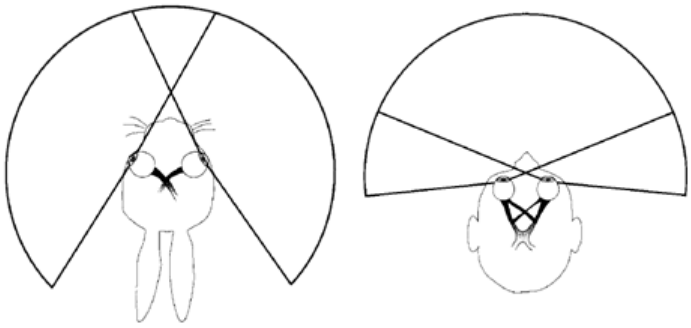

Figura 2.2: Comparação do olho humano com o de um coelho (TOMMASELLI, 2009).

Já a visão monocular conta com elementos para uma percepção rudimentar da profundidade, valendo-se apenas das leis da perspectiva, onde o tamanho aparente dos objetos diminui à medida que esses se afastam do observador. Assim, os objetos mais próximos acabam escondendo, atrás de si, os objetos mais distantes que se encontram sobre o mesmo eixo de perspectiva.

2.3 Visão binocular

O conceito de visão estereoscópica, ou também chamada de binocular, utilizado nas câmeras é baseado no sistema de visão humano, onde através de duas lentes, posicionadas paralelamente em relação aos seus eixos, captam imagens de diferentes perspectivas, e quando juntas, podem formar um ambiente 3D através da interpretação de nosso cérebro. A Figura 2.3 demonstra como o cérebro interpreta e realiza a fusão de duas imagens captadas pelos olhos, onde cada olho obtém uma imagem da cena, e as duas são sobrepostas e o cérebro as transforma em uma imagem única.

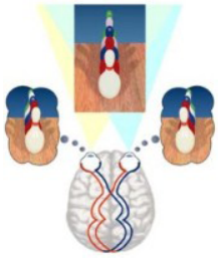

Figura 2.3: Interpretação do cérebro humano (SEUBA; JOAQUIM, 2009).

A visão binocular permite que seja obtida a visão em 3D, possibilitando estimar distâncias entre os objetos dentro do campo de visão, assim como a distância de um objeto para o observador; diferente da visão monocular, que utiliza elementos de percepção rudimentar da profundidade, utilizando apenas conceitos de perspectiva, onde o tamanho dos objetos é levado em consideração, onde um objeto mais próximo esconde atrás de si outros objetos que estão mais distantes (KIRNER; TORI, 2004). A Figura 2.4 mostra a perspectiva de cada olho num mesmo cenário.

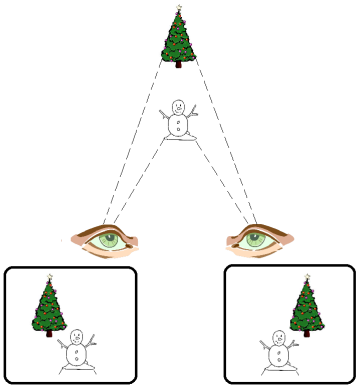

Figura 2.4: Perspectiva de cada olho numa cena com dois objetos.

É possível observar que a posição e o tamanho de cada objeto de acordo com a visão obtida, faz com que seja possível interpretar qual objeto está mais próximo e qual está mais distante.

2.4 Geometria epipolar

A geometria epipolar é uma propriedade intrínseca entre duas imagens, onde seus planos possuem uma relação geométrica de intersecção com o conjunto de planos epipolares, onde suas propriedades não dependem da estrutura da cena, e sim dos parâmetros internos e de posição dos pontos de observação (AIRES, 2010). Dado um ponto tridimensional *X*, e dois pontos de observação *C* e *C'*, são obtidas duas imagens onde os seus pontos *x* e *x'*, respectivos de *C* e *C'*, são a posição da projeção do ponto *X* em cada imagem. Estes cinco pontos citados pertencem ao plano epipolar, enquanto *l* e *l'* representam as linhas epipolares referentes as respectivas projeções *x* e *x'*, como pode ser visto na Figura 2.5 . Utilizando as linhas epipolares é possível descobrir a posição de equivalência entre as duas projeções, assim a busca de pontos correspondentes entre projeções fica restrita apenas a percorrer as linhas epipolares (HARTLEY, 1999). Este processo será melhor detalhado no capítulo de proposta do projeto.

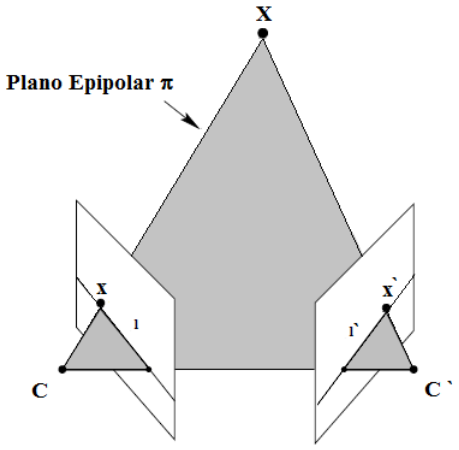

Figura 2.5: Plano epipolar. Adaptado de Karlstroem (2007).

2.5 Calibração

A calibração é um passo importante para construir um sistema estéreo eficiente, pois com um sistema estéreo bem calibrado, é possível obter os parâmetros de cada câmera e assim conseguir calcular e obter recursos como a matriz de disparidade, as linhas epipolares e imagens retificadas que poderão ser utilizadas no processo de estimação de distâncias de objetos em relação às câmeras; neste contexto, existem dois tipos de calibração em sistemas estéreos (COELHO; TAVARES, 2003):

Parâmetros extrínsecos: São parâmetros obtidos para determinar a posição das câmeras em relação ao sistema de coordenadas globais (coordenadas 3D), ou seja, esta calibração tenta obter o posicionamento entre as duas câmeras, como, por exemplo, a distância entre elas, a regulagem de altura e o posicionamento angular que as câmeras devem estar alinhadas.

Parâmetros intrínsecos: Esta calibragem busca por parâmetros internos da câmera como, por exemplo, a distância focal, que é a representação da distância, em pixels, entre o centro de projeção e o plano da imagem; o centro óptico da imagem e as distorções da lente de cada câmera.

Segundo Zhang (2000), o procedimento para realizar uma calibragem pode ser seguido por:

1. Imprimir uma imagem de modelo padrão e colocá-la em uma superfície plana.
2. Tirar algumas fotos do modelo padrão em diferentes orientações, podendo mover tanto a câmera, quanto o modelo.

3. Detectar pontos de destaque na imagem.
4. Estimar os parâmetros intrínsecos e extrínsecos.
5. Estimar os coeficientes de distorção.
6. Refinar todos os parâmetros por minimização.

O processo de calibração será melhor detalhado no capítulo 5, onde serão mostrados os métodos estudados para este projeto.

2.6 Retificação

Dado um par de imagens estéreo, o processo de retificação determina uma transformação de cada plano de imagem de modo que os pares de linhas epipolares fiquem colineares e paralelos ao seu eixo (FUSIELLO; TRUCCO; VERRI, 2000). Com imagens retificadas, a busca por pixels semelhantes entre as duas imagens fica restrita apenas ao eixo x, ou seja, uma busca em 1-D; a Figura 2.6 exemplifica uma imagem em que é aplicado o processo de retificação, comparando com a própria imagem antes da aplicação do processo de retificação; é possível perceber que na imagem de baixo, as imagens das duas câmeras estão alinhadas.

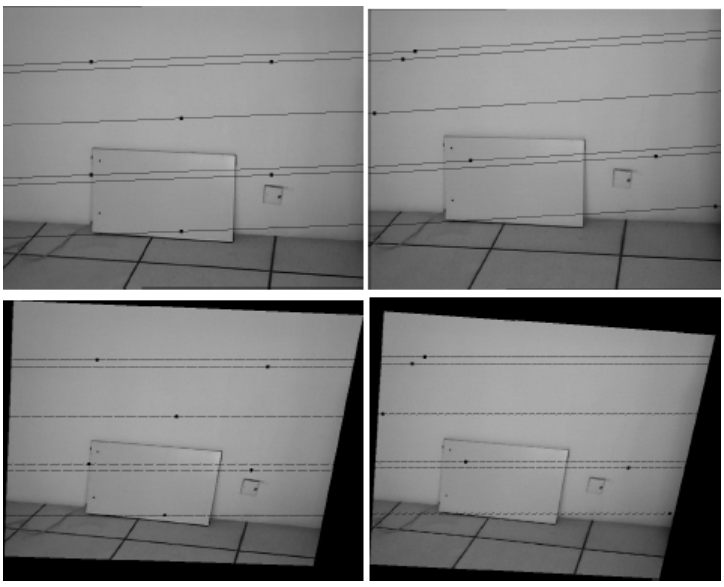

Figura 2.6: Comparação de imagens não retificadas e imagens retificadas. Adaptado de Zhu et al. (2007).

Embora seja possível realizar o processo de retificação sem a calibração das câmeras, ainda é necessário obter certos parâmetros para realizar este método, como pode ser visto em Kumar et al. (2010).

2.7 Estimação de distância

Uma das grandes vantagens de sistemas com duas câmeras em relação a sistemas com apenas uma câmera, é a possibilidade de estimar a distância entre seus centros ópticos e de objetos presentes em seu campo de visão. Com um sistema estéreo bem calibrado e com suas imagens retificadas, é possível obter valores de distância utilizando determinados métodos, dos quais é possível destacar:

2.7.1 Disparidade

Segundo Marques e Luis (1998), disparidade é a diferença de valores de dois pontos equivalentes (epipolares) entre duas imagens, onde o cálculo desta distância pode ser mostrado como:

$$d = x_E - x_D \tag{2.1}$$

Onde x_E é o pixel da imagem esquerda e x_D é o pixel da imagem direita. Com o valor da distância e de parâmetros obtidos pela calibração é possível estimar a profundidade por:

$$Z = \frac{b.f}{d} \tag{2.2}$$

Onde Z é a profundidade, b é a base(distância entre as duas câmeras) e f é a distância focal. Utilizando o calculo de profundidade em todos os pixels presentes nas duas imagens, é possível obter o mapa de disparidade completo como é observado na Figura 2.7 , onde quanto mais claro está um objeto, mais próximo o mesmo está da câmera, formando um mapa onde cada pixel possui um valor de 0 a 255 na escala de cinzas.

Figura 2.7: Imagens estereoscópicas e seu respectivo mapa de disparidade (FERNANDES et al., 2010).

2.7.2 Triangulação

A triangulação é um método de estimação de pontos 3D, possibilitando o cálculo de distância entre a linha base das câmeras e um dado objeto P. Utilizando conceitos de trigonometria, como a lei dos senos, é possível chegar a distância desejada. A Figura 2.8 mostra um triangulo formado pelas câmeras L e R, e o objeto P, onde D é a distância entre o objeto e a linha base das duas câmeras b; e as distâncias entre cada câmera e o objeto são representadas por l e r. Através do método da lei dos senos é possível obter o valor de D calculando:

$$r = \frac{b.sen(\widehat{L})}{sen(\pi - \widehat{L} - \widehat{R})} \tag{2.3}$$

$$D = \widehat{L}.sen(\widehat{R}) \tag{2.4}$$

Este cálculo só é possível ser utilizado com sistemas calibrados e com os centros focais das duas câmeras paralelos, permitindo encontrar o valor da linha base b.

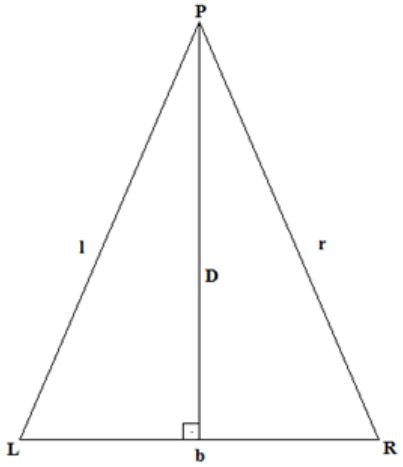

Figura 2.8: Triangulo utilizado para o cálculo de distância por triangulação (GARDIMAN, 2008)

2.8 Considerações finais

Neste capítulo, foram reunidas características da visão estereoscópica, como o seu funcionamento em seres vivos, e posteriormente, foram mostradas técnicas para preparação e utilização de duas câmeras em visão computacional. No próximo capítulo, serão abordados temas relacionados às técnicas utilizadas em visão computacional e processamento digital de imagens, que poderão ser integradas a técnicas de visão estereoscópica.

Capítulo 3

TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

3.1 Considerações iniciais

Como foi visto no capítulo anterior, a visão computacional é uma área muito explorada atualmente, e com a sua utilização, são necessárias técnicas que trabalhem na imagem captada pela visão para transformar os dados brutos vindos de uma imagem e dados relevantes ao contexto em que a captação da imagem está inserida. Estas técnicas podem ser comparadas ao cérebro humano, como foi mostrado no capítulo anterior, os olhos humanos captam as imagens e cabe o cérebro interpretá-las.

Segundo Filho e Neto (1999), o processamento digital de imagens pode ser dividido em duas categorias: (1) o aprimoramento de informações de imagens para interpretação humana; e (2) a análise automática por computador de informações extraídas de uma cena. Neste capítulo serão abordadas as técnicas mais utilizadas no contexto de captura e tratamento de imagens utilizando processamento digital de imagens.

Técnicas de processamento são utilizadas para o tratamento preliminar dos dados brutos para aprimorar a qualidade das imagens, como por exemplo, a correção de distorções, eliminação de ruídos, a calibração da radiometria das imagens, diminuição de brilho para posteriormente serem devidamente realçadas e classificadas. A maioria das operações efetuadas em processamento de imagens trabalham diretamente com valores de intensidades de *pixels* (FILHO; NETO, 1999).

3.2 Tipos de imagens

Em processamento de imagens, os tipos de imagens mais trabalhados são as imagens coloridas, imagens em tons de cinza e por último as imagens em preto e branco. Cada tipo de imagem possui valores de *pixels* e níveis diferentes, o que fará com que cada uma se enquadre em diferentes tipos de necessidade para aplicações, como por exemplo, processamento de imagem em tempo real quererem imagens com poucos níveis de intensidade.

Segundo Queiroz e Gomes (2006), uma imagem monocromática é uma função bidimensional continua $f(x,y)$, no qual x e y são coordenadas espaciais, e o valor f é proporcional a intensidade luminosa do ponto escolhido, e já que computadores não tem capacidade de processar imagens continuas, é necessário representa-las através de *arrays* de números digitais, e cada ponto deste arranjo é denominado *pixels*.

3.2.1 Imagens coloridas

Uma imagem digital colorida no sistema *RGB (Red, Blue e Green)*, possui *pixels* que são representados por vetores no qual seus componentes representam as intensidades de vermelho, verde e azul. De acordo com Queiroz e Gomes (2006), uma imagem colorida $f(x,y)$ pode ser representada por:

$$f(x,y) = f_R(x,y) + f_G(x,y) + f_B(x,y) \tag{3.1}$$

Onde:

$f_R(x,y)$ representa a intensidade vermelha, $f_G(x,y)$ representa a intensidade verde, e por fim $f_B(x,y)$ representa a intensidade azul.

Na figura 3.1 é possível verificar como uma imagem colorida *RGB* é composta, um somatório de 3 planos monocromáticos, onde cada plano possui seu respectivo tom:

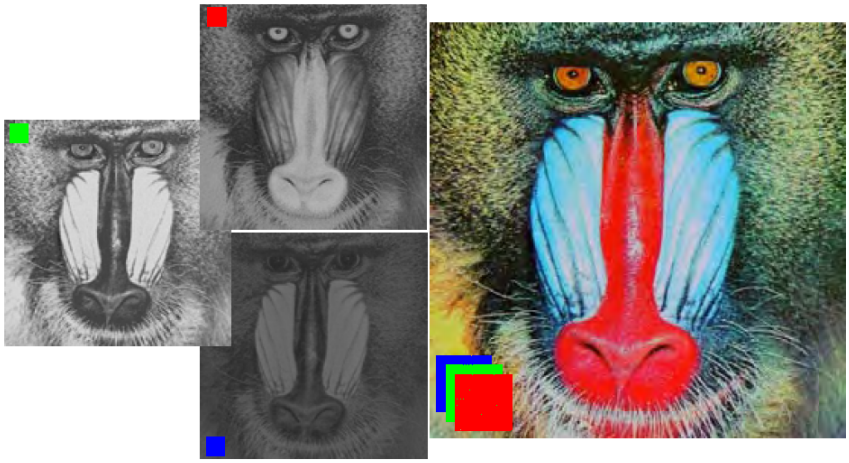

Figura 3.1: Representação de uma imagem colorida (QUEIROZ; GOMES, 2006).

3.2.2 Imagens em escala de cinza

Diferente das imagens *RGB*, imagens em tons de cinza possuem apenas um plano monocromático, e sua resolução pode variar dependendo o número de bits que a imagem possui, por exemplo uma imagem de 8 bits possui 256 níveis de cinza (PEDRINO, 2003), ou seja, os valores presentes na matriz da imagem podem varia de 0 a 255, onde 0 representa a cor preta e 255 representa a cor branca, e os valores entre eles são os níveis de cinza, portanto quanto mais níveis de cinza a imagem possuir, maior será o detalhamento da imagem.

Imagens com 256 níveis de cinza são normalmente usadas em técnicas de processamento de imagem por possuírem apenas um plano, o que resulta em dados menores a serem manipulados, porém mantendo as informações relevantes possíveis de serem acessadas. A Figura 3.2 mostra uma imagem em escala de cinza com 256 níveis:

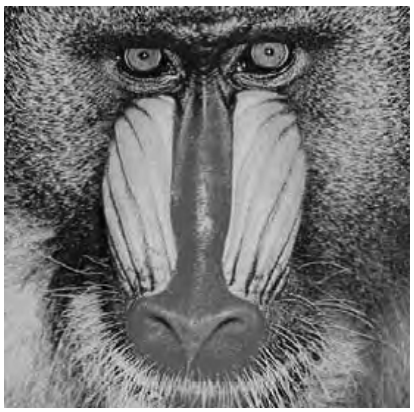

Figura 3.2: Representação de um imagem em escala de cinza com 256 níveis (QUEIROZ; GOMES, 2006).

3.2.3 Imagens em preto e branco

São as imagens em escala de cinza mais simples em termos de dados, onde possuem apenas dois valores possíveis: 0 (zero) que representa a cor preta e 1 (um) que representa a cor branca, por isso esse tipo de imagem é conhecido como imagem binária.

Este tipo de imagem é geralmente utilizado para aplicações que necessitam rapidez em sua resposta, como por exemplo o reconhecimento de placas de automóveis assim como podem ser imagens resultantes de operações de processamento, como por exemplo a segmentação, detecção de bordas ou aplicações de máscaras. A Figura 3.3 pode exibir como é uma imagem no formato binário, onde a imagem é formada por uma matriz binária.

Figura 3.3: Representação de um imagem binária (QUEIROZ; GOMES, 2006).

3.3 Segmentação

A segmentação de imagens se baseia em dividir uma imagem em unidades significativas, ou seja, em regiões com objetos de características potenciais. Normalmente, algoritmos de segmentação devem parar apenas quando um objeto de interesse é isolado, como por exemplo uma letra do alfabeto inserida em uma placa de um automóvel. Segundo (GONZALEZ; WOODS, 2000), a segmentação de imagens não triviais é uma das tarefas mais difíceis em processamento de imagem, pois vários fatores podem atrapalhar a precisão, como por exemplo a semelhança de objetos, onde um dos objetos detectados e isolados não é exatamente aquele desejado.

O algoritmo de Otsu (OTSU, 1975) é um dos mais utilizados em processamento de imagens, pois traz bons resultados de segmentação utilizando imagens binárias originárias de imagens em tons de cinza. Segundo Facon (2004), o algoritmo, chamado de limiarização bimodal de Otsu

é baseado na análise de discriminante, onde a limiarização particiona os *pixels* de uma imagem em duas classes C_0 e C_1, que representam respectivamente um objeto e o fundo de uma imagem, desta forma:

$$C_0 = \{0, 1, ..., t\} \tag{3.2}$$

$$C_1 = \{t+1, t+2, ..., l\} \tag{3.3}$$

Onde t é o nível de cinza, em que minimização para se obter um ótimo limiar pode ser dada por:

$$\eta = \frac{\sigma_B^2}{\sigma_T^2} \tag{3.4}$$

Onde σ_B^2 é a variância da classe e σ_T^2 é a variância total.

Sendo que:

$$\sigma_T^2 = \sum_{i=0}^{l-1} (i - \mu T)^2 P_i \tag{3.5}$$

$$\mu T = \sum_{i=0}^{l-1} i P_i \tag{3.6}$$

$$\sigma_B^2 = \omega_0 \omega_1 (\mu_1 \mu_0)^2 \tag{3.7}$$

$$\sigma_0 = \sum_{i=0}^{t} P_i \tag{3.8}$$

$$\omega_1 = 1 - \omega_0 \tag{3.9}$$

$$\mu_1 = \frac{\mu T - \mu_t}{1 - \mu_0} \tag{3.10}$$

$$\mu_0 = \frac{\mu_t}{\omega_0} \tag{3.11}$$

$$\mu_t = \sum_{i=0}^{t} iP_i \tag{3.12}$$

$$P_i = \frac{n_i}{n} \tag{3.13}$$

Onde n_1 é o número de *pixels* com níveis de cinza i e n é o número total de *pixels* da imagem. Os pesos ω_0 e ω_1 representam as classes de probabilidade que indicam onde a área de cada classe está ocupada. As classes μ_0 e μ_1 servem como estimativa de níveis médios em níveis de cinza e P_i é a probabilidade de ocorrência de nível de cinza i.

Ao encontrar o limiar de níveis de cinza entre as classes, é possível separá-las e isolá-las através da binarização.

Uma aplicação em *MATLAB* pode explicar o algoritmo de Otsu na prática, como pode ser visto na sequência de figuras abaixo, onde na Figura 3.4 é possível verificar uma imagem original em escala de cinza de grãos de arroz espalhada.

Figura 3.4: Imagem de grãos de arroz em tons de cinza.

Na imagem, é possível notar que a imagem está em tons de cinza, e como existem duas classes, o grão de arroz e o fundo, é gerada a imagem binária como pode ser visto na Figura 3.5:

Figura 3.5: Imagem de grãos de arroz binarizada.

Quando a imagem é binarizada, é possível notar apenas dois valores de *pixels* na imagem: 0 e 1. Com isso é possível verificar e contabilizar componentes com valores 1 na imagem. Na Figura 3.6, através de um método, é possível escolher um determinado componente encontrado na cena e isolá-lo, neste caso, o componente 50 foi o escolhido.

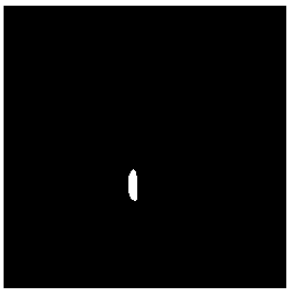

Figura 3.6: Componente detectado e isolado.

O maior desafio do algoritmo é quando um cenário possui dois ou mais objetos que estão muito próximos ou encostados uns nos outros, dificultando a segmentação e posteriormente a identificação de cada um de forma separada, pois como os objetos estão conectados, normalmente são interpretados como um objeto singular.

3.4 Técnicas de detecção e extração de características

Algoritmos de detecção e extração de características são muito utilizados em processamento de imagem, pois conseguem extrair informações precisas de um objeto em uma cena, através apenas de uma imagem. Estas técnicas podem ser utilizadas para a detecção de placas veiculares, classificação de espécies de peixes, identificação de placas de trânsito, mapeamento de degradação em florestas por meio de fotos aéreas, detecção e reconhecimento de rostos, etc.

Dentre os algoritmos mais utilizados para a detecção e extração de características, podem ser citados os algoritmos SIFT(LOWE, 2004), SURF(BAY; TUYTELAARS; G, 2006) e FAST(TRAJKOVIĆ; HEDLEY, 1998).

3.4.1 Algoritmo SIFT

O algoritmo SIFT é um dos mais populares em visão computacional devido a sua robustez na detecção e extração de características, utilizando imagens em diferentes escalas, rotações e iluminações (GUERRERO, 2011). O SIFT (*Scale Invariant Feature Transform*) foi desenvolvido por David Lowe (LOWE, 2004), onde seu artigo apresenta um método para a detecção de

características de invariâncias distintivas que podem ser utilizadas para realizar a comparação de objetos com diferentes ângulos, rotações e escalas em uma cena.

De acordo com Guerrero (2011), o algoritmo SIFT possui quatro estágios principais:

1. Detecção extrema de espaço-escala;
2. Localização de pontos-chave;
3. Determinação de orientação;
4. Descritor de pontos-chave.

Sendo que a transição desses estágios é feita de forma de cascata, funcionando de forma robusta, porém significantemente lento em relação a outras técnicas. A técnica utiliza uma comparação com vários tamanhos diferentes (pirâmide de imagens) utilizando a diferença da Laplaciana da Gaussiana, comparando os *pixels* e seus vizinhos de uma imagem com uma com escala diferente, e quando são encontrados a mesma área de *pixels* iguais em escalas diferentes, essas são classificadas como áreas de pontos-chave.

A Figura 3.7 demonstra na pratica a extração e detecção de características de um objeto alvo e de uma cena contendo esse objeto, porém em outra escala(menor) e com outra orientação; apesar de ocorrerem algums pontos falso-positivos, é possível notar grande precisão no reconhecimento do objeto em uma cena com outros objetos.

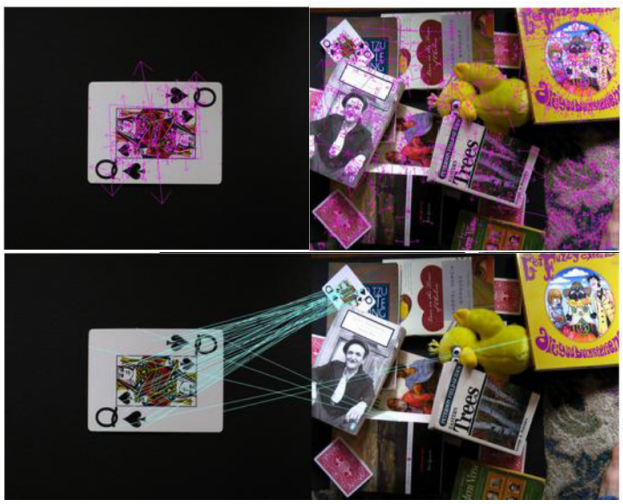

Figura 3.7: Demonstração do algoritmo SIFT(adaptado de (GUERRERO, 2011).

3.4.2 Algoritmo SURF

A criação do algoritmo SURF(*Speed-Up Robust Feature*) é de Bay (BAY; TUYTELAARS; G, 2006) e é uma alternativa ao algoritmo SIFT, pois este embora robusto, tinha como característica um alto custo de tempo, e para aplicações que necessitavam de processamentos mais rápidos, poderia não responder satisfatoriamente. O SURF de um modo geral pode ser considerado tão robusto e mais rápido que o SIFT, um dos fatores de seu melhor tempo de resposta é o fato do método possuir três fases, uma a menos que o SIFT que possui quatro:

1. Detecção;
2. Descrição;
3. Comparação.

Comparando com o algoritmo SIFT, o SURF utiliza a comparação de pirâmide de imagens com a Laplaciana da Gaussiana com a adição de um filtro de caixa, ou seja filtros mais radicais, sem áreas de transições, como era no algoritmo SIFT, esses filtros podem ser observados na Figura 3.8, onde é mostrada a diferença entre eles sendo que os filtros **a** e **b** são filtros do algoritmo SIFT e os filtros **c** e **d** são filtros de caixa, utilizados no algoritmo SURF.

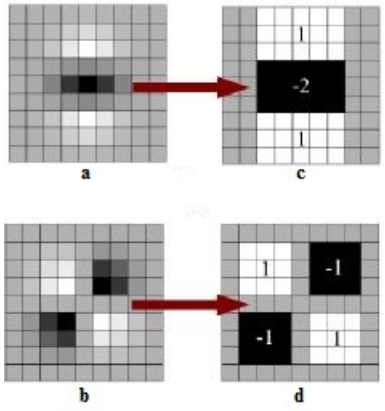

Figura 3.8: Comparação dos filtros do SIFT e SURF.

Segundo Guerrero (2011), a utilização do filtro de caixa, deixa o processo de detecção mais rápido porém com chances da precisão cair, já que não consideradas as intensidades nas regiões de transição. A fase de descrição é baseada na Transformada de Haar junto a imagem integral funcionando de forma eficiente com custo computacional reduzido (BORTH et al., 2013) e pôr fim, a fase de comparação realiza a detecção de segmentos de correspondência entre as duas imagens (cena, e objeto).

3.4.3 Algoritmo FAST

O FAST(*Features from Accelerated Segment Test*) foi criado em 1998 por Trajkovic (TRAJKOVIĆ; HEDLEY, 1998), e é um método de detecção de cantos, utilizada para extrair pontos de características. Segundo Guerrero (2011), a escolha de cantos se deu pelo fato de que os eles são um dos tipos mais intuitivos de características que mostram uma mudança significativa de intensidade facilitando a comparação de pontos vizinhos e seus principais critérios de satisfação são:

- Consistência, onde os pontos devem manter um padrão em relação a variação de intensidade;
- Precisão, onde os cantos devem ser detectados os mais próximos possíveis das posições;
- Velocidade, onde a detecção de cantos deve ser feita de maneira rápida o suficiente.

De acordo com o estudo de Trajković e Hedley (1998), os algoritmos até então conseguiam atender os dois primeiros critérios, então foi desenvolvido um algoritmo para satisfazer os três critérios. Dada a Figura 3.9, que mostra a detecção de um canto através da circulação de um dado *pixel*, onde é possível observar os *pixels* que circulam o *pixel* são numerados de 1 a 16:

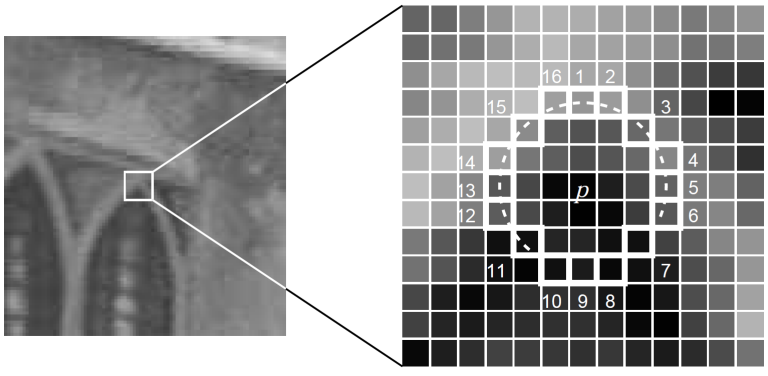

Figura 3.9: Detecção de característica através do algoritmo FAST(VISWANATHAN, 2011).

O algoritmo FAST funciona da seguinte maneira (VISWANATHAN, 2011):

1. Seleciona um *pixel p* na imagem;
2. Indica um valor de limiar de intensidade;

3. Considera-se um círculo com 16 *pixels* que estão em volta do *pixel p*;

4. Ao menos *N pixels* vizinhos precisam ter intensidade diferente da intensidade do *pixel p*, onde $N = 12$;

5. Compara-se a intensidade dos vizinhos 1,5,9 e 13 com a intensidade do *pixel p* e pelo menos três desses quatros *pixels* devem satisfazer o limiar para existir um ponto de interesse.

6. Caso dois ou mais *pixels* desses quatro não satisfaçam o limiar, o *pixel p* não é um ponto de interesse. Caso contrário, checa-se os *pixels* restantes e se ao menos 12 deles satisfazerem o critério, então *p* é um ponto de interesse e consequentemente um canto.

7. Repetir o processo com todos os *pixels* restantes da imagem.

3.5 Técnicas de detecção e extração de características

Segundo Filho e Neto (1999), o princípio básico da morfologia matemática consiste na extração de informações geométricas e topológicas de um conjunto desconhecido em uma imagem, através de um conjunto definido, conhecido como elemento estruturante.

A morfologia pode ser aplicada em várias áreas de processamento de imagem, como o realce, filtragem, segmentação, detecção de bordas, entre outras. Dentre as diversas operações utilizadas na morfologia, duas podem ser consideradas as operações básicas, a dilatação e a erosão, e através delas é possível realizar operações mais complexas (PEDRINO, 2003).

3.5.1 Dilatação

Sejam A e B conjuntos no espaço Z^2, e $\emptyset$ um conjunto vazio. A denotação da dilatação de A por B é $A \oplus B$ e sua definição é:

$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \emptyset\} \tag{3.14}$$

Sendo assim, o processo primeiramente obtém a reflexão de B em torno de sua origem e depois deslocar essa reflexão de x. A dilatação de A por B, é consequentemente o conjunto de todos os deslocamentos x, onde a intersecção de B refletido deslocado de x e A, incluindo um elemento diferente de 0, por fim, o conjunto B é denominado elemento estruturante da operação (PEDRINO, 2003).

A Figura 3.10 mostra a operação de dilatação de um conjunto A, com três elementos estruturantes B diferentes; é perceptível a dilatação, ou expansão da imagem após a operação dependendo do elemento estrutural, que dita pra qual sentido a dilatação é feita.

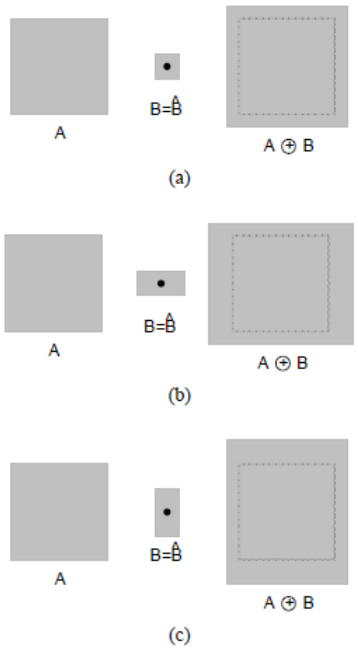

Figura 3.10: Processo de dilatação (FILHO; NETO, 1999).

3.5.2 Erosão

O processo de erosão faz o oposto da dilatação, isto é, encolhe o objeto da imagem. Sejam A e B conjuntos no espaço Z^2 e a denotação da dilatação de A por B é $A \ominus B$, sua definição é:

$$A \ominus B = \{(B)_x \subseteq A\} \tag{3.15}$$

Onde, a erosão de A por B resulta num conjunto de deslocamentos x, tais que B, esteja contido em A (PEDRINO, 2003).

A Figura 3.11 exibe as operações de erosão, com três tipos de elementos estruturantes B diferentes, onde é possível observar o encolhimento da imagem dependendo do elemento estruturante.

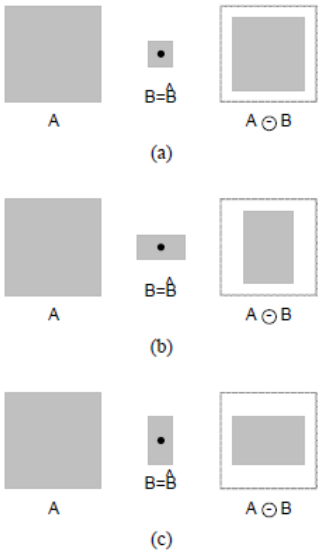

Figura 3.11: Processo de erosão (FILHO; NETO, 1999).

3.6 Filtros de realce

Segundo Gonzalez e Woods (2000), o principal objetivo de técnicas de realce é processar uma imagem de forma que o resultado seja mais relevante para uma aplicação do que a imagem original. Dentre os tipos de técnicas de realce, podem ser divididas em duas grandes categorias: técnicas no domínio espacial e técnicas de domínio da frequência, onde a primeira se refere ao próprio plano da imagem e a segunda são baseadas nas Transformadas de Fourier. No contexto de técnicas de domínio da frequência, dois filtros podem ser citados como os mais utilizados: filtro Passa-baixas e filtro Passa-altas.

3.6.1 Filtro Passa-baixas

Um filtro Passa-baixas é utilizado para eliminar altas frequências de uma imagem, enfatizando áreas homogêneas de tons similares, e consequentemente servem para suavizar a aparência de uma imagem, contudo a resolução da imagem é diminuída, podendo ocorrer a perda de informações (PEDRINO, 2003). Sua utilização mais comum é na remoção de ruídos eletrônicos em imagens de sensoriamento remoto.

Um tipo comum de filtro passa-baixas é denominado Filtro de Média, no qual consiste em substituir um *pixel* central pela média aritmética do *pixel* e seus vizinhos através de máscaras de pesos, como pode ser observado na figura 3.12, onde é possível notar a mudança dos valores dos pixels de uma imagem de acordo com sua média (PEDRINO, 2003).

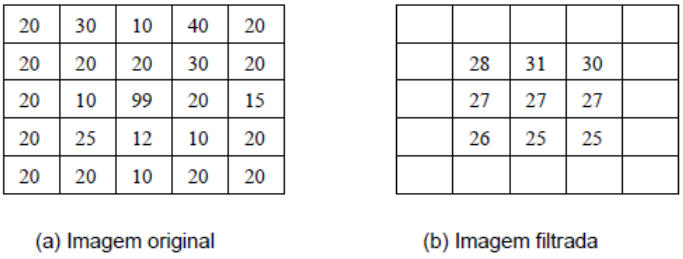

Figura 3.12: Filtro passa-baixas (PEDRINO, 2003).

3.6.2 Filtro Passa-altas

Um filtro passa-altas faz o contrário de um filtro passa-baixas, isto é, realça a aparência dos detalhes finos de uma imagem, sendo utilizado para realçar de bordas. Segundo Pedrino (2003), uma utilização para um filtro passa-altas, é utilizar o filtro passa-baixas e subtrair o resultado obtido pela imagem original.

Um tipo comum de filtro passa-altas são os Filtros Laplacianos, que são geralmente utilizados na detecção de bordas, onde a soma dos pesos usados na máscara é igual a zero. Existem uma infinidade de filtros laplacianos, entre eles, podem ser destacados na Figura 3.13:

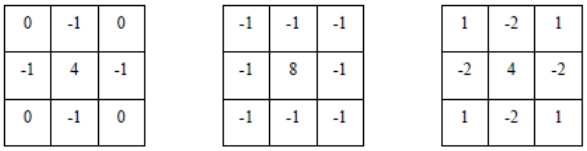

Figura 3.13: Máscaras de filtros Laplacianos para Passa-altas (PEDRINO, 2003).

3.7 Técnicas de detecção de bordas

Segundo Filho e Neto (1999), define-se borda, como a fronteira entre duas regiões cujos níveis de cinza são diferentes, indicando uma possível mudança de luminosidade, cor, ou textura. Para a detecção de bordas, a maioria das técnicas utiliza um operador local diferencias, isto é, máscaras, como foi visto na técnica FAST.

Dentre as várias técnicas, podem ser citadas: Roberts (ROBERTS, 1963), Sobel (SOBEL, 1990), Canny (CANNY, 1986), que serão brevemente apresentadas a seguir e posteriormente serão comparadas no capítulo da descrição do projeto.

3.7.1 Algoritmo Roberts

Um dos primeiros algoritmos de detecção de bordas, utilizando uma matriz $2x2$ como máscara, onde Gx e Gy são as máscaras, sendo que a segunda nada mais é que a rotação em 90^o da primeira.

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \tag{3.17}$$

Onde o cálculo do gradiente de intensidade (variações abruptas) para a avaliação de borda é dado por:

$$|G| = Gx^2 + Gy^2 \tag{3.18}$$

E o cálculo da direção da borda é dado por:

$$\alpha = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.19}$$

3.7.2 Algoritmo Sobel

O algoritmo de Sobel utiliza as mesmas formulas do algoritmo de Roberts para encontrar o gradiente e direção de possíveis bordas, porém sua máscara é de dimensão $3x3$, aumentando a robustez em detecção de bordas e contornos. A máscara de Sobel é dada por:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.20}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{3.21}$$

3.7.3 Algoritmo Canny

O algoritmo Canny de detecção de bordas que utiliza um filtro de convolução que filtra ruídos e localiza bordas. Segundo Canny (1986), um algoritmo de detecção de bordas deve possuir os seguintes objetivos:

- O algoritmo deve possuir uma baixa taxa de erros, onde deve detectar apenas bordas, com nenhuma borda faltando;
- O algoritmo deve indicar uma distância mínima entre os *pixels* de borda da imagem original e o os *pixels* detectados como borda, isto é, deve manter uma boa localização;
- Cada borda da imagem deve ser marcada uma vez, evitando duplicação de *pixels* de bordas.

Os passos do algoritmo de Canny podem ser descritos como:

1. Remoção de Ruídos através de um filtro Gaussiano;
2. Detecção de gradientes e suas respectivas direções;
3. Buscar *pixels* de maior valor, descartando os de valor mais baixos;
4. Classificar bordas fortes e bordas fracas;
5. Incorporar bordas fracas próximas as bordas fortes.

A Figura 3.14 mostra o resultado de uma imagem com as bordas detectadas utilizando o algoritmo de Canny, onde é possível observar a detecção do contorno do cubo e o contorno de seus quadrados, porém com alguns ruídos:

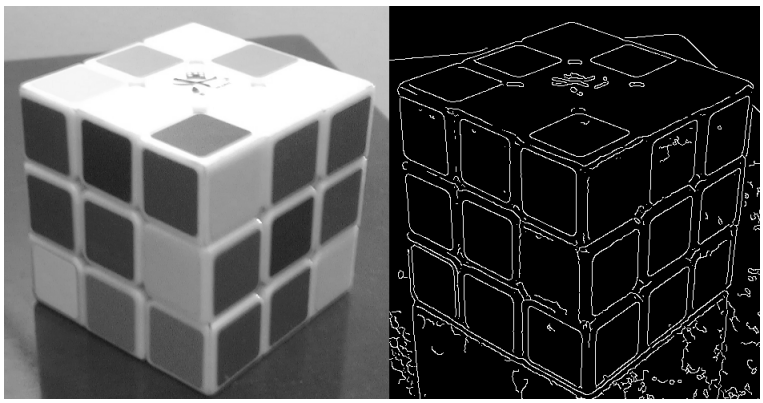

Figura 3.14: Imagem original e imagem com bordas detectadas utilizando o algoritmo de Canny.

3.8 Considerações finais

Neste capítulo foram apresentadas algumas técnicas existentes em processamento de imagem, desde sua aquisição, pré-processamento, processamento, interpretação e utilização dos dados gerados a partir de imagens. Estas técnicas são de muita importância para este projeto, sendo que grande parte das citadas foram estudadas e utilizadas para o desenvolvimento do algoritmo de auxílio a navegação, desde a parte de detecção de caminhos livres, até a extração e identificação de pontos de interesse. No próximo capítulo, serão exibidos e detalhados os trabalhos relacionados com este projeto.

Capítulo 4

TRABALHOS RELACIONADOS

4.1 Considerações iniciais

Como foi visto nos capítulos anteriores, a visão computacional possui técnicas como a estereoscópica, que podem ser utilizadas para diferentes finalidades, e em muitas vezes, trabalhando com outros tipos de sensores. Este capítulo tem como objetivo mostrar diferentes tipos de utilização de visão computacional que possuem alguma técnica de processamento de imagem, aquisição de imagem, interpretação de dados ou interface de comunicação relacionados com deficientes visuais que podem ser relacionados ao trabalho proposto.

4.2 Conceitos relacionados

Este trabalho utiliza conceitos presentes em outros trabalhos que não estão relacionados a deficientes visuais porém possuem técnicas de navegação e extração de características que podem ser aproveitadas para o desenvolvimento do protótipo proposto.

4.2.1 Navegação autônoma de robôs

Um problema comum encontrado em robótica e automação de atividades de navegação é a falta de percepção da máquina em interpretar ambientes em tempo real de maneira semelhante ao ser humano. Utilizando sensores com câmeras estéreos e algoritmos de detecção e desvio de obstáculos, é possível automatizar a navegação de sistemas móveis; como é utilizado pela NASA com o robô de exploração *Curiosity*, que, dentre dezenas de sensores, também utiliza um par de câmeras estéreas para se locomover em terreno marciano; pois a utilização de câmeras não é dependente de variáveis de temperatura por exemplo (KAUFMAN, 2012).

Outros exemplos que podem ser citados são os trabalhos de Ahmed (2006) que utiliza câmeras estereoscópicas para navegação autônoma de veículos, como carros, para navegação em ambientes externos e também o trabalho de navegação autônoma de Hamzah, Salim e Rosly (2010) que utiliza conceitos de processamento de imagens estéreo para detecção de obstáculos utilizando um robô de navegação em ambientes internos, como mostra a figura 4.1, um robô acoplaco com câmeras e um computador.

Figura 4.1: Veículo autônomo (HAMZAH; SALIM; ROSLY, 2010).

O trabalho de CORREA (2013) apresenta um sistema de navegação autônoma para robôs móveis de vigilância em ambientes internos, onde é capaz de navegar de forma segura, eficaz no ambiente, desviando de obstáculos, reconhecer pontos de referência como portas e corredores no ambiente através de redes neurais para navegar de um recinto a outro e por fim detectar formas humanas através de um sensor térmico e a câmera RGB do *Kinect*.

A ferramenta utilizada para realizar a navegação e reconhecimento de portas e corredores foi o *Kinect*, utilizando a câmera de profundidade para reconhecer obstáculos e a câmera RGB para o reconhecimento de padrões de pontos de referência no ambiente como portas em um corredor.

O processo de mapeamento do ambiente é no formato de autômato, isto é, a transição de um ambiente para outro é representada pela transição de um nodo para outro de um autômato, onde antes de cada transição realizada é feito o reconhecimento de padrões de portas em corredores.

Para a detecção de seres humanos, foi feita uma combinação da câmera RGB do *Kinect* para o reconhecimento de formas humanas, e de um sensor térmico, o *FLIR PathFinderIR*, uma câmera que é utilizada para ambientes escuros. A Figura 4.2 mostra a visão do robô, reconhecimento um caminho livre e portas próximas onde é possível ver a câmera de profundidade do Kinect exibindo ruídos e obstáculos em preto e caminho livre em cinza.

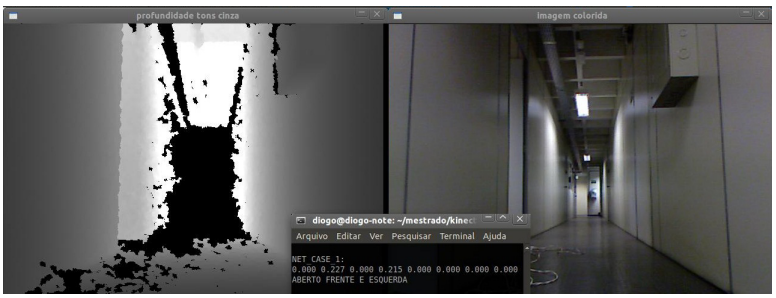

Figura 4.2: Reconhecimento de um caminho livre na câmera de profundidade e de uma porta na câmera RGB (CORREA, 2013).

4.2.2 Estimação de distância

O trabalho de Wilson (2009) busca estimar a distância de objetos em relação às câmeras estereoscópicas para utilização em diversas funcionalidades como navegação para robôs, auxilio a navegação para cegos, entre outros. A ideia consiste em obter duas fotos estéreo, e realizar uma comparação de características entre a imagem da esquerda e a imagem da direita, e assim conseguir a variação (deslocamento) em *pixels* das características das imagens. São feitas diversas comparações de distância entre as câmeras e o objeto, e foi feita uma relação entre esta distância e a distância de deslocamento dos *pixels*, como pode ser mostrado na Figura 4.3:

Variação vertical (pixels)	Variação horizontal (pixels)	Distância (cm)
−7	−16	92
−11	−11	66.7
−11	3	50.8
−11	25	38.1
−11	40	33
−19	73	29
−17	109	25
−17	137	20.5
−12	184	12.7

Figura 4.3: Comparativo entre distancias e variações de *pixels*. Adaptado de Wilson (2009).

Com valores obtidos, foi possível estimar um polinômio para o cálculo de distância:

$$D = \frac{1}{19.693x10^{-9}p^3 - 4.0361x10^{-6}p^2 + 400.83x10^{-6}p + 18.946x10^{-3}} \tag{4.1}$$

Onde p é a variação horizontal dos *pixels* e d é a distância.

Já o trabalho de Khoshelham e Elberink (2012) analisa a qualidade das informações obtidas pela câmera de profundidade do *Kinect*, fazendo testes com várias distâncias e medindo a taxa

de erro obtido.

O teste consiste em tirar várias fotos com diferentes distâncias em relação a um objeto, e depois verificar o desempenho da câmera de profundidade, medindo a taxa de erro de cada estimação.

Como resultado, foi possível concluir que quanto maior à distância, maior a incidências de erros de estimação de profundidade. Como a Figura 4.4 demonstra, quanto mais distante do objeto a ser estimado a distância, menor a área de padrão que o sensor consegue reconhecer. Foi concluído que a distância ideal para realizar mapeamento e reconhecimento de padrões com o *Kinect* foi entre 1 e 3 metros.

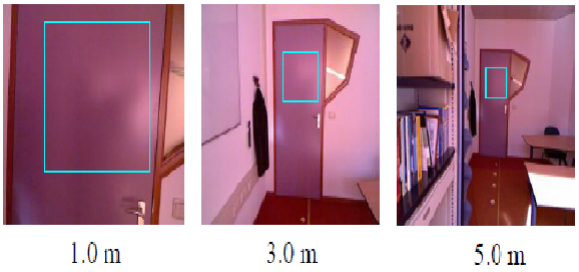

Figura 4.4: Diferentes distancias geram diferentes áreas de reconhecimento. Adaptado de Khoshelham e Elberink (2012).

4.2.3 Detecção de características

O trabalho de Nam et al. (2011), é proposto um sistema de reconhecimento de pedestres através de um robô utilizando duas câmeras como forma de captura do que acontece no ambiente. Este trabalho possibilita realizar ações como controle de multidão durante grandes eventos, sendo sensível ao contexto apresentado, reagindo de forma diferente, dependendo do fluxo de pedestres reconhecidos na cena. A figura 4.5 mostra a captação e o tratamento de uma cena com pedestres.

Figura 4.5: Detecção de pedestres (NAM et al., 2011).

O trabalho de Prasad et al. (2010) mostra a possibilidade utilizar sistemas estéreos para o reconhecimento de gestos com as mãos e rosto, utilizando duas webcams e seguindo os passos de calibração, retificação e posteriormente um mapa de disparidade, proporcionando um mapeamento de profundidade aos gestos, e assim obter uma perspectiva tridimensional e assim determinar junto a um banco de dados, qual é o gesto captado pelo sistema. A figura 4.6 mostra os passos de retificação e mapa de disparidade obtidos com a captura de um gesto de mãos.

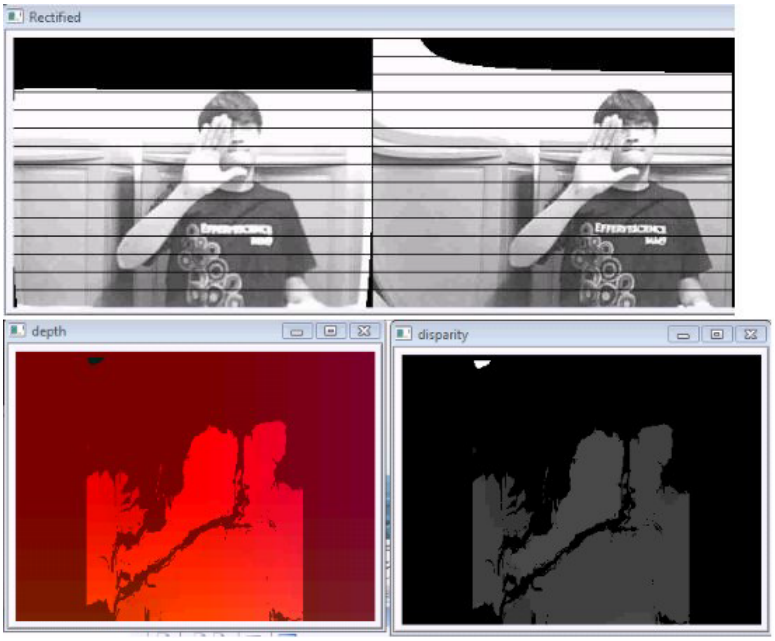

Figura 4.6: Mapa de disparidade de gestos (PRASAD et al., 2010).

Como é possível observar na imagem de profundidade obtida, o padrão feito pela mão é comparado com um banco de dados e então exibido qual o gesto aplicado.

4.3 Trabalhos relacionados

Sistemas de auxílio à navegação podem ser utilizados de várias maneiras para auxiliar deficientes visuais a se locomover, como por exemplo a utilização de câmeras estereoscópicas, GPSs e detectores de características. A seguir serão citados alguns trabalhos relacionados com este trabalho.

4.3.1 *A machine vision based navigation system for the blind*

Sistemas estéreos podem ser utilizados como ferramentas de auxílio à navegação para pessoas com problemas na visão, trabalhos como os de Tandayya, Limna e Suvonvorn (2009),

Shamsi, Al-Qutayri e Jeedella (2011) e Wenqin, Wei e Jian (2011) são exemplos de sistemas de auxílio à navegação de deficientes visuais. A Figura 4.7 mostra um exemplo de um sistema de auxílio à navegação sendo utilizado por um usuário, contendo um sistema de câmeras estéreo, fones de ouvidos acoplados a um computador dentro de uma mochila.

Figura 4.7: Protótipo de sistema de auxílio a navegação de deficientes físicos (WENQIN; WEI; JIAN, 2011).

Este sistema ajuda o usuário a ter mais independência e segurança, uma vez que buscará caminhos com o menor número de obstáculos e poderá captar e avisar em tempo real, eventuais mudanças bruscas que possam ocorrer durante o trajeto, como por exemplo, uma pessoa ou animal cruzando seu caminho.

O trabalho de Wenqin, Wei e Jian (2011) propõe um sistema que detecta obstáculos e encontra caminhos livres para o usuário utilizando câmeras estereoscópicas calibradas. Segundo Wenqin, Wei e Jian (2011), o sistema possui três objetivos:

- Extrair a região ”caminhável”do ambiente;
- Detectar obstáculos na região e medir a distância entre os obstáculos e o usuário;
- Informar o usuário através de voz sobre possíveis obstáculos.

Através de duas imagens obtidas das câmeras é realizada a detecção de cantos das imagens através do algoritmo de Sobel.

Após a detecção de cantos o algoritmo realiza a sobreposição das duas imagens e com uma matriz de correspondências entre as duas imagens, é possível encontrar pontos planos na imagem pois os mesmos são correspondentes, enquanto objetos e obstáculos não ficam correspondentes.

Com a detecção da região plana, um caminho livre é encontrado e informado ao usuário. A Figura 4.8 mostra diferentes ambientes e seus respectivos caminhos livres.

Figura 4.8: Detecção de obstáculos em diferentes cenários (WENQIN; WEI; JIAN, 2011).

4.3.2 *Wearable Real-Time Stereo Vision for the Visually Impaired*

O trabalho de Balakrishnan et al. (2007) é um sistema capaz de determinar a distância utilizando visão stereo para auxiliar deficientes visuais. O nome do sistema criado é *SVETA - Stereo Vision based Electronic Travel Aid*. Este sistema consiste em um capacete com um computador, duas câmeras, e fones de ouvido, como pode ser observado na Figura 4.9:

Figura 4.9: SVETA: Protótipo de auxílio a navegação de deficientes visuais (BALAKRISHNAN et al., 2007).

A duas câmeras ficam posicionadas um pouco acima dos olhos e captam as imagens, repassando-as para o computador embarcado, que por sua vez obtém o mapa de disparidade para verificar distancias e detectar obstáculos para informar por som o que está à frente do usuário.

Para ser calculada a distância, fatores como calibração e posição das câmeras, aquisição das imagens, processamento e comparação das imagens são necessários para obter a distância do objeto.

O sistema é calibrado utilizando a técnica de Zhang (2000), retificado e então são aplicados dois filtros para retirar ruídos das imagens:

- O filtro Laplaciano da gaussiana, onde são suavizados pontos onde ocorrem uma mudança brusca de intensidade, tornado os contornos mais evidentes.

- O filtro de baixa textura, que localiza características uniformemente espalhadas pela imagem, sendo assim, evita que obstáculos ou objetos importantes passem despercebidos.

Aplicando estes filtros, foi possível obter uma imagem de disparidade precisa para detecção de obstáculos onde objetos mais claros estão mais próximos da câmera enquanto os mais escuros tendem a ficar mais longe, como pode ser vista na Figura 4.10:

Figura 4.10: Imagem de disparidade com filtros aplicados (BALAKRISHNAN et al., 2007).

4.3.3 *A navigation and object location device for the blind Impaired*

O trabalho de Caperna et al. (2009) é um sistema de auxílio à navegação de deficiente visuais, com sua navegação utilizando GPS. Além disso, o sistema também realiza o reconhecimento de pontos de interesse utilizando técnicas de visão computacional para a detecção das características dos pontos. O sistema utiliza a técnica SIFT de detecção, onde uma câmera acoplada em um óculos realiza a aquisição da imagem, e informa ao usuário a presença de placas representando pontos de interesse. A Figura 4.11 mostra o usuário em frente a várias placas de ponto de interesse.

Figura 4.11: Usuário em frente de placas de pontos de interesse (CAPERNA et al., 2009).

Como pode ser visto, as placas possuem uma figura significativa e a palavra que representa o ponto de interesse, fazendo com que o reconhecimento tenha mais precisão ao encontrar características e evita encontrar resultados falso-positivos. Ao encontrar uma placa de ponto de interesse, o usuário é informado. A figura 4.12 mostra o reconhecimento de uma placa de um banheiro.

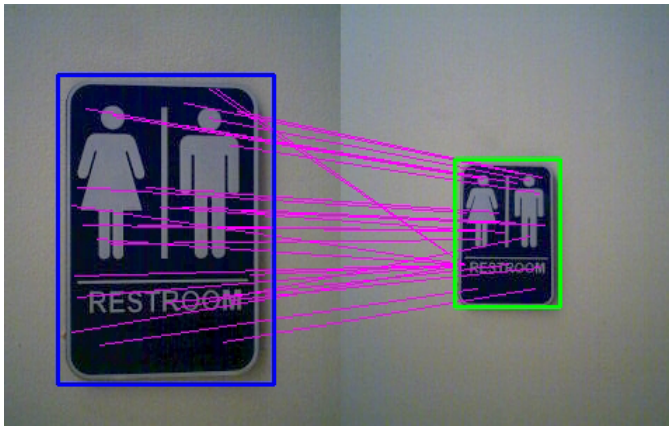

Figura 4.12: Reconhecimento de uma placa (CAPERNA et al., 2009).

4.3.4 *Um Dispositivo Háptico De Auxílio À Navegação Para Deficientes Visuais*

O trabalho de Rodrigues (2006) é baseado em um dispositivo háptico que informa o usuário através de vibrações no pulso do usuário com pulseiras. Segundo Rodrigues (2006), um dispositivo háptico é um dispositivo que utiliza o sentido do tato como forma de interação com o usuário.

O trabalho utiliza um ambiente de simulação para testar a pulseira denominada *V-eye* (de *virtual eye*, ou olho virtual) que é um dispositivo de interface háptico no formato de uma pulseira, que pode ser integrado a diferentes sistemas de auxílio à navegação. A Figura 4.13 mostra o conceito do *V-eye* onde é possível observar a interface integrada com um sistema de navegação.

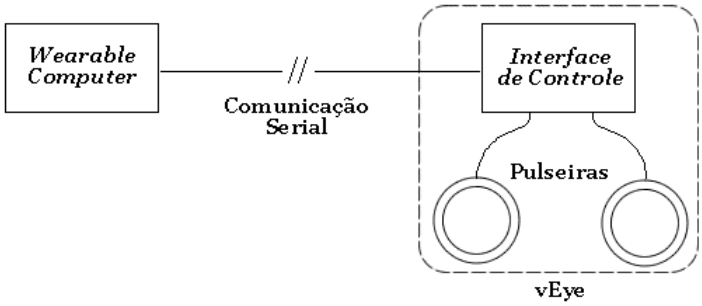

Figura 4.13: Conceito do V-eye (RODRIGUES, 2006).

As pulseiras são conectadas a uma interface de controle, onde recebe os dados vindos de um dispositivo que realiza o processamento de navegação. Os dados recebidos são convertidos em diferentes sinais que a pulseira emite ao usuário, como vibrações mais longas, curtas ou intermitentes, como é possível observar na Figura 4.14 onde são relacionados os diferentes comandos da pulseira.

Instrução	Comportamento do vEye
Ir em frente	Vibração constante, de baixa freqüência, em ambas as pulseiras.
Mover-se lateralmente para a esquerda	Vibração constante, de baixa freqüência, na pulseira esquerda.
Mover-se lateralmente para a direita	Vibração constante, de baixa freqüência, na pulseira direita.
Rotacionar para a esquerda	Vibração intermitente, de alta freqüência, na pulseira esquerda.
Rotacionar para a direita	Vibração intermitente, de alta freqüência, na pulseira direita.
Parar	Vibração constante, de baixa freqüência, em ambas as pulseiras, mas de maneira alternada.

Figura 4.14: Diferentes interações com o V-eye (RODRIGUES, 2006).

4.4 Considerações finais

Este capítulo mostrou diferentes aplicações envolvendo visão computacional ou sistema de auxílio à navegação de deficientes visuais. O próximo capítulo envolve o detalhamento do sistema proposto para este trabalho, como suas técnicas, ferramentas e resultados obtidos.

Capítulo 5

SISTEMA DE AUXÍLIO À NAVEGAÇÃO

5.1 Considerações iniciais

Como os capítulos anteriores abordaram visão estereoscópica e técnicas de processamento de imagens, o desenvolvimento de um sistema que utiliza esses dois conceitos pode ser realizado, uma vez que apresentam características que podem ser aproveitadas para criar um protótipo de auxílio à navegação de deficientes visuais.

Neste capitulo serão detalhados os procedimentos realizados neste trabalho, com o detalhamento dos objetivos, do protótipo a ser implantado e da metodologia de trabalho. Serão descritas também as ferramentas e os métodos auxiliares estudados, assim como os resultados obtidos.

5.2 Objetivos do trabalho

Este projeto de mestrado propõe um protótipo de um sistema de auxílio à navegação e orientação de deficientes visuais em ambientes internos previamente conhecidos e mapeados, utilizando visão computacional através de um dispositivo estereoscópico.

O protótipo será implementado para a validação do projeto, buscando avaliar o tempo de resposta para os algoritmos de detecção de características do ambiente e do algoritmo de navegação.

Os objetivos deste trabalho aqui estabelecidos são:

- Estudar e comparar dois diferentes sistemas estereoscópicos: Duas *webcams* fixadas lado a lado formando um sistema estéreo, e o *Microsoft Kinect*.

- Implementar um algoritmo de reconhecer pontos de interesse no ambiente.
- Implementar um algoritmo de navegação que detecte obstáculos e forneça soluções e orientações para navegação
- Sintetizar e relacionar os algoritmos anteriores para a criação de um protótipo de mapeamento, navegação e orientação de ambiente.
- Validar os métodos proposto neste trabalho.

O processo de mapeamento convencional utiliza equipamentos como lasers e algoritmos de inteligência e visão computacional, como os trabalhos de CORREA (2013) e Silva, Spatti e Flauzino (2010), que utilizam o mapeamento estrutural dos ambientes, onde toda a área interna é mapeada, servindo como locomoção de robôs. Neste trabalho, o mapeamento será feito de forma em que apenas pontos de interesse sejam relevantes para o usuário, pois a navegação será de escolha do usuário, cabendo o sistema auxiliá-lo a desviar de obstáculos.

O protótipo deste projeto consiste em um sistema que simulará um usuário utilizando um sistema de auxílio a navegação, utilizando estímulos presentes da literatura como dispositivos hápticos que estão relacionados ao sentido do tato (RODRIGUES, 2006), dispositivos de som Wenqin, Wei e Jian (2011) ou estímulos elétricos (PEREIRA; JR; JR, 2004). O fluxograma de funcionamento do sistema pode ser visto na Figura 5.1:

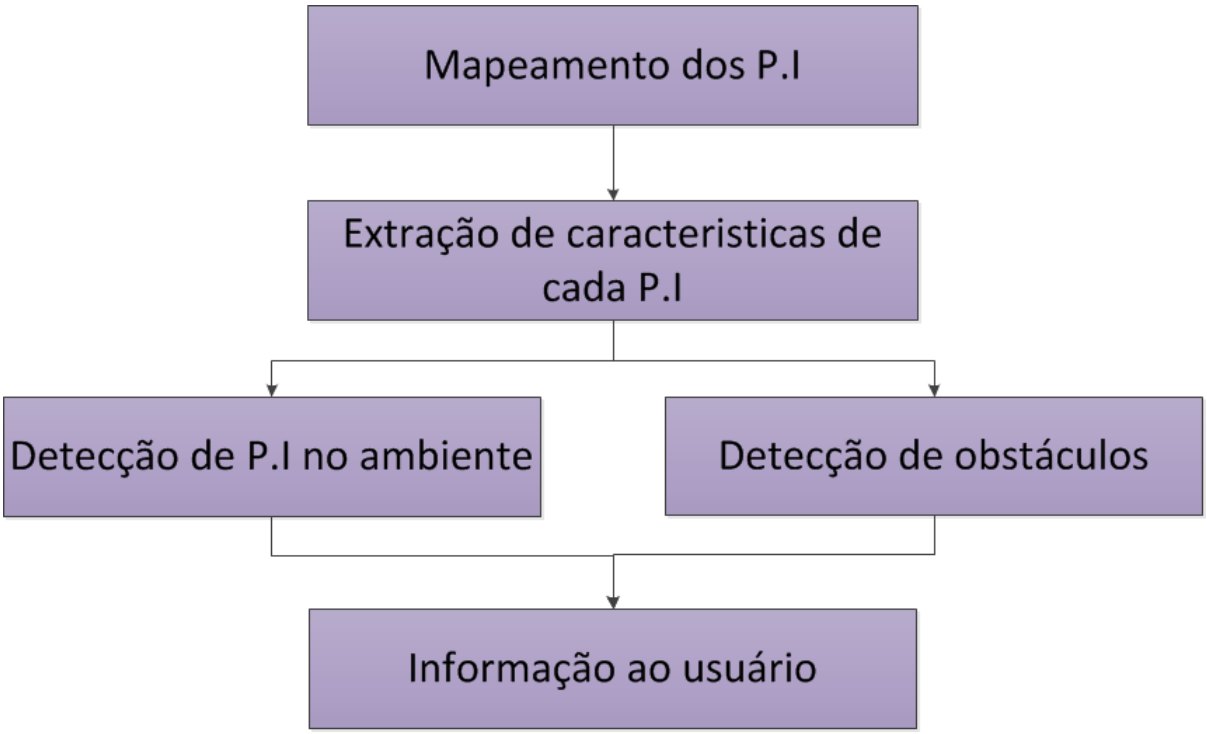

Figura 5.1: Fluxograma do sistema.

O processo deste sistema se iniciará com o mapeamento do ambiente, fixando marcadores em lugares ou objetos que sejam pontos de interesse, como portas, janelas, mesas, entre outros.

O próximo passo consiste em informar ao sistema quais são os símbolos dos pontos de interesse, para que esta possa reconhecer o seu respectivo ponto de interesse.

No passo de navegação e detecção de obstáculos, o usuário se locomoverá para a direção desejada, e quando um obstáculo for encontrado, o usuário será informado do melhor caminho existente.

Quando o usuário se aproximar de um ponto de interesse, o marcador será reconhecido e o sistema informará o ponto de interesse à frente. A Figura 5.2 exibe um cenário com obstáculos e com marcadores de pontos de interesse nas paredes.

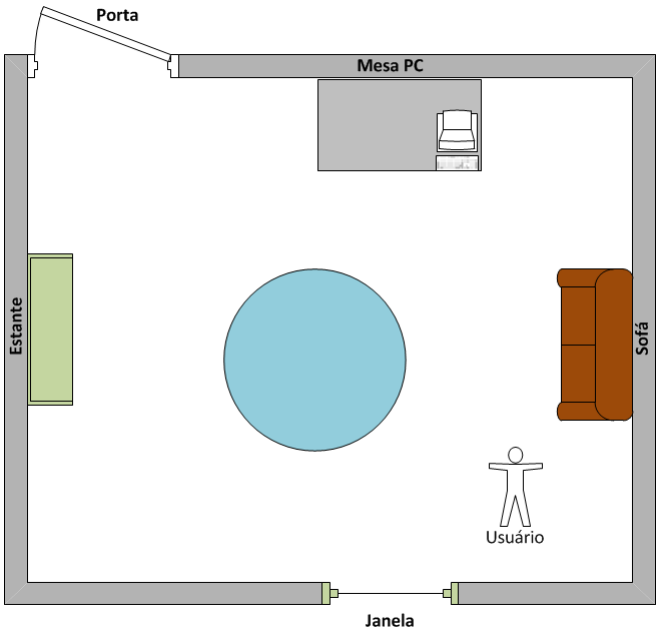

Figura 5.2: Cenário de uma possível utilização do protótipo.

5.3 Ferramentas estudadas

Para este projeto, as principais ferramentas a serem estudadas e posteriormente aplicadas no protótipo são duas:

5.3.1 Kinect

Em 2010, a empresa *Microsoft* lançou o *Kinect*, um sensor de movimentos desenvolvido para o console *Xbox 360*, e posteriormente para computadores, com o lançamento do *Kinect for Windows* e de drivers de compatibilidade com o sistema operacional *Windows*. O *Kinect* possui diversos componentes que funcionam como ferramentas de entrada e saída:

- Câmera RGB com resolução de 640x480 *pixels* com uma frequência de 30Hz;
- Câmera infravermelha para o cálculo de profundidade com 320x240 *pixels* com uma frequência de 30Hz;
- Emissor de infravermelho para o melhor funcionamento do cálculo de profundidade;
- Microfone embutido;
- *Hardware* com processador capaz de calcular profundidade, reconhecimento de voz e reconhecimento do esqueleto humano.

O cálculo da profundidade pela câmera infravermelha é feito a partir do emissor de raios infravermelhos que o *Kinect* possui, que envia raios infravermelhos para o ambiente e a câmera infravermelha capta o sinal refletido e consegue calcular a profundidade dos objetos do ambiente, esta técnica é conhecida como ”luz estruturada”(SILBERMAN; FERGUS, 2011). A imagem gerada pela câmera de profundidade do *Kinect* pode ser observada da Figura 5.3, onde é possível observar que objetos mais escuros estão mais próximos, objetos mais longes ficam com tons de cinza mais claros e objetos em preto não são reconhecidos pelo *Kinect* por estarem ou muito próximos ou muito longe do alcance da câmera.

Figura 5.3: Imagem de profundidade do *Kinect*.

O *Kinect* tem total compatibilidade com a versão do *MATLAB 2013*, e por esta razão, a versão a ser utilizada neste projeto será esta. A Figura 5.4 apresenta o *Kinect* utilizado para este projeto.

Figura 5.4: *Microsoft Kinect* que será utilizado neste projeto.

5.3.2 *Webcams* estereoscópicas

Para este trabalho proposto, além do *Kinect*, foi confeccionado um sistema estéreo, utilizando duas *Webcams* e um suporte de alumínio de modo que as câmeras ficassem fixas paralelamente entre si e na mesma altura, para obter uma maior precisão para calibrar, retificar e encontrar a disparidade das imagens obtidas por elas. A câmera escolhida para este sistema estéreo foi a *webcam Microsoft LifeCam VX-2000* que apresenta uma resolução de 640x480 *pixels*, o suficiente para este trabalho. A Figura 5.5 apresenta o sistema construído.

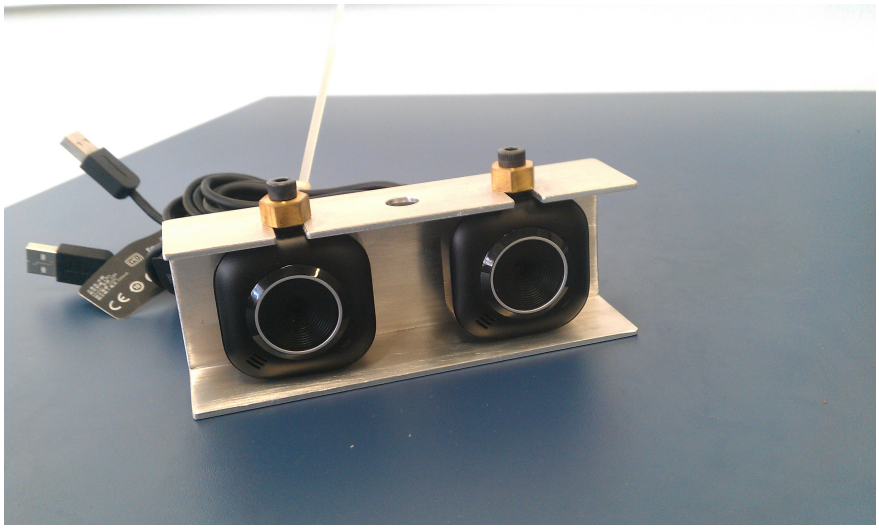

Figura 5.5: Sistema estéreo construído com duas *webcams*.

5.4 Métodos auxiliares estudados

Para este projeto, foram estudados e aplicados diversos métodos que atuaram de forma auxiliar relacionados com o sistema estereoscópio criado para realizar a comparação de dois tipos de câmeras e assim validar e justificar a escolha do *Kinect* como câmera utilizada no protótipo.

5.4.1 *Toolkit* de calibração de sistemas estereoscópicos

O método proposto por Zhang (2000) e implementado por Bouguet (2008), é o mais utilizado para calibrar sistemas estéreos no *MatLab*, e foi escolhido para ser aplicado no sistema estéreo construído para este projeto. O processo para realizar a calibração é dado por:

1. Primeiro, é necessário calibrar as câmeras individualmente, utilizando a extração de cantos, onde o modelo padrão para a calibragem é o modelo de tabuleiro, que possui cantos destacados em escala de cinza; para isso, é necessário estabelecer um número de amostras.

2. Com o número de amostras estabelecido, é necessário obter as amostras, onde várias fotos são obtidas contendo diferentes variações de perspectiva do tabuleiro, podendo este estar inclinado para direita, esquerda, cima, baixo, de perto e de longe, como podem ser observadas na Figura 5.6:

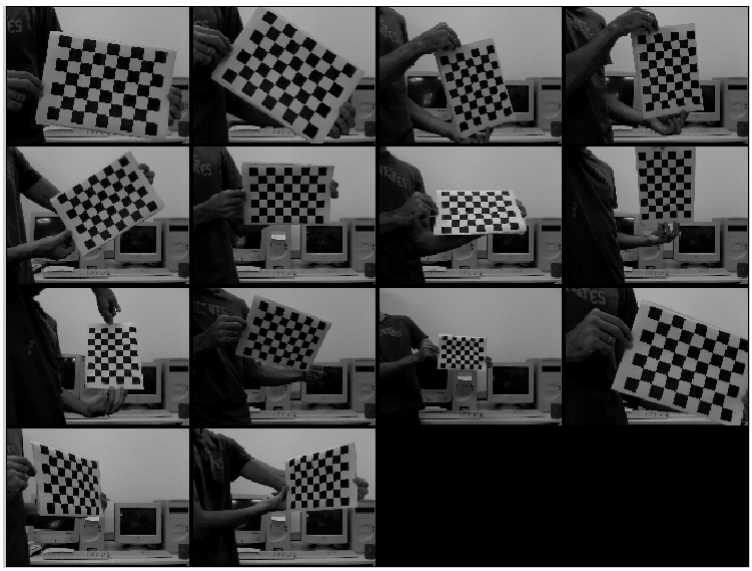

Figura 5.6: Amostras tiradas com o modelo de tabuleiro para extração de cantos.

3. O próximo passo é extrair os cantos de cada amostra. O processo se inicia quando é selecionada a opção *"extract grid corners"* no menu de calibração, onde as imagens de amostras, aparecerão uma de cada vez na tela, e será necessário clicar manualmente em cada canto, formando um perímetro, onde é necessário manter a ordem de indicação de cantos, que normalmente é adotada a indicação no sentido horário. Este processo requer a indicação dos cantos de todas as amostras propostas individualmente. A Figura 5.7 mostra uma amostra com os cantos detectados após a indicação do perímetro

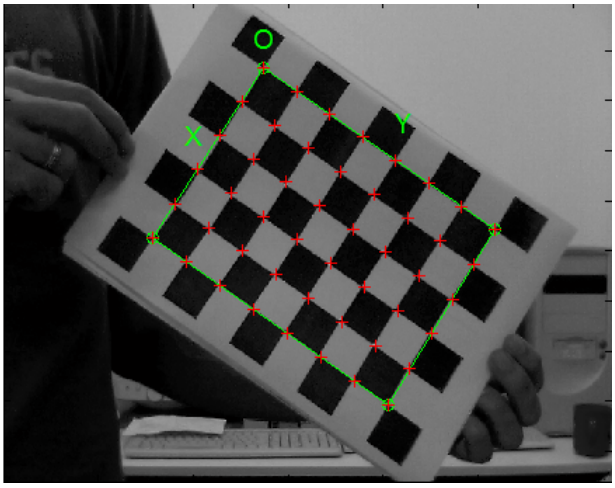

Figura 5.7: Tabuleiro marcado com seus cantos marcados.

4. Com a extração de canto de todas as amostras, basta chamar a função de calibração do *toolkit*, onde é gerado um arquivo contendo parâmetros de calibração como distância

focal, erro, distorção da lente, entre outros.

5. Uma vez feito o processo descrito acima, com as duas câmeras do sistema estéreo, é necessário calibrar as duas câmeras. Para isso, basta abrir a calibração estéreo do *toolkit*, importar os parâmetros de cada câmera, e chamar a função de calibração estéreo, onde é gerado um novo arquivo contendo os parâmetros do sistema estéreo, além de exibir um mapa de profundidade de todas as amostras juntas, como pode ser observado na Figura 5.8.

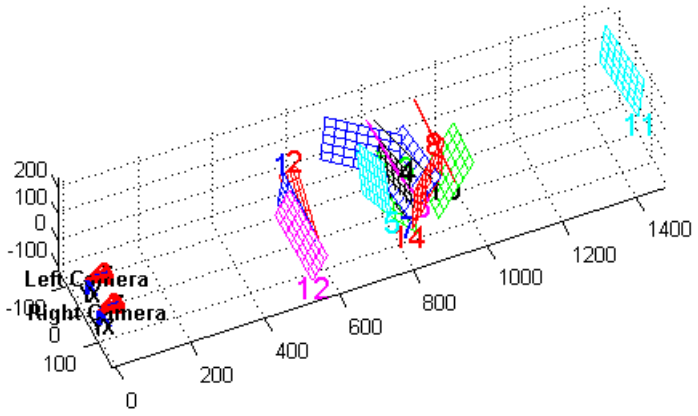

Figura 5.8: Mapeamento do cenário de calibração.

Com o sistema calibrado, é possível realizar a retificação, calcular a triangulação e encontrar a disparidade das cenas captadas pelas câmeras. O próximo método estudado é a retificação.

5.4.2 Retificação de sistemas estereoscópicos

O método de retificação estudado está presente também no *toolkit* de calibração de Bouguet (2008), sendo possível realizar a retificação logo após calibrar as câmeras. Como entrada, o método necessita dos parâmetros obtidos com a calibração e os arquivos das imagens recém calibradas. Para este trabalho, esta função foi modificada para retificar qualquer imagem obtida através do sistema estéreo construído, necessitando apenas dos parâmetros de calibração e do par de imagens a serem retificadas. O processo de retificação é em tempo de processamento com imagens estáticas.

O algoritmo de retificação funcionou de maneira esperada, onde as duas imagens retificadas possuíam linhas epipolares equivalentes. A Figura 5.9 mostra duas imagens retificadas e sobrepostas, onde é possível verificar que ambas estão alinhadas verticalmente, isto é, estão correspondentes na dimensão x do plano cartesiano.

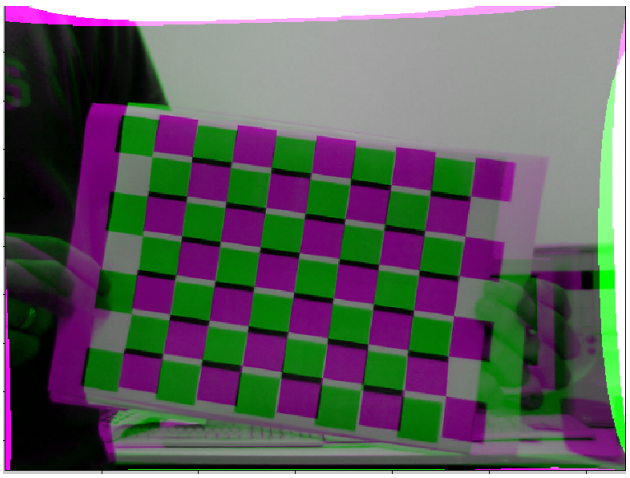

Figura 5.9: Duas imagens retificadas e sobrepostas.

5.4.3 Imagem de disparidade

O método de disparidade está presente na *toolbox* de processamento de imagens do *MatLab*, onde sua função apresenta como parâmetro o par de imagens que se deseja obter a imagem de disparidade; estas imagens devem estar retificadas e ter as mesmas dimensões de tamanho.

Nos testes realizados, a disparidade obtida apresentou pouca precisão, com vários ruídos, como é possível observar na Figura 5.10, além de que o tempo de processamento para a obtenção da imagem de disparidade é superior ao encontrado na obtenção com o *Kinect*, portanto o uso do *Kinect* foi escolhido por apresentar uma imagem com poucos ruídos, melhor definição de borda de objetos e possuir melhor tempo de atualização de *frames*.

Figura 5.10: Imagem de disparidade.

5.4.4 *Toolbox* de integração do *Kinect* com *MatLab*

Para o funcionamento do *Kinect* no *MatLab*, é necessário instalar o *driver Kinect for Windows* e o ambiente de desenvolvimento *Kinect SDK* e por fim baixar a *toolkit* presente nos complementos da ferramenta e possuir uma conta no *MathWorks*, a empresa criadora do *MatLab* e que fornece complementos para o *MatLab*.

5.5 Algoritmo de detecção de obstáculos

O algoritmo utilizado para a detecção de obstáculos, e consequentemente de caminhos livres é baseado no tutorial de robôs de navegação do *RobotRealm* (ROBOTREAM, 2013). O processo conta com seis passos para a obtenção da informação do ambiente, onde é sempre mostrado o caminho com a maior região caminhável livre de obstáculos. As etapas do algoritmo podem ser dadas por: aquisição(1), detecção de bordas(2), aplicação de linhas verticais(3), erosão(4), segmentação(5), dilatação(6) e pôr fim, a interpretação dos dados obtidos(7). Esta subseção será destinada a explicação de cada etapa do algoritmo, utilizando uma imagem de um cenário com obstáculos obtida da câmera de profundidade do *Kinect* como exemplo.

5.5.1 Aquisição de imagem

A etapa da aquisição de imagem é realizada logo no começo do algoritmo, onde um *frame* do vídeo da câmera de profundidade do *Kinect* é obtido automaticamente e como consequência a imagem é convertida para tons de cinza para que a próxima etapa, a detecção de bordas possa ser executada. A Figura 5.11 mostra uma imagem retirada da cena pela câmera de profundidade do *Kinect* onde é possível notar duas cadeiras como obstáculos obstruindo os lados direito e esquerdo do caminho, onde o caminho central é o único possível já que o chão é representado pelo tom de cinza mais claro, e parte em preto não é reconhecida pela câmera do *Kinect* por questões de distância.

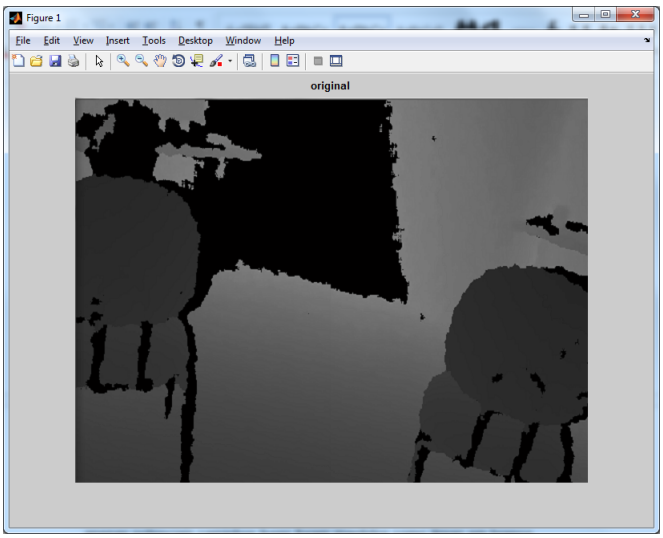

Figura 5.11: Cena adquirida pela câmera de profundidade.

5.5.2 Detecção de bordas

Após a aquisição do cenário e a conversão em uma imagem de escala de cinza, é utilizado o algoritmo de detecção de bordas, e dentre as citadas no capítulo 3 desta dissertação, três técnicas foram testadas: A Figura 5.12 mostra a aplicação da técnica de detecção de Roberts(1), a Figura 5.13 mostra a aplicação da técnica de Sobel(2) e pôr fim a Figura 5.14 mostra o resultado da técnica de Canny(3).

Figura 5.12: Detecção de bordas por Roberts.

Figura 5.13: Detecção de bordas por Sobel.

Figura 5.14: Detecção de bordas por Canny.

Como é possível observar, a técnicas de Roberts e Sobel resultam em detecções de bordas bem parecidas, porém se comparadas com o resultado obtido com a técnica de Canny, é possível observar que o resultado desta técnica garante a detecção mais precisa de bordas em relação as duas primeiras, portanto a escolha da detecção de bordas por Canny foi escolhida para este algoritmo.

5.5.3 Preenchimento vertical

Com a detecção de bordas realizada pelo método de Canny, como resultado foi obtida uma imagem binária contendo as linhas brancas que representam as bordas objetos do cenário. Uma maneira de se interpretar onde é o limiar entre o chão e um objeto é traçar linhas verticais por

todo eixo X da imagem começando de baixo para cima e quando um *pixel* branco é detectado, significa a presença de um objeto, portanto o preenchimento passa para a próxima posição de X, até o fim da imagem. A Figura 5.15 mostra o preenchimento das linhas verticais por toda imagem, e onde é branco, é um possível caminho livre.

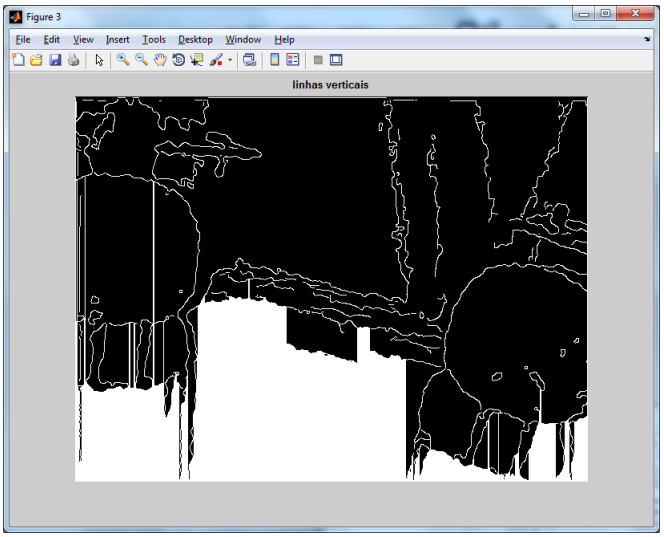

Figura 5.15: Preenchimento de linhas verticais.

Como é possível observar, existem linhas verticais brancas que conseguiram ”atravessar”os objetos através de *pixels* presentes nas bordas que não estão conectados, porém como a informação desejada de potenciais caminhos livres foi obtida, as bordas dos objetos já não têm mais relevância, e portanto a próxima etapa cuidará de isolar os caminhos obtidos, retirando todos os dados irrelevantes ou ruídos da imagem.

5.5.4 Erosão

Esta etapa tem como objetivo isolar todos os possíveis caminhos, retirando da cena dados que não são mais necessários, como as bordas ou linhas que atravessam os objetos, que neste caso são tratados como ruído e são retirados utilizando a técnica de erosão, onde componentes com área pequena como linhas tendem a ser removidos ou diminuídos, como é possível observar na Figura 5.16, onde todas as linhas de bordas e outros dados irrelevantes foram removidos e apenas potenciais caminhos livres foram mantidos como áreas em branco.

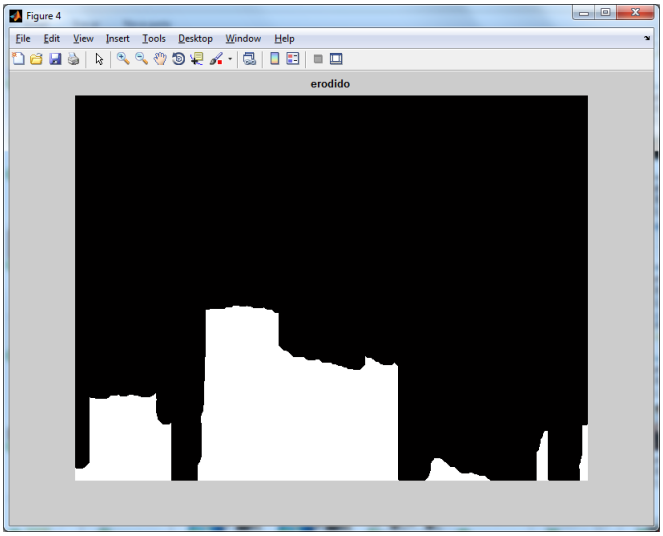

Figura 5.16: Imagem erodida com ruídos retirados.

5.5.5 Segmentação

Utilizando o método de Otsu (OTSU, 1975), é possível isolar componentes de uma imagem que possuem uma área de um determinado tamanho a ser definido, e com a erosão aplicada, componentes mais significativos em área são mantidos na imagem, enquanto que os menores tendem a sumir. A Figura 5.17 mostra a cena com apenas o maior componente sendo exibido na imagem.

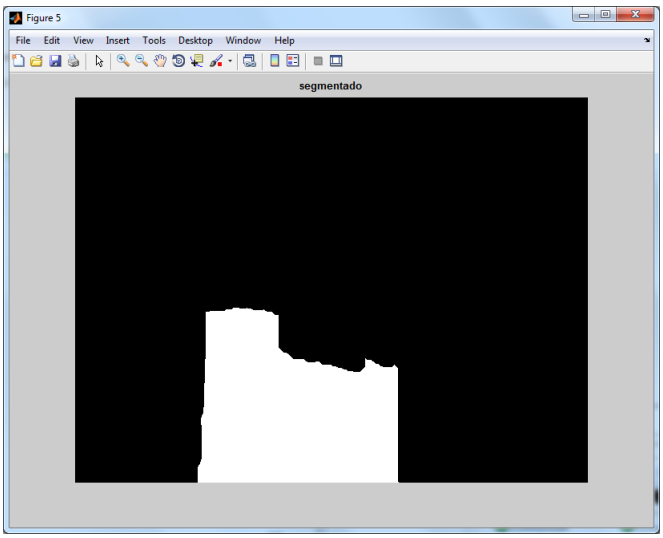

Figura 5.17: Maior componente isolado através da segmentação.

5.5.6 Dilatação

Com o componente isolado, é necessário realizar a operação de dilatação para compensar a erosão anteriormente aplicada, assim restaurando o tamanho original do componente e consequentemente aumentando a precisão do tamanho do caminho livre; a Figura 5.18 mostra o aumento da área do componente isolado depois do processo de dilatação:

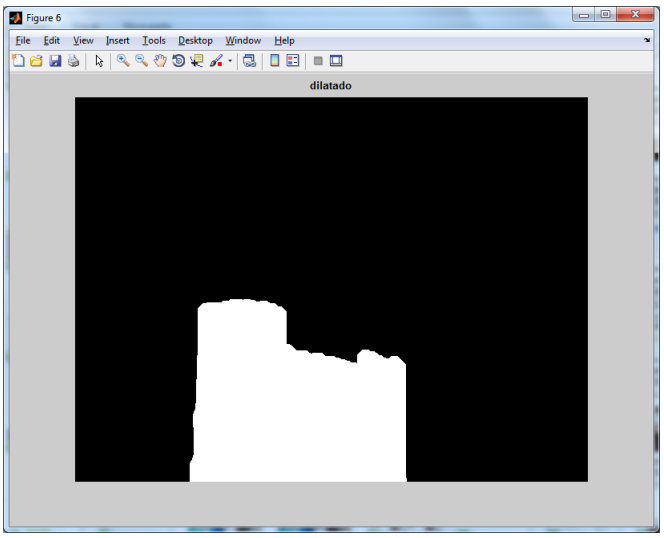

Figura 5.18: Componente dilatado.

5.5.7 Interpretação e saída

Com a indicação de um componente que tem potencial de ser um caminho livre, a imagem é então dividida em três partes iguais, onde são denominadas as orientações esquerda, centro e direita, sendo que cada orientação possuí uma resolução de $213x160$, isto é, $1/3$ da imagem original que possui a resolução de $640x480$. Para determinar qual das três orientações tem o melhor caminho livre de obstáculos, é feita a checagem de números de *pixels* brancos em cada orientação, onde para ser um caminho livre, a orientação deve seguir as seguintes condições:

- A orientação com caminho livre deve possuir mais *pixels* brancos que as outras duas orientações;
- Caso exista igualdade na soma de pixels brancos, a orientação que possuir o pixel mais alto terá prioridade;
- Caso o número de *pixels* seja inferior a um limiar de 30% do total de *pixels* da orientação, o caminho não será considerado livre;

- Caso nenhuma orientação possua os critérios acima, nenhum caminho livre será detectado e o usuário é informado que não foi encontrado nenhum caminho livre a frente;
- Caso uma orientação atenda os critérios acima, o usuário é informado da existência de obstáculos e qual é a melhor orientação para se locomover.

O usuário será informado sobre a existência de obstáculos através de áudios com fala previamente gravados, como por exemplo se a orientação do centro possuir um caminho livre a interface informara o usuário com a seguinte frase: ”Caminho livre à frente”. Para evitar problemas de repetições de uma frase ou de áudios sobrepostos, será respeitado um tempo de três segundos até uma nova informação. A Figura 5.19 mostra o caminho livre sendo destacado, onde neste caso é a orientação do centro o melhor caminho.

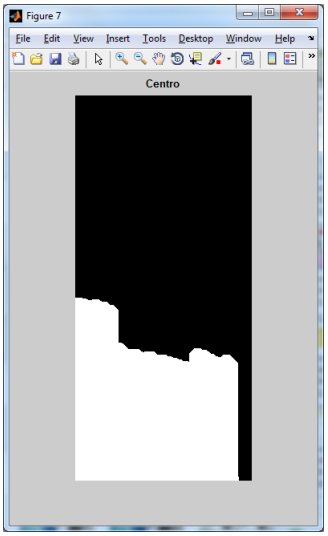

Figura 5.19: O melhor caminho é mostrado.

Após a detecção do melhor caminho, o algoritmo aguarda o tempo de espera para o comando de voz ser dito e volta ao começo da etapa 1 para realizar a aquisição de uma nova imagem, seguindo sempre no fluxo apresentado.

5.6 Algoritmo de detecção de pontos de interesse

Assim como no trabalho de extração de características de Caperna et al. (2009), o algoritmo utilizado para a detecção de pontos de interesse tem como objetivo buscar em uma cena, possíveis pontos de interesses pré-cadastrados e informar ao usuário quando o mesmo passar

em frente deste ponto. O algoritmo possui cinco etapas e são elas: Aquisição do ponto de interesse(1), aquisição da cena(2), detecção de características(3), comparação de características(4) e pôr fim a conversão dos dados obtidos em informação ao usuário(5). Todo o processo é realizado através da câmera RGB do *Kinect*, funcionado em conjunto com a câmera de profundidade e consequentemente com o algoritmo de detecção de obstáculos.

5.6.1 Aquisição do ponto de interesse

A etapa da aquisição do ponto de interesse ocorre logo no começo da iteração, onde é informada qual imagem será carregada para a etapa de comparação com a imagem da cena obtida. A imagem deve conter um símbolo representativo do objeto do ponto de interesse de forma limpa e de modo que o objeto esteja contido inteiro na imagem, como por exemplo a Figura 5.20, onde é mostrada a representação de um ponto de interesse relacionado a um banheiro.

Figura 5.20: Exemplo de um ponto de interesse para um banheiro.

Como é possível observar, a imagem possui traços fortes de características, como muitas bordas, e símbolos, o que facilita no reconhecimento de características fortes pelo método de obtenção de características. Porém, como a câmera estará apontada para o chão, considera-se uma boa imagem de ponto de interesse aquela em que o objeto já está fixo no cenário, como mostra a Figura 5.21, onde é possível observar o ponto de interesse já fixado em seu local, facilitando a busca do ponto no cenário pois sua escalabilidade, rotação e angulação estarão iguais a imagem captada pelo cenário.

Figura 5.21: Ponto de interesse para um banheiro fixo no cenário.

5.6.2 Aquisição da cena

Com os pontos de interesses devidamente informados, uma iteração repetitiva ocorrerá onde será adquirida a imagem do cenário filmada através da câmera RGB do *Kinect* conforme o usuário se locomove. A Figura 5.22 mostra a aquisição de um cenário navegado pelo usuário.

Figura 5.22: Cena contendo um ponto de interesse.

5.6.3 Detecção de características

Utilizando o algoritmo *SURF*, é aplicada a detecção automática de características fortes na imagem do ponto de interesse e da imagem da cena, onde são guardados em um vetor, os pontos mais fortes de cada imagem, para posteriormente serem comparados. A escolha do algoritmo *SURF* em relação ao *FAST* deu-se pelo motivo de que o primeiro embora mais lento, extrai e compara pontos de característica com mais precisão e em relação ao *SIFT*, ambos possuem uma detecção robusta, porém o *SURF* se mostrou com respostas mais rápidas entre cada iteração, por esse motivo, a técnica *SURF* foi escolhida para esta etapa do algoritmo. Como explicado no capítulo 3 deste trabalho, a detecção de características é o resultado de mudanças bruscas na imagem como cantos ou mudanças de cor. As Figuras 5.23 e 5.24 mostram respectivamente os pontos de característica fortes da imagem do ponto de interesse e da cena.

Figura 5.23: Pontos de características do símbolo.

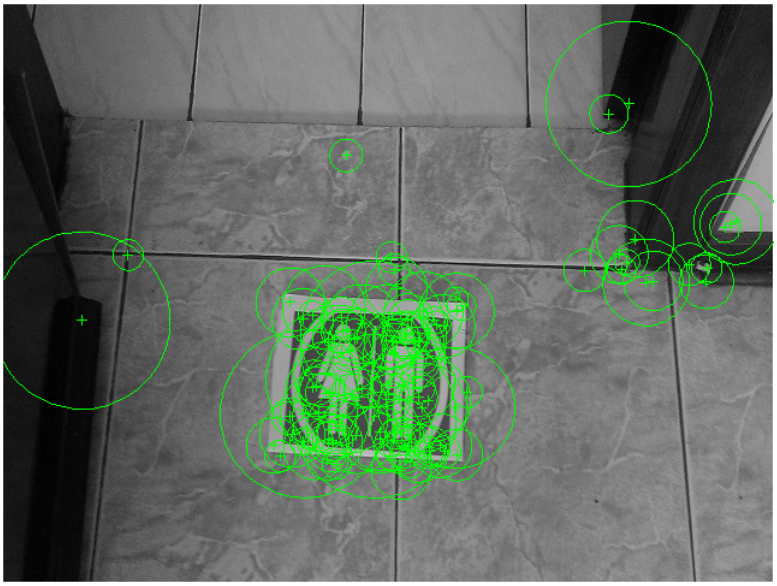

Figura 5.24: Pontos de características da cena.

É possível observar que na detecção de pontos de característica da imagem do ponto de interesse, a maioria dos pontos fortes são aqueles presentes em mudanças de tons, quinas e bordas do símbolo. Na detecção dos pontos da cena, a maioria dos pontos encontrados ficou concentrado também no símbolo, comprovando que quanto mais detalhes e mudanças de tons o símbolo possuir, mas fácil será a detecção de pontos de fortes na cena. Após a extração dos pontos de característica de cada imagem, o próximo passo será a comparação.

5.6.4 Comparação de características

Esta etapa tem como objetivo a comparação dos pontos de características da imagem do símbolo e da cena, obtidas na etapa anterior, afim de determinar a possibilidade da cena possuir o símbolo de um ponto de interesse. A comparação é feita utilizando os dois vetores obtidos na etapa anterior, comparando os pontos das coordenadas. Como resultado, é gerado um novo vetor contendo os pares de pontos correspondentes nas duas imagens, a Figura 5.25 mostra a detecção de pontos correspondentes entre a imagem do ponto de interesse e a imagem da cena.

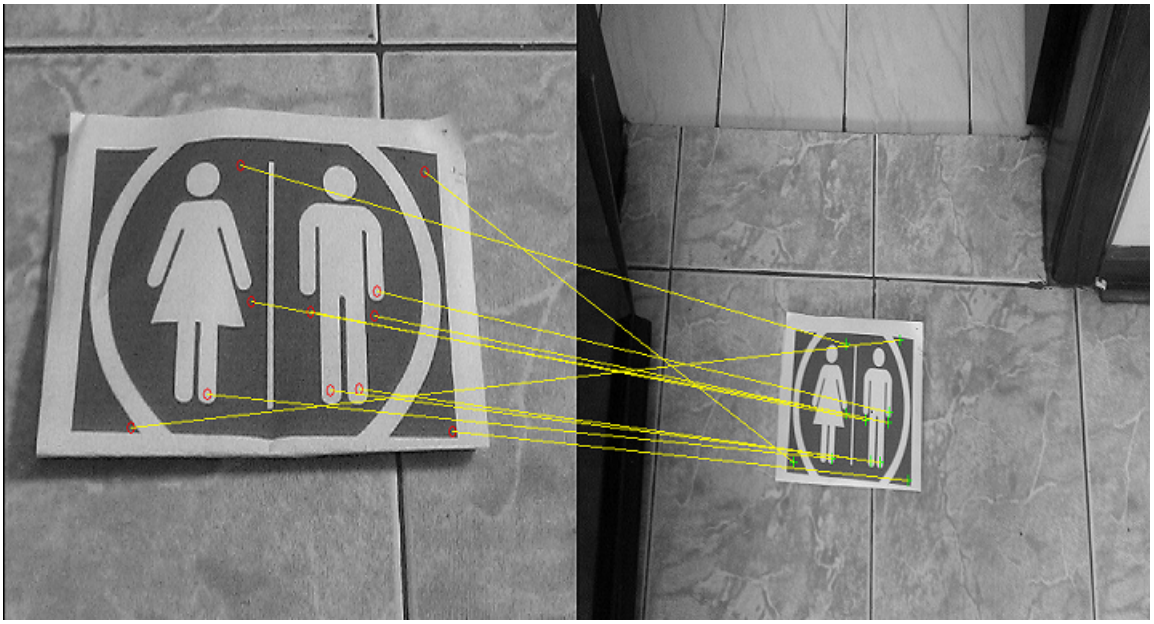

Figura 5.25: Pontos correspondentes entre as duas imagens.

É possível observar que alguns pontos que foram considerados como correspondentes estão apontando para pontos distintos, porém todos dentro do símbolo. Essa falta de precisão não afeta o desempenho, pois mesmo alguns apontando para pixels errados, ainda assim ficam concentrados dentro do símbolo.

5.6.5 Interpretação e saída

Com as etapas anteriores completadas, é necessária a interpretação dos dados obtidos. Para que um ponto de interesse seja confirmado em uma cena, é necessário que o mesmo possua no mínimo um número N de pontos correspondentes, onde após testes, o número de 20 pontos deve ser o mínimo existente, onde N é a soma dos pontos encontrados na imagem do ponto de interesse com a imagem da cena, portanto, cada imagem deve possuir 10 pontos encontrados.

O usuário será informado sobre a existência de pontos de interesse através de áudios com fala previamente gravados, como por exemplo se a foi encontrado o símbolo do banheiro, o áudio dirá ”Banheiro a frente”. Para evitar problemas de repetições de uma frase ou de áudios sobrepostos, será respeitado um tempo de três segundos até uma nova informação, assim como acontece com o algoritmo de detecção de obstáculos. A Figura 5.26 mostra a detecção de um ponto de interesse em um cenário.

Figura 5.26: Ponto de interesse detectado em uma cena.

5.6.6 Resultados obtidos

Como foi explicado na metodologia deste trabalho, a escolha do algoritmo SURF foi em consideração aos resultados obtidos na aplicação e comparação dos algoritmos SIFT, SURF e FAST, utilizando cinco diferentes símbolos de pontos de interesse, como podem ser vistos na Figura 5.27, onde são mostrados os símbolos utilizando no experimento.

Figura 5.27: Símbolos utilizados para comparação das técnicas de identificação de pontos de interesse.

A Tabela 5.1 mostra a média de número de pontos de características encontrados em 50 amostras de diferentes perspectivas, distâncias e angulação de cinco diferentes símbolos de ponto de interesse:

Tabela 5.1: Média do número de pontos de características.

Amostra	SIFT	SURF	FAST
1	45	44	38
2	119	99	81
3	227	227	191
4	164	167	118
5	145	144	85

É possível observar que o algoritmo SIFT obteve, em média mais pontos de características, o SURF ficou em segundo lugar com pouca diferença e o FAST ficou em último. O trabalho de (guerrero) mostra a comparação de aquisição de pontos de características, como pode ser observado no gráfico da Figura 5.28.

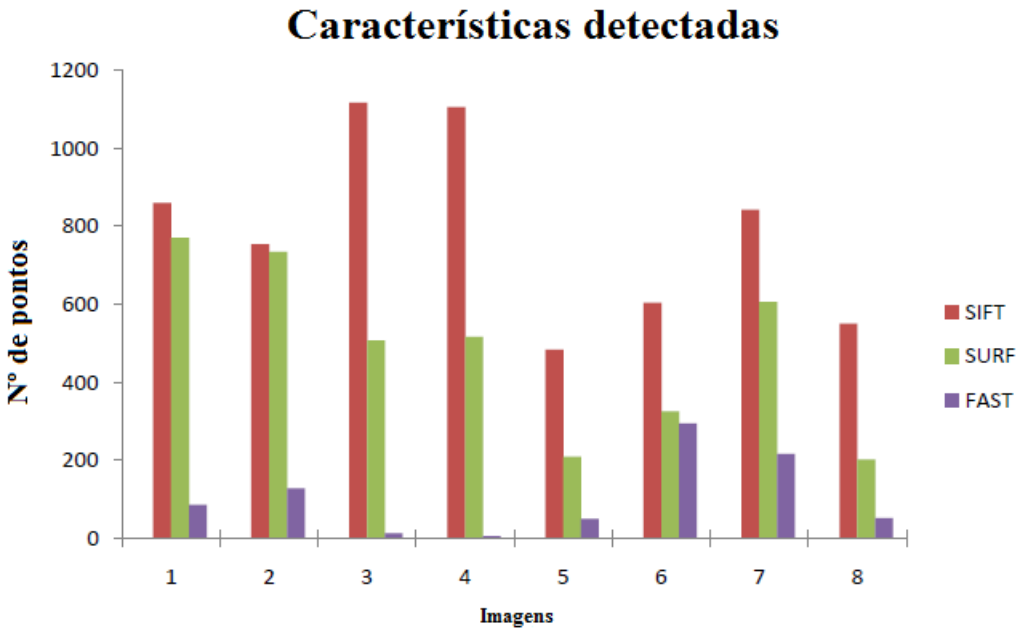

Figura 5.28: Gráfico de comparação entre os algoritmos do trabalho de Guerrero (2011).

Este gráfico demonstra que a média de detecção de características do algoritmo SIFT é maior do que o SURF e a detecção do SURF é maior que a do FAST.

Para comparações, os resultados em formato de gráfico da média de 70 amostras dos cinco marcadores testados neste experimento podem ser observados na Figura 5.29, onde é possível observar a semelhança dos resultados respeitando as proporções.

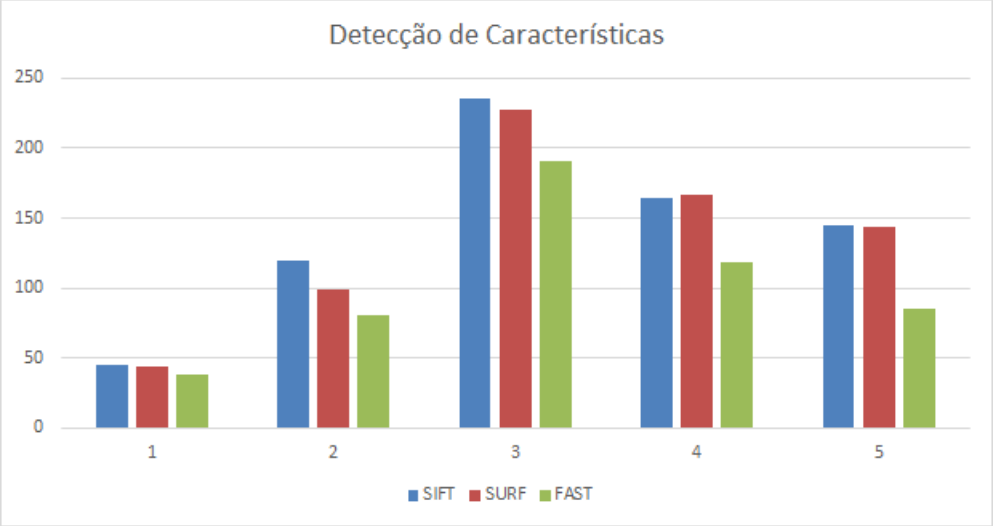

Figura 5.29: Gráfico de comparação entre os algoritmos deste trabalho.

Para pontos de características correspondentes, a tabela 5.2 mostra a média obtida de 50 amostras de cada um dos cinco marcadores na comparação entre o símbolo de ponto de interesse e no cenário.

Tabela 5.2: Média do número de pontos correspondentes de características.

Amostra	SIFT	SURF	FAST
1	18	18	4
2	48	46	8
3	92	100	6
4	65	64	12
5	41	44	6

O número de pontos correspondentes dos algoritmos SIFT e SURF são próximos entre si, já os pontos correspondentes do algoritmo FAST ficaram em um número reduzido em relação aos outros dois.

Por fim, a Tabela 5.3 mostra a média do tempo de execução em segundos de cada algoritmo com as 50 amostras dos cinco marcadores, desde a captura da imagem, até a detecção do ponto de interesse.

Com os dados mostrados nas tabelas exibidas, o algoritmo que se mostrou mais adequado para esta aplicação foi o SURF, pelo fato de conseguir identificar um número superior de pontos de características e pontos correspondentes em relação ao FAST, e ter um tempo de execução menor em relação ao SIFT.

Com a junção dos algoritmos de detecção de obstáculos e de detecção de pontos de inte-

Tabela 5.3: Média do tempo de execução em segundos de cada algoritmo.

Amostra	SIFT	SURF	FAST
1	0.34	0.17	0.1
2	0.76	0.3	0.13
3	0.79	0.64	0.14
4	0.61	0.46	0.12
5	0.38	0.2	0.12

resse, é possível sintetizar e testar o protótipo do sistema de auxílio à navegação em ambientes internos. Como o hardware do *Kinect* não possui características para mobilidade, o teste foi feito em ambientes controlados, porém em trabalhos futuros espera-se obter um hardware com mobilidade utilizando as técnicas propostas nesse trabalho.

Antes de aplicar o modo de comunicação do protótipo com o usuário, foi feito um questionário presencial (presente no "Anexo A"desta dissertação) com cinco deficientes visuais, de faixas etárias diferentes, contendo pessoas com idade de 10 a 63 anos com perguntas relacionadas ao trabalho proposto e sobre preferencias em relação à interface de comunicação onde os dados podem ser analisados a seguir:

- 100% acreditam que o sistema terá utilidade no dia-dia;
- 20% já utilizaram alguma vez um sistema de auxílio eletrônicos: Um usuário experimentou uma bengala que emite som ao se aproximar de obstáculos e outro usuário utiliza um GPS acoplado em um celular adaptado para localização;
- 80% se importam com a aparência do dispositivo;
- 60% preferem que as informações sejam transmitidas por voz, 20% preferem que as informações sejam informadas por pulseiras vibrantes e 20% com *Bip* seguido de voz.
- 60% dos usuários preferem utilizar apenas um fone de ouvido, 20% utilizar fone nos dois ouvidos e 20% preferem não utilizar fones de ouvidos;
- 100% acreditam que a funcionalidade que deve ter maior prioridade é a de detecção de pontos de interesse.

Com os dados obtidos, o modo de comunicação escolhido foi áudio por voz e como os algoritmos possuem certos aspectos conflitantes, como por exemplo, a prioridade de detecção de obstáculo ou detecção de pontos de interesse, foi escolhido como primeira prioridade a detecção de pontos de interesse pois este possui mais uma ocorrência menor de acontecer, e quando o usuário passa por um ponto de interesse, essa detecção não pode ser deixada em segundo plano.

Como resultado, os testes em diferentes ambientes internos com diferentes pontos de interesse e obstáculos foram testados:

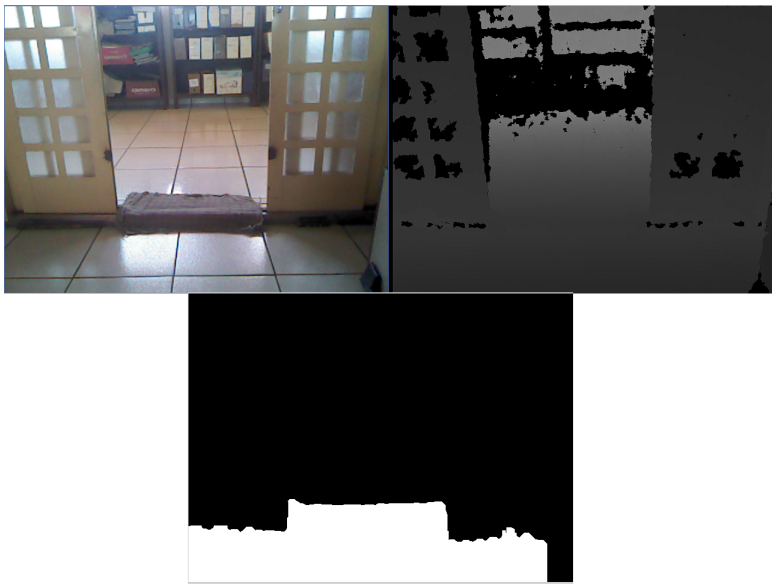

Figura 5.30: Cenário e suas respectivas imagens de profundidade e indicação de caminho livre pela orientação pelo centro.

Figura 5.31: Cenário e suas respectivas imagens de profundidade e indicação de caminho livre pela orientação pela esquerda.

Figura 5.32: Reconhecimento do ponto de interesse de escada.

Figura 5.33: Reconhecimento do ponto de interesse de saída de emergência.

Como pode ser observado nas Figuras 5.30 e 5.31, diferentes posições em relação a uma porta foram testados, e o protótipo respondeu bem aos diferentes tipos de posições de caminhos livres, enquanto que nas Figuras 5.32 e 5.33 em que são mostradas as detecções de dois pontos de interesse. Foram testados vários cenários onde a tabela 5.4 mostra os dez locais escolhidos para realizar os testes de circuitos, levando em conta a incidência de luz externa presente no ambiente. Dos dez locais, foram escolhidos dois locais externos para a verificação da interferência da luz do sol no reconhecimento de pontos de interesse e obstáculos.

Tabela 5.4: Locais de teste e suas respectivas incidências de luz externa.

Ambiente	Local	Iluminação externa
1	Corredor do departamento	Média
2	Sala de aula cortinas fechadas	Baixa
3	Sala de aula cortinas abertas	Média
4	Escritório	Média
5	Laboratório	Baixa
6	Casa	Média
7	Biblioteca	Média
8	Área Externa da Faculdade	Alta
9	Quintal	Alta
10	Auditório	Baixa

Como pode ser observado, foram escolhidos ambientes diferentes como um escritório, biblioteca, casa e seu quintal, onde os circuitos foram testados com obstáculos e pontos de interesse em locais aleatórios destes ambientes. O teste foi realizado com uma pessoa vendada e outra ao lado acompanhando os resultados obtidos no computador e verificando a segurança da trajetória da pessoa vendada. Cada circuito foi percorrido três vezes, respeitando a mesma configuração de obstáculos e pontos de interesse e foi realizado uma média de instruções fornecidas pelo sistema, assim como a média de instruções de navegações corretas e a média de instruções de pontos de interesse corretas. A tabela 5.5 mostra o resultado dos testes realizados nos ambientes.

Tabela 5.5: Resultados obtidos nos teste de circuito.

Ambiente	Média de Instruções	Navegação	Acertos	P.I	Acertos
1	11	10	9	1	1
2	22	20	20	2	2
3	22	20	18	2	2
4	32	28	24	4	3
5	18	17	16	1	1
6	26	23	20	3	2
7	23	20	18	3	3
8	19	16	11	3	2
9	17	16	11	1	1
10	21	19	18	2	2

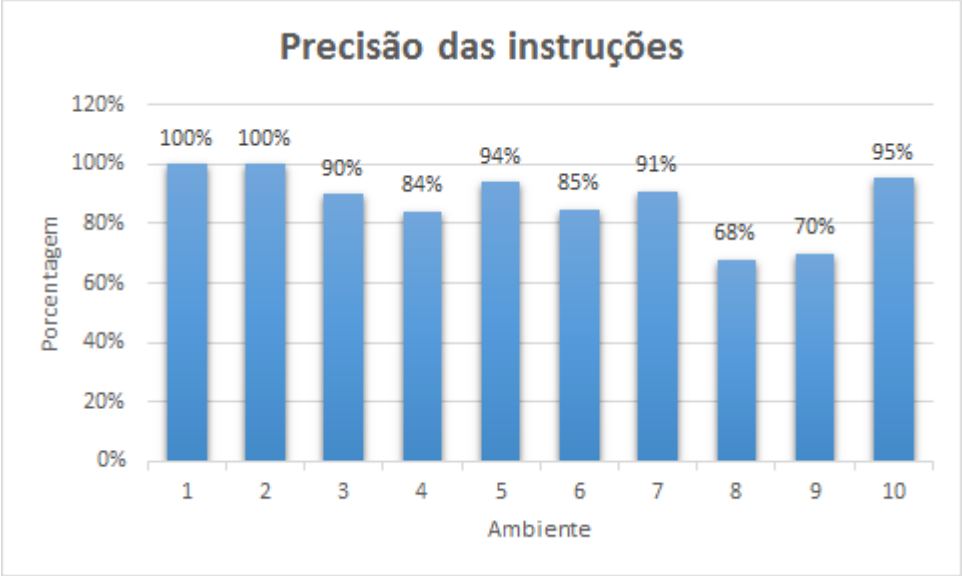

Figura 5.34: Gráfico de precisão das instruções de cada ambiente.

Como pode ser observado, o número de instruções corretas para navegação e pontos de interesse de cada ambiente foi variado, assim como seus respectivos acertos. É possível observar no gráfico da Figura 5.34 que em ambientes com média e baixa incidência de iluminação externa, o número de acertos pode ser considerado como bom, entretanto em ambientes com alta incidência de luz do sol, isto é, feixes e reflexos de luz inseridos no chão, o número de acertos foi considerado mediano. Este tipo de problema está relacionado com a câmera infravermelha do *Kinect*, em que em determinadas intensidades de luz do sol, pode existir uma interferência que leva o sistema a interpretar um feixe de luz ou reflexo como um obstáculo a ser detectado. Os erros de precisão de instrução foram considerados quando uma instrução é omitida, atrasada ou errada. Dos dez ambientes, foram filmados três testes para a demonstração do funcionamento do protótipo deste projeto, e seus respectivos *links* podem ser encontrados no ”Anexo C”desta dissertação.

5.7 Considerações finais

Este capítulo foi dedicado ao detalhamento do trabalho proposto, onde foram descritas as ferramentas, os métodos estudados e aplicados, e por fim foram exibidos os resultados através de um questionário e de aplicações do sistema em diferentes ambientes. O próximo capítulo é destinado as conclusões obtidas nesse trabalho, assim como as possibilidades para trabalhos futuros.

Capítulo 6

CONCLUSÕES

6.1 Considerações iniciais

No capítulo anterior foi mostrado o estudo de técnicas e ferramentas e desenvolvimento do protótipo proposto neste trabalho, assim como os resultados obtidos. Este capítulo traz as conclusões obtidas no desenvolvimento e teste do trabalho apresentado.

6.1.1 Conclusões obtidas

Este trabalho utilizou apenas sensores visuais, através de técnicas e ferramentas relacionadas à visão computacional e a conclusão obtida foi que mesmo com informações muito relevantes, um sistema de auxílio a navegação necessita de mais sensores para obter resultados mais precisos. Um exemplo é o fato de o sistema implementado neste trabalho não possuir detecção de aproximação de outras orientações como atrás do usuário, e em seus lados e levando em conta que o sistema utiliza a informação por voz, a aplicação de sensores de proximidade poderiam auxiliar ainda mais o usuário a interpretar o que acontece ao seu redor, em diferentes orientações.

O sistema também necessitou de um computador robusto para realizar a exibição em tempo real dos vídeos das duas câmeras do Kinect. Para a realização dos teste foi utilizando um computador com proecessador *Intel i7 2ghz*, 8gb de memória, placa de vídeo dedicada de 2gb. Para trabalhos futuros, os algoritmos serão melhorados para que seja necessário computadores menos robustos e mais portáteis.

O sistema apresenta uma boa resposta, porém em ambientes com baixa iluminação ou ausência de luz, o sistema se torna ineficaz por depender apenas da visão computacional. Além dos sensores de navegação já citados, possíveis soluções para este problema podem ser relaci-

onadas à utilização de etiquetas de identificação por proximidade, ou a utilização de câmeras com infravermelho que conseguem captar as informações visuais mesmo em ambientes com baixa ou ausência de luz.

Embora os problemas acima citados ocorram, este trabalho foi bem aceito dentre as pessoas deficientes visuais no qual o trabalho foi descrito e demonstrado presencialmente, tornando a ideia deste trabalho válida, possibilitando a continuidade em trabalhos futuros para o aprimoramento do protótipo.

Como possíveis trabalhos futuros podem ser citados:

- Aprimoramento nas técnicas desenvolvidas neste trabalho;
- Utilização de novos sensores para aumentar a confiabilidade dos dados;
- Implementação de um hardware dedicado para maximizar os recursos, assim como obter resultados com melhor desempenho;
- Avaliação de portabilidade do sistema para *smartphones* utilizando a própria câmera do dispositivo;
- Criação de um protótipo portátil e utilizável para deficientes visuais;
- Atuação do protótipo em ambientes internos e externos;
- Teste e avaliação do protótipo com deficientes visuais;

6.2 Considerações finais

Este capítulo mostrou as conclusões obtidas neste trabalho, mostrando a viabilidade com alguns pontos positivos e outros que devem ser aprimorados, deixando a possibilidade para possíveis trabalhos futuros para continuação em um trabalho de doutorado.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, M. *Development of a stereo vision system for outdoor mobile Robots*. Tese (Doutorado) — University of Florida, 2006.

AIRES, G. *Estimação de movimento de uma câmera via geometria epipolar*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro – UFRJ, 2010.

BALAKRISHNAN, G. et al. Wearable real-time stereo vision for the visually impaired. *Engineering Letters*, v. 14, n. 2, p. 6–14, 2007.

BAY, H.; TUYTELAARS, T.; G, L. V. Surf: Speeded up robust features. In: *Computer Vision–ECCV 2006*. [S.l.]: Springer, 2006. p. 404–417.

BORTH, M. R. et al. Análise da extração de atributos do algoritmo surf em espécies de peixe. 2013.

BOUGUET, J.-Y. Camera calibration toolbox for matlab (2008). *URL http://www. vision. caltech. edu/bouguetj/calib_doc*, 2008.

CANNY, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, n. 6, p. 679–698, 1986.

CAPERNA, S. et al. A navigation and object location device for the blind. 2009.

COELHO, M. C. F. S. P.; TAVARES, J. M. R. da S. Toolbox de calibração de câmaras para matlab. 2003.

CORREA, D. S. O. *Navegação autônoma de robôs móveis e detecção de intrusos em ambientes internos utilizando sensores 2D e 3D.* Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, ¡http://www.teses.usp.br/teses/disponiveis/55/55134/tde-26082013-100127/¿. Acesso em: 2013-08-07, 2013.

FACON, J. *Limiarização Bimodal de Otsu*. [S.l.]: PUCPR, 2004.

FERNANDES, H. et al. Stereo vision in blind navigation assistance. In: IEEE. *World Automation Congress (WAC), 2010*. [S.l.], 2010. p. 1–6.

FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999.

FUSIELLO, A.; TRUCCO, E.; VERRI, A. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, Springer, v. 12, n. 1, p. 16–22, 2000.

GARDIMAN. Implementação de um sistema de visão estéreo e triangulação como técnica para determinação de distância. 2008.

GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Edgard Blucher, 2000.

GUERRERO, M. A comparative study of three image matcing algorithms: Sift, surf, and fast. 2011.

HAMZAH, R. A.; SALIM, S.; ROSLY, H. N. An effective distance detection of obstacles in stereo vision application. *Canadian Journal on Electrical and Electronics Engineering*, v. 1, n. 3, p. 49–53, 2010.

HARTLEY, R. I. Theory and practice of projective rectification. *International Journal of Computer Vision*, Springer, v. 35, n. 2, p. 115–127, 1999.

KARLSTROEM, A. *Estimaçao de posiçao e quantificaçao de erro utilizando geometria epipolar entre imagens*. Tese (Doutorado), 2007.

KAUFMAN, M. *Mars Landing 2012: Inside the NASA Curiosity Mission*. [S.l.]: National Geographic Books, 2012.

KHOSHELHAM, K.; ELBERINK, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, Molecular Diversity Preservation International, v. 12, n. 2, p. 1437–1454, 2012.

KIRNER, C.; TORI, R. Realidade virtual. *Apostila de graduação. Faculdade de informática. Fundação Eurípides de Marília*, 2004.

KUMAR, S. et al. Stereo rectification of uncalibrated and heterogeneous images. *Pattern Recognition Letters*, Elsevier, v. 31, n. 11, p. 1445–1452, 2010.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, Springer, v. 60, n. 2, p. 91–110, 2004.

MARQUES, F.; LUIS, C. Geração automática de mapas de disparidade em visão estéreo. 1998.

NAM, B. et al. Pedestrian detection system based on stereo vision for mobile robot. In: IEEE. *Frontiers of Computer Vision (FCV), 2011 17th Korea-Japan Joint Workshop on*. [S.l.], 2011. p. 1–7.

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, v. 11, n. 285-296, p. 23–27, 1975.

PEDRINO, E. C. *Apostila de Tratamento Digital de Imagens)*. [S.l.]: Ufscar, 2003.

PEREIRA, M. C.; JR, A. C.; JR, F. K. Substituição sensorial para auxílio à mobilidade de deficientes visuais via eletroestimulação tátil. In: *III Congresso Iberoamericano Iberdiscap, Costa Rica*. [S.l.: s.n.], 2004.

PRASAD, J. et al. Gesture recognition by stereo vision. In: ACM. *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*. [S.l.], 2010. p. 155–162.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *RITA*, v. 13, n. 2, p. 11–42, 2006.

ROBERTS, L. G. *MACHINE PERCEPTION OF THREE-DIMENSIONAL soups*. Tese (Doutorado) — Massachusetts Institute of Technology, 1963.

ROBOTREAM. *Obstacle Avoidance*. jun. 2013. Disponível em: <http://www.roborealm.com/tutorial/>.

RODRIGUES, C. Um dispositivo háptico de auxílio à navegação para deficientes visuais. *Trabalho de Graduacao ao Centro de Infromática da Universidade Federal de Pernambuco para a obtencao do grau de Bacharel em Ciência da Computacao*, 2006.

SEUBA, M.; JOAQUIM, P. Estimació trinocular de mapes de profunditat. Universitat Politècnica de Catalunya, 2009.

SHAMSI, M.; AL-QUTAYRI, M.; JEEDELLA, J. Blind assistant navigation system. In: IEEE. *Biomedical Engineering (MECBME), 2011 1st Middle East Conference on*. [S.l.], 2011. p. 163–166.

SILBERMAN, N.; FERGUS, R. Indoor scene segmentation using a structured light sensor. In: IEEE. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. [S.l.], 2011. p. 601–608.

SILVA, I. d.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas. *São Paulo: Artliber*, p. 33–111, 2010.

SOBEL, I. An isotropic 3× 3 image gradient operator. *Machine Vision for three-demensional Sciences*, Academic Press, 1990.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer, 2011.

TANDAYYA, P.; LIMNA, T.; SUVONVORN, N. Stereo vision utilizing parallel computing for the visually impaired. 2009.

TOMMASELLI, A. M. G. Fotogrametria básica. *Notas de aula de Fotogrametria. UNESP. Presidente Prudente*, 2009.

TRAJKOVIĆ, M.; HEDLEY, M. Fast corner detection. *Image and vision computing*, Elsevier, v. 16, n. 2, p. 75–87, 1998.

VISWANATHAN, D. G. *Features from Accelerated Segment Test (FAST)*. [S.l.]: nd, 2011.

WENQIN, S.; WEI, J.; JIAN, C. A machine vision based navigation system for the blind. In: IEEE. *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*. [S.l.], 2011. v. 3, p. 81–85.

WILSON, C. Distance estimation using stereo vision. 2009.

ZHANG, Z. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 22, n. 11, p. 1330–1334, 2000.

ZHU, M. et al. Stereo vision rectification based on epipolar lines match and three variables projective matrix. In: IEEE. *Integration Technology, 2007. ICIT'07. IEEE International Conference on*. [S.l.], 2007. p. 133–138.

Anexo A

QUESTIONÁRIO

Dos dias 7 a 8 de Outubro foram entrevistados cinco pessoas portadoras de algum tipo de deficiência visual de diferentes idades, genêros e tipo de deficiência visual. Foi mostrado aos participantes o funcionamento do protótipo deste trabalhoo e depois aplicado o questionário.

Formulário - Sistema de auxílio à navegação

O objetivo do trabalho é criar um protótipo de um sistema de auxílio à navegação para deficientes visuais em ambientes internos. O protótipo é composto por uma câmera que detecta possíveis obstáculos no caminho, como cadeiras, estantes, degraus entre outros e informa ao usuário o melhor caminho livre. Além de detectar obstáculos, o sistema detecta pontos de interesse marcados com um símbolo que é reconhecido e informado ao usuário, como por exemplo porta de banheiro, escadas, elevadores entre outros.
Ao responder este questionário, o trabalho será avaliado, e as informações obtidas serão utilizadas para projetos futuros.

1. **Um sistema que auxilia o usuário apenas em ambientes fechados ainda seria útil?**
   Ambientes fechados: casa, bibliotecas, restaurantes, hoteis, etc
   Mark only one oval.

   - ◯ Sim
   - ◯ Não

2. **Qual o seu interesse sobre o uso de um dispositivo que informa ao usuario possiveis obstaculos?**
   Mark only one oval.

   - ◯ Muito Baixo
   - ◯ Baixo
   - ◯ Médio
   - ◯ Alto
   - ◯ Muito Alto

3. **Qual o seu interesse sobre o uso de um dispositivo que informa ao usuario possiveis pontos de interesse, como portas de acesso, locais e objetos?**
   Mark only one oval.

   - ◯ Muito Baixo
   - ◯ Baixo
   - ◯ Médio
   - ◯ Alto
   - ◯ Muito Alto

4. **Qual a melhor forma que um aparelho pode se comunicar com um usuário deficiente visual?**
   Check all that apply.

   - [ ] Sons como bips
   - [ ] Som com vozes
   - [ ] Pulseiras vibrante
   - [ ] Pulseiras vibrantes com bips
   - [ ] Pulseiras vibrantes com vozes
   - [ ] Other: .....

5. **Em caso de áudio, qual a melhor maneira?**
   Pular essa pergunta se escolheu outra opção que não use áudio.
   Mark only one oval.

   - ◯ Fone nos dois ouvidos
   - ◯ Fone em apenas um ouvido

6. **Já utilizou algum sistema eletrônico de auxilio?**
   Mark only one oval.

   - ◯ Sim
   - ◯ Não

7. **Caso a resposta anterior seja SIM, como foi a experiência em utilizá-lo?**
   Caso a resposta da pergunta anterior tenha sido NÃO, favor ignorar esse pergunta.

   .....

8. **Você se importa com a aparência estética do dispositivo?**
   Mark only one oval.

   - ◯ Sim
   - ◯ Não

9. **Possui alguma sugestão para algum equipamento que poderia te ajudar?**

   .....

Anexo B

CÓDIGO-FONTE

```
1
2
3  %Declaração dos arquivos de audio referentes a detecção de obstáculos
4  [som_nenhum,Fs,NBITS]=wavread('audionavnenhum.wav');
5  [som_centro,Fs,NBITS]=wavread('audionavcentro.wav');
6  [som_direita,Fs,NBITS]=wavread('audionavdireita.wav');
7  [som_esquerda,Fs,NBITS]=wavread('audionavesquerda.wav');
8
9  %Declaração dos arquivos de audio e instancia dos objetos referentes aos
10 %pontos de interesse
11 obj1 = rgb2gray(imread('fotoporta2.png'));
12 [som_obj1,Fs,NBITS]=wavread('audioporta.wav');
13
14 obj2 = rgb2gray(imread('fotobanheiro2.png'));
15 [som_obj2,Fs,NBITS]=wavread('audiobanheiro.wav');
16
17 obj3 = rgb2gray(imread('fotoescada.png'));
18 [som_obj3,Fs,NBITS]=wavread('audioescada.wav');
19
20 obj4 = rgb2gray(imread('fotosaida.png'));
21 [som_obj4,Fs,NBITS]=wavread('audiosaida.wav');
22
23 obj5 = rgb2gray(imread('fotocopa.png'));
24 [som_obj5,Fs,NBITS]=wavread('audiocopa.wav');
25
26 %Atribuição das cameras do kinect
27 %camera de profundidade que encontrará obstáculos
28 depth = videoinput('kinect',2,'depth_640x480');
```

```
29 %camera rgb que encontrará pontos de interesse
30 rgb = videoinput('kinect',1,'rgb_640x480');
31 preview(depth)
32 preview(rgb)
33
34 %Controle para fechar a iteração caso um botão de fechar seja apertado
35 H = uicontrol('Style', 'PushButton', ...
36     'String', 'Break', ...
37     'Callback', 'delete(gcbf)');
38
39 count = 0;
40
41
42 %detecta os pontos de cada objeto
43 obj1points = detectSURFFeatures(obj1);
44 obj2points = detectSURFFeatures(obj2);
45 obj3points = detectSURFFeatures(obj3);
46 obj4points = detectSURFFeatures(obj4);
47 obj5points = detectSURFFeatures(obj5);
48
49
50 %extrai os pontos caracteristicos de cada objeto
51 [obj1_caracteristicas, obj1points] = extractFeatures(obj1, obj1points);
52 [obj2_caracteristicas, obj2points] = extractFeatures(obj2, obj2points);
53 [obj3_caracteristicas, obj3points] = extractFeatures(obj3, obj3points);
54 [obj4_caracteristicas, obj4points] = extractFeatures(obj4, obj4points);
55 [obj5_caracteristicas, obj5points] = extractFeatures(obj5, obj5points);
56
57
58 while (ishandle(H))
59     img_rgb = getsnapshot(rgb);
60     sceneImage = rgb2gray(img_rgb);
61
62     %
       ------------------------------------------------------------------------
63     %
       ------------------------------------------------------------------------
64     %detecta os pontos da cena
65     scenePoints = detectSURFFeatures(sceneImage);
66
67     %extrai os pontos caracteristicos da cena
```

```
68      [sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints)
            ;
69
70      %compara os pontos da cena e dos objetos
71      obj1par = matchFeatures(obj1_caracteristicas, sceneFeatures, '
            Prenormalized', true);
72      obj2par = matchFeatures(obj2_caracteristicas, sceneFeatures);
73      obj3par = matchFeatures(obj3_caracteristicas, sceneFeatures);
74      obj4par = matchFeatures(obj4_caracteristicas, sceneFeatures);
75      obj5par = matchFeatures(obj5_caracteristicas, sceneFeatures);
76
77      %verifica o numero de pontos encontrados
78      n1 = numel(obj1par);
79      n2 = numel(obj2par);
80      n3 = numel(obj3par);
81      n4 = numel(obj4par);
82      n5 = numel(obj5par);
83      %
            ----------------------------------------------------------------------------------------------------
84      %
            ----------------------------------------------------------------------------------------------------
85      %Captura do frame da iteração da camera de profundidade
86      img_depth = getsnapshot(depth);
87      %Conversão para a escala de cinza
88      I = mat2gray(img_depth);
89      %%Chama metodo de encontrar as bordas de possiveis objetos do cenario
90      BW2 = edge(I,'canny');
91
92      %%Função de incluir linhas verticais brancas na imagem,de baixo para
            cima e da esquerda para a direita, sendo que em
93      %%cada coluna de pixel da imagem a condição de parada é encontrar um
            pixel branco, e quando
94      %%é encontrado, a função percorre a próxima coluna
95      for j=1:640
96          i = 480;
97          while (i-1 >0) & (BW2(i,j) == 0)
98
99              BW2(i,j) = 1;
100             i = i-1;
101         end
102
```

```
103
104     end
105
106     %Função de erosão da imagem para retirar ruídos
107     se = strel('disk',11);
108     erodedBW = imerode(BW2,se);
109
110
111
112
113     %Função de segmentação da imagem, onde são separados os componentes
114     %separados.
115     cc = bwconncomp(erodedBW, 4);
116     graindata = regionprops(cc,'basic');
117     grain_areas = [graindata.Area];
118
119     %Função que isola o maior componente, que representa o maior caminho
120     %para o usuário passar
121     [max_area, idx] = max(grain_areas);
122     grain = false(size(erodedBW));
123     grain(cc.PixelIdxList{idx}) = true;
124
125
126     %A imagem é dilatada para realçar o componente para compensar a erosão
127     %feita anteriormente
128     Br = imdilate(grain,se);
129         pixelmaisalto = 0;
130         auxj=0;
131
132     %Função de encontrar o ponto branco mais alto da imagem
133     for j=50:640
134         i = 480;
135         while (i-1 >0) & (Br(i,j) == 1)
136
137
138             if i > pixelmaisalto
139                 pixelmaisalto=i;
140                 auxj=j;
141             end
142             i = i-1;
143         end
144
145
```

```
146     end
147
148     pixelmaisalto
149     auxj
150
151     %Atributos que representam o número de pixels em cada região da imagem
152     %que foi dividida em esquerda, centro e direita
153     %A imagem de resolução de 640x480 foi divida em 3 partes
154     esquerda = Br(1:480, 1:210);
155     centro = Br(1:480, 211:430);
156     direita = Br(1:480, 431:640);
157     esq = 0;
158     cen = 0;
159     dir = 0;
160
161     %
            ==========================================================================
162     %FOR PARA VERIFICAR A QUANTIDADE DE PIXELS BRANCOS NA ESQUERDA%
163     for j=1:210
164         i = 480;
165         while (i-1 >0)
166             if esquerda(i,j) == 1
167                 esq = esq + 1;
168             end
169             i = i-1;
170         end
171
172     end
173     %
            ==========================================================================
174     %
            ==========================================================================
175     %FOR PARA VERIFICAR A QUANTIDADE DE PIXELS BRANCOS NO CENTRO%
176     for j=1:220
177         i = 480;
178         while (i-1 >0)
179             if centro(i,j) == 1
180                 cen = cen + 1;
181             end
182             i = i-1;
```

```
183         end
184
185     end
186     %
            ==========================================================================

187     %
            ==========================================================================

188     %FOR PARA VERIFICAR A QUANTIDADE DE PIXELS BRANCOS NA DIREITA%
189     for j=1:210
190         i = 480;
191         while (i-1 >0)
192             if direita(i,j) == 1
193                 dir = dir + 1;
194             end
195             i = i-1;
196         end
197
198     end
199     %
            ==========================================================================

200
201     %Abaixo, as condições para encontrar pontos de interesse e melhores
202     %caminhos
203
204     %Condição para encontrar o objeto 1, onde n deve ser maior que 20 pares
205     if (n1 > 10) & count <= 0;
206         count = 5;
207         n1
208         texto = 'OBJETO UM ENCONTRADO';
209         disp(texto);
210         count
211
212
213         obj1_pontos_encontrados = obj1points(obj1par(:, 1), :);
214         matchedScenePoints = scenePoints(obj1par(:, 2), :);
215
216         [tform, inlier_obj1_Points, inlierScenePoints] =
                estimateGeometricTransform(obj1_pontos_encontrados,
                matchedScenePoints, 'affine');
217
```

```
218         obj1_Polygon = [1, 1;...
219             size(obj1, 2), 1;...
220             size(obj1, 2), size(obj1, 1);...
221             1, size(obj1, 1);...
222             1, 1];
223 
224         new_obj1_Polygon = transformPointsForward(tform, obj1_Polygon);
225 
226 
227         %exibe o objeto contornado com uma linha amarela
228         figure(2)
229         imshow(sceneImage);
230         line(new_obj1_Polygon(:, 1), new_obj1_Polygon(:, 2), 'Color', 'y');
231         %Audio do objeto 1 sendo reproduzido
232         sound(som_obj1,Fs,NBITS);
233 
234         %Condição para encontrar o objeto 2, onde n deve ser maior que 20
                pares
235     elseif (n2 > 20) & count <= 0;
236         count = 5;
237 
238         n2
239         texto = 'OBJETO DOIS ENCONTRADO.';
240         disp(texto);
241 
242         count
243         obj2_pontos_encontrados = obj2points(obj2par(:, 1), :);
244         matchedScenePoints = scenePoints(obj2par(:, 2), :);
245 
246         [tform, inlier_obj2_Points, inlierScenePoints] =
                estimateGeometricTransform(obj2_pontos_encontrados,
                matchedScenePoints, 'affine');
247 
248 
249         obj2_Polygon = [1, 1;...
250             size(obj2, 2), 1;...
251             size(obj2, 2), size(obj2, 1);...
252             1, size(obj2, 1);...
253             1,1];
254         new_obj2_Polygon = transformPointsForward(tform, obj2_Polygon);
255 
256         figure(2)
257         imshow(sceneImage);
```

```
258         line(new_obj2_Polygon(:, 1), new_obj2_Polygon(:, 2), 'Color', 'g');
259         %Audio do objeto 2 sendo reproduzido
260         sound(som_obj2,Fs,NBITS);
261
262         %Condição para encontrar o objeto 3, onde n deve ser maior que 20
                pares
263     elseif (n3 > 20) & count <= 0;
264         count = 5;
265
266         n3
267         texto = 'OBJETO DOIS ENCONTRADO.';
268         disp(texto);
269
270         count
271         obj3_pontos_encontrados = obj3points(obj3par(:, 1), :);
272         matchedScenePoints = scenePoints(obj3par(:, 2), :);
273
274         [tform, inlier_obj3_Points, inlierScenePoints] =
                estimateGeometricTransform(obj3_pontos_encontrados,
                matchedScenePoints, 'affine');
275
276
277         obj3_Polygon = [1, 1;...
278             size(obj3, 2), 1;...
279             size(obj3, 2), size(obj3, 1);...
280             1, size(obj3, 1);...
281             1,1];
282
283         new_obj3_Polygon = transformPointsForward(tform, obj3_Polygon);
284
285         figure(2)
286         imshow(sceneImage);
287         line(new_obj3_Polygon(:, 1), new_obj3_Polygon(:, 2), 'Color', 'g');
288         %Audio do objeto 2 sendo reproduzido
289         sound(som_obj3,Fs,NBITS);
290
291     %Condição para encontrar o objeto 4, onde n deve ser maior que 20 pares
292     elseif (n4 > 20) & count <= 0;
293         count = 5;
294
295         n4
296         texto = 'OBJETO DOIS ENCONTRADO.';
297         disp(texto);
```

```
298
299         count
300         obj4_pontos_encontrados = obj4points(obj4par(:, 1), :);
301         matchedScenePoints = scenePoints(obj4par(:, 2), :);
302
303         [tform, inlier_obj4_Points, inlierScenePoints] =
                estimateGeometricTransform(obj4_pontos_encontrados,
                matchedScenePoints, 'affine');
304
305
306         obj4_Polygon = [1, 1;...
307             size(obj4, 2), 1;...
308             size(obj4, 2), size(obj4, 1);...
309             1, size(obj4, 1);...
310             1,1];
311
312         new_obj4_Polygon = transformPointsForward(tform, obj4_Polygon);
313
314         figure(2)
315         imshow(sceneImage);
316         line(new_obj4_Polygon(:, 1), new_obj4_Polygon(:, 2), 'Color', 'g');
317         %Audio do objeto 2 sendo reproduzido
318         sound(som_obj4,Fs,NBITS);
319
320     %Condição para encontrar o objeto 5, onde n deve ser maior que 20 pares
321     elseif (n5 > 10) & count <= 0;
322         count = 5;
323
324         n5
325         texto = 'OBJETO DOIS ENCONTRADO.';
326         disp(texto);
327
328         count
329         obj5_pontos_encontrados = obj5points(obj5par(:, 1), :);
330         matchedScenePoints = scenePoints(obj5par(:, 2), :);
331
332         [tform, inlier_obj5_Points, inlierScenePoints] =
                estimateGeometricTransform(obj5_pontos_encontrados,
                matchedScenePoints, 'affine');
333
334
335         obj5_Polygon = [1, 1;...
336             size(obj5, 2), 1;...
```

```
337             size(obj5, 2), size(obj5, 1);...
338             1, size(obj5, 1);...
339             1,1];
340 
341         new_obj5_Polygon = transformPointsForward(tform, obj5_Polygon);
342 
343         figure(2)
344         imshow(sceneImage);
345         line(new_obj5_Polygon(:, 1), new_obj5_Polygon(:, 2), 'Color', 'g');
346         %Audio do objeto 2 sendo reproduzido
347         sound(som_obj5,Fs,NBITS);
348 
349 
350         %Caso nenhum ponto de interesse for encontrado, será analisado o
351         %cenário com melhor caminho
352 
353         %Caminho da esquerda
354     elseif (esq > dir) & (esq > cen) & count<=0 & esq>130*210;
355         figure(2); imshow(esquerda);
356        % figure(2); imshow(BW2);
357         title('Esquerda','FontWeight','bold')
358         esq
359         %o contador é reiniciado para dar tempo ao sistema de reproduzir o
360         %audio
361         sound(som_esquerda,Fs,NBITS);
362         count=6;
363 
364         %Caminho do centro
365     elseif (cen > dir) & (cen > dir) & count <=0 & cen>130*220;
366         figure(2);imshow(centro);
367         %figure(2); imshow(BW2);
368         title('Centro','FontWeight','bold')
369         cen
370         %o contador é reiniciado para dar tempo ao sistema de reproduzir o
371         %audio
372         count=6;
373         sound(som_centro,Fs,NBITS);
374 
375         %Caminho da direita
376     elseif (dir > esq) & (dir > cen) & count<=0 & dir>130*210;
377         figure(2); imshow(direita);
378         %figure(2); imshow(BW2);
379         title('Direita','FontWeight','bold')
```

```
380             dir
381             sound(som_direita,Fs,NBITS);
382             %o contador é reiniciado para dar tempo ao sistema de reproduzir o
383             %audio
384             count=6;
385         elseif dir<130*210 & cen<130*220 & cen<130*210 & count<=0;
386             sound(som_nenhum,Fs,NBITS);
387             count =  6;
388         else
389             count = count -1;
390
391             figure(2)
392             imshow(BW2);
393         end
394
395 end
396
397 %Caso o botão de fechar seja apertado o programa é fechado.
398 if (~ishandle(H))
399     closepreview(depth);
400     closepreview(rgb);
401 end
```

Anexo C

LINKS DE VÍDEOS

Os links abaixo mostram três circuitos testados utilizando o sistema proposto.

Circuito 1 http://youtu.be/gn28ljuSKVQ

Circuito 2 http://youtu.be/bpoggAYlXiI

Circuito 3 http://youtu.be/Ps-WVj1vvIs