**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**
**CENTRO DE CIÊNCIAS AGRÁRIAS**
**DEPARTAMENTO DE COMPUTAÇÃO**

**ISAAC VIEIRA DE MORAIS**

# NAVEGAÇÃO DE UM ROBÔ CONTROLADO POR SMARTPHONE UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

ALEGRE
2014

**ISAAC VIEIRA DE MORAIS**

# NAVEGAÇÃO DE UM ROBÔ CONTROLADO POR SMARTPHONE UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Agrárias da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. MSc. Thiago Meireles Paixão.

ALEGRE

2014

ISAAC VIEIRA DE MORAIS

**NAVEGAÇÃO DE UM ROBÔ CONTROLADO POR SMARTPHONE UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS**

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Agrárias da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 10 de dezembro de 2014.

COMISSÃO EXAMIDADORA

**Prof. M.Sc Thiago Meireles Paixão**
**Universidade Federal do Espírito Santo**
**Orientador**

Jacson R. C. Silva

**Prof. M.Sc. Jacson Rodrigues Correia Silva**
**Universidade Federal do Espírito Santo**

**Profa. M.Sc. Valéria Alves da Silva**
**Universidade Federal do Espírito Santo**

*A meus pais, Herachton e Marta que me apoiaram e me deram força para seguir em frente durante toda graduação.*

## AGRADECIMENTOS

A Deus, por me permitir concluir esta etapa na minha vida.

A toda minha família, especialmente minha irmã Mays, e meus pais Herachton e Marta, que apesar de todas as dificuldades durante os anos de graduação nunca deixaram de me apoiar e acreditar que eu conseguiria chegar ao final da graduação.

A todos os professores do DCOMP, pelos ensinamentos transmitidos.

Ao professor Thiago, por ter aceitado o desafio de me orientar durante o desenvolvimento deste trabalho de conclusão de curso.

A todas as amizades construídas na faculdade, pelos trabalhos, festas, e histórias que nunca esquecerei.

A todos, muito obrigado.

**RESUMO**

Navegação autônoma é um dos tópicos fundamentais em robótica. Em síntese, trata-se do desenvolvimento de técnicas que possibilitam robôs se locomoverem “por conta própria”, o que é essencial para muitas aplicações. Nesse contexto, o controle de trajetória é uma tarefa essencial, que pode ser realizada através da interpretação geométrica do ambiente sem prévio conhecimento do mesmo. Ambientes internos (*indoor*), em geral, apresentam uma estrutura mais regular, o que possibilita um melhor entendimento espacial do ambiente. Corredores são exemplos clássicos de ambientes altamente estruturados. A proposta deste trabalho foi implementar uma estratégia de controle de trajetória de um robô (controlado por *smartphone*) utilizando técnicas de processamento digital de imagens para navegação em ambientes internos (corredores). Os resultados obtidos mostram que é possível a utilização de dispositivos móveis para sistemas de controle de trajetória de protótipos robóticos.

Palavras-chave: Navegação Autônoma; Controle de Trajetória; Processamento Digital de Imagens; Ambientes Internos; *Smartphone*.

**ABSTRACT**

Autonomous navigation plays a central role in the field of robotics. In short, it covers the development of techniques that enable robots move around "on their own", an essential task used in many applications. In this context, trajectory control is a critical task that can be performed by the geometric scene understating without prior knowledge of it. In general, indoor environments are more structured, which enables a better understanding of the spatial environment. Corridors are populars examples of highly structured environments. This work aims to implement a trajectory control strategy for a smartphone-driven robot using digital image processing techniques for navigation in indoor environments (corridors). Results show the feasibility of using mobile devices for trajectory control of robotic prototypes.

Keywords: Autonomous Navigation; Trajectory Control; Digital Image Processing; Indoor environments; Smartphone.

## LISTA DE FIGURAS

Figura 1: À esquerda um corredor em um ambiente interno, já na direita a mesma foto com marcações em vermelho indicando padrões de retas e, elipses verdes mostrando a interseção entre as retas detectadas. ..... 1
Figura 2: Navegação de veículos aéreos em miniatura em corredores utilizando detecção de ponto de fuga (BILLS et al., 2011). ..... 2
Figura 3: Navegação de um robô humanoide (FARAGASSO et al, 2013). ..... 3
Figura 4: Drone utilizado pela Amazon para realizar a entrega de mercadorias. Fonte: AMAZON, 2014 ..... 4
Figura 5: Visão geral do sistema de orientação de trajetória para navegação autônoma. ..... 11
Figura 6: à esquerda um quadro capturado pela câmera do smartphone. À direita a mesma imagem após a aplicação do método de detecção de bordas Canny ..... 13
Figura 7: Divisão das regiões de decisão na imagem ..... 16
Figura 8: Impossibilidade de realizar o cálculo do ponto do de fuga por não existir interseções de retas. Em azul a borda detectada pelo algoritmo ..... 17
Figura 9: Quadrantes da imagem para decisões heurísticas ..... 18
Figura 10: Caso 1 da decisão heurística. Os dois pontos que formam a semirreta estão no quadrante 2, assim, a semirreta é classificada como um tipo do quadrante 2. ..... 19
Figura 11: Caso 2 da decisão heurística. Os dois pontos que formam a semirreta estão em quadrantes diferentes. Assim, é calculada a distância de cada ponto em relação ao ponto central da imagem. O ponto que tiver a maior distância em relação ao centro da imagem é o que determina a classificação da reta. Neste exemplo, a semirreta é classificada como um tipo do quadrante 2. ..... 19
Figura 12: Visão geral dos componentes do protótipo robótico ..... 21
Figura 13: Estrutura física do robô (visão lateral) ..... 21
Figura 14: Estrutura físico do robô (visão traseira) ..... 22
Figura 15: Foto do Corredor 1. Na base da imagem uma linha horizontal cinza indicando o ponto de partida do robô ..... 27

Figura 16: Foto do Corredor 2. Na base da imagem uma linha horizontal vermelha indicando o ponto de partida do robô. ..... 28
Figura 17: Software desenvolvido para realização da análise off-line ..... 29
Figura 18: Execução do experimento no Corredor 1 ..... 36
Figura 19: Sequência capturada entre o quadro 18 e o quadro 19 relativo ao gráfico da Figura 16 ..... 37
Figura 20: Execução do experimento do Corredor 2 ..... 39

## LISTA DE TABELAS

Tabela 1: Características do Corredor 1 .....27
Tabela 2: Características do Corredor 2 .....27
Tabela 3: Parâmetros para o método Canny.....30
Tabela 4: Parâmetros para a Transformada de Hough Probabilística.....31
Tabela 5: Parâmetros de tempo .....32
Tabela 6: Parâmetros limiares.....33
Tabela 7: Multiplicadores .....33
Tabela 8: Resumo da execução da metodologia no Corredor 1 .....38
Tabela 9: Resumo da execução da metodologia no Corredor 2 .....40

## LISTA DE ABREVIATURAS E SIGLAS

ADT - Android Developer Tools

API - Application Programming Interface

CCD - Charge-Coupled Device

CD - Curva para a Direita

CE - Curva para a Esquerda

CSD - Curva Suave para a Direita

CSE - Curva Suave para a Esquerda

DC - Direct Current

DH - Diferença Horizontal

GPS - Global Positioning System

HSV - Hue Saturation Value

IA - Inteligência Artificial

IDE - Integrated Development Environment

OpenCV - Open Source Computer Vision Library

PDI - Processamento Digital de Imagens

PWM - Pulse-Width Modulation

RGB - Red Green Blue

SDK - Software Development Kit

SF - Siga em Frente

TaAD - Taxa de Acerto de Decisão

THP - Transformada de Hough Probabilística

**SUMÁRIO**

1 INTRODUÇÃO ..... 1

1.1 O PROBLEMA E SUA IMPORTÂNCIA ..... 2
1.2 OBJETIVOS ..... 5
1.2.1 Objetivo geral ..... 5
1.2.2 Objetivos específicos ..... 5
2 REVISÃO DA LITERATURA ..... 6

2.1 NAVEGAÇÃO AUTÔNOMA ..... 6
2.1.1 Paradigma Hierárquico/Deliberativo ..... 6
2.1.2 Paradigma Reativo ..... 6
2.1.3 Paradigma Híbrido ..... 7
2.2 NAVEGAÇÃO AUTÔNOMA BASEADA EM PONTO DE FUGA ..... 7
2.2.1 Ambientes internos ..... 7
2.2.2 Ambientes externos ..... 8
2.3 DETECÇÃO DE PONTO DE FUGA ..... 9
2.3.1 Abordagem baseada em Esfera de Gauss ..... 9
2.3.2 Abordagem baseada em Espaço de Imagem ..... 10
2.3.3 Abordagem baseada em Espaço de Hough ..... 10
3 METODOLOGIA ..... 11

3.1 PROCESSO DE NAVEGAÇÃO ..... 11
3.1.1 Captura da cena ..... 12
3.1.2 Extração de informação ..... 12
3.1.3 Tomada de decisão ..... 16
3.2 PROTÓTIPO ROBÓTICO ..... 19
3.2.1 Software ..... 20
3.2.2 Hardware ..... 20
3.3 TECNOLOGIAS UTILIZADAS ..... 24
4 RESULTADOS EXPERIMENTAIS ..... 26

4.1 Descrição Geral dos Experimentos ..... 26

4.2 Configuração dos Parâmetros ..... 29
4.2.1 Ajustes na detecção de bordas ..... 29
4.2.2 Ajustes na detecção de semirretas ..... 30
4.2.3 Ajustes de tempo ..... 31
4.2.4 Ajustes de limiares e multiplicadores ..... 32
4.3 Critérios de Validação ..... 34
4.3.1 Taxa de acerto de decisão (TaAD) ..... 34
4.3.2 Diferença Horizontal (DH) ..... 34
4.4 Análise dos resultados ..... 35
4.4.1 Experimento 01 ..... 35
4.4.2 Experimento 02 ..... 38
5 CONCLUSÃO ..... 41
6 REFERÊNCIAS ..... 43

# 1 INTRODUÇÃO

A autonomia de um robô diz respeito ao seu poder de realização de tarefas sem a contínua interferência humana. Em navegação autônoma, particularmente, o objetivo é o desenvolvimento de técnicas que possibilitem robôs se locomoverem “por conta própria” de forma segura, uma habilidade essencial para diversas aplicações.

Singhal (1997) descreve o problema de navegação sobre duas perspectivas gerais: navegação em ambientes internos (*indoor*) e navegação em ambientes externos (*outdoor*). Em geral, ambientes internos são mais estruturados, ou seja, mais ricos em padrões que servem de referência para localização e orientação do robô. Corredores são exemplos típicos de ambientes internos estruturados. Trata-se de caminhos retos que se ligam formando junções em ângulos retos, conforme pode ser observado na Figura 1.

Figura 1: À esquerda um corredor em um ambiente interno, já na direita a mesma foto com marcações em vermelho indicando padrões de retas e, elipses verdes mostrando a interseção entre as retas detectadas.

Para se locomover, um robô deve perceber o ambiente através de sensores. Os mais comuns em navegação são: sensores infravermelho e ultrassom (sensores ativos) e sensores de imagem (sensores passivos). O Dispositivo de Carga Acoplada (CCD, do inglês *Charge-Coupled Device*) é o sensor de imagem presente em câmeras digitais. O uso da imagem digital como fonte de dados caracteriza a navegação baseada em visão, tema amplamente abordado na literatura (FARAGASSO, 2013; SANTOSH, 2008; KAISER, 2010).

No exemplo do corredor (Figura 1), pode se perceber da imagem que a junção das paredes com o piso caracterizam linhas que no plano projetivo tendem a convergir para uma localidade chamada *ponto de fuga*. Algoritmos para detecção de ponto de fuga são bastante utilizados para navegação em ambientes internos para estimar a orientação do robô em relação ao caminho percorrido (Ebrahimpour et al., 2009).

A proposta deste trabalho é implementar uma estratégia de controle de trajetória de um robô baseada em detecção do ponto de fuga considerando como escopo de aplicação ambientes internos (p. ex., corredores). Temas como tratamento de colisões, mapeamento de ambiente ou busca por um alvo específico não foram abordados neste trabalho.

## 1.1 O PROBLEMA E SUA IMPORTÂNCIA

Em certas aplicações, é desejável que robôs sejam capazes de se locomoverem em ambientes internos sem o prévio conhecimento dos mesmos. Nesse tipo de situação, a intepretação geométrica do ambiente pode servir como base para correção ou mesmo controle de trajetória.

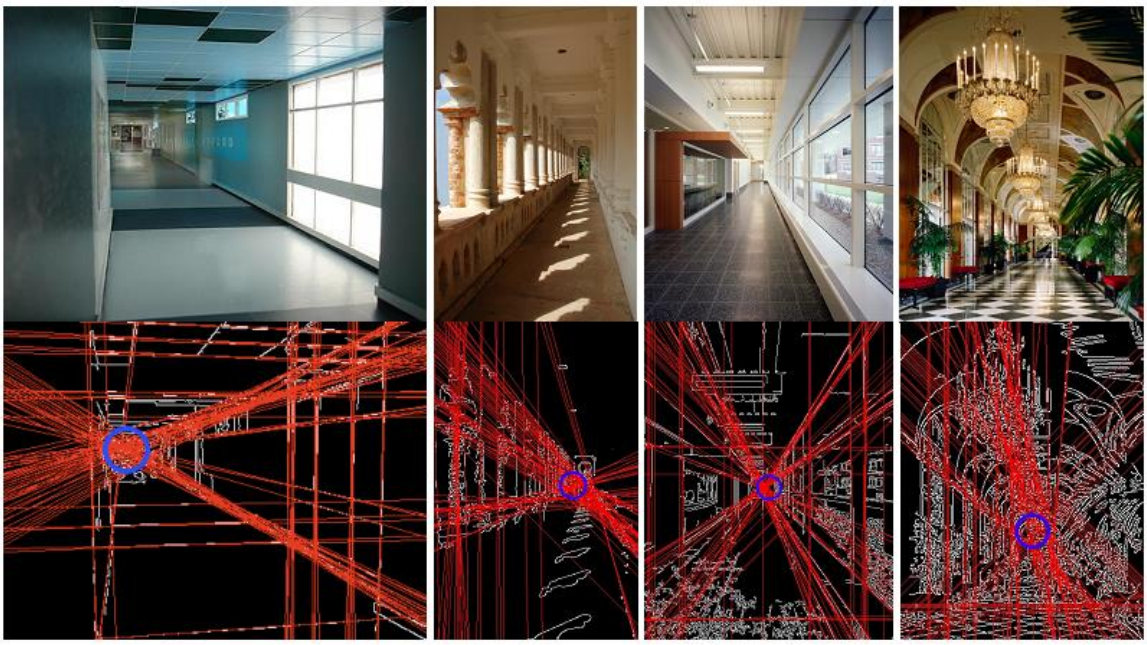

Figura 2: Navegação de veículos aéreos em miniatura em corredores utilizando detecção de ponto de fuga (BILLS et al., 2011).

Bills et al. (2011), por exemplo, tratam do problema de navegação de veículos aéreos em miniatura, mais especificamente veículos do tipo *Parrot AR Drone*[1] que podem navegar por corredores e escadas sem a necessidade da construção de um modelo espacial 3D. Os autores utilizam detecção do ponto de fuga para a navegação em corredores, a exemplo deste trabalho (ver Figura 2).

Outro exemplo interessante que demonstra a aplicabilidade da interpretação geométrica para navegação é o trabalho de Faragasso et al. (2013). Na ocasião, aborda-se a navegação em robôs humanoides (robôs com formas de seres humanos) em labirintos de corredores utilizando apenas a informação visual , como se pode ver na Figura 3.

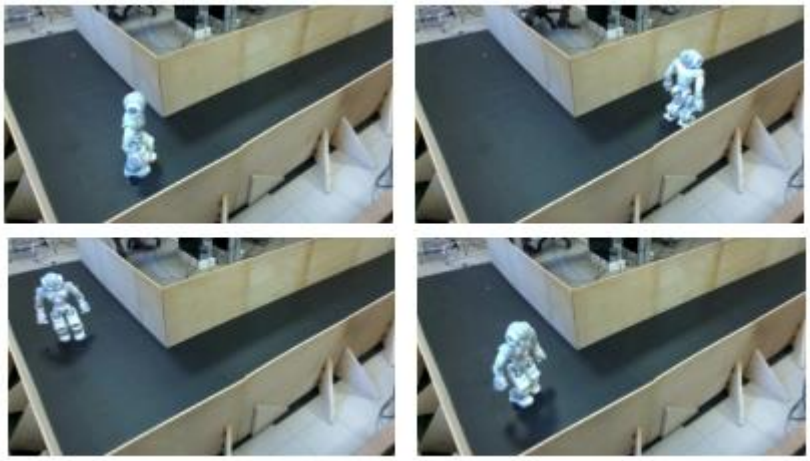

Figura 3: Navegação de um robô humanoide (FARAGASSO et al, 2013).

A abordagem em questão pode servir para extensão de novos serviços baseados em robótica. A CNN publicou em seu site em Dezembro de 2013 que a Amazon, empresa de comércio eletrônico, utiliza *drones* para realizar entrega de produtos (Figura 4). Os veículos são capazes de percorrer áreas externas guiados pelo Sistema de Posicionamento Global (*Global Positioning System* – GPS). Porém, para navegar em ambientes internos o GPS não apresenta alta precisão e não oferece informações a respeito da estrutura física do ambiente, como obstáculos, paredes, escadas, dentre outros.

[1] *Drone* é um termo em inglês para designar um Veículo Aéreo Não Tripulado (VANT).

Figura 4: Drone utilizado pela Amazon para realizar a entrega de mercadorias. Fonte: AMAZON, 2014

Deve-se enfatizar o interesse neste trabalho pela utilização de *smartphone*. A evolução na capacidade de processamento dos dispositivos móveis têm permitido experiências diversas relacionadas à aquisição e entendimento de cenas, como auxílio na direção de veículos (REN et al. 2013) e o auxílio na orientação de deficientes visuais (ASAD et al., 2012). A metodologia empregada neste trabalho pode trazer avanços para aplicações envolvendo dispositivos móveis que não envolvam necessariamente a construção de robôs.

Outros fatores interessantes para a adoção desta tecnologia podem ser listados: maior autonomia energética (principalmente se comparado com laptops), diferentes tecnologias integradas aos dispositivo (p. ex., GPS, internet e sensor de gravidade), e facilidade de disseminação de aplicativos através de sites especializados, como por exemplo o Google Play (PLAY, 2014).

Para a aplicação da técnica desenvolvida neste trabalho, foi construído um protótipo robótico, que será detalhado no capítulo 3, baseado na plataforma de prototipagem eletrônica Arduino.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo geral

O objetivo geral deste trabalho é implementar o controle de trajetória de um robô utilizando técnicas de Processamento Digital de Imagem (PDI).

### 1.2.2 Objetivos específicos

a) Utilização de algoritmos de PDI para interpretação geométrica da cena (detecção de ponto de fuga).
b) Construção de um protótipo robótico utilizando a plataforma de prototipagem Arduino e *smartphone*.
c) Implementação de algoritmos de controle para o microcontrolador do Arduino.
d) Implementação da interface de comunicação *Bluetooth* entre o smartphone e o robô.
e) Realização de experimentos para validação da técnica implementada.

# 2 REVISÃO DA LITERATURA

Detecção de trajetória (ou cálculo do caminho) para navegação autônoma em robôs é uma tarefa que pode ser realizada através de sensores ativos (e.g., infravermelho e ultrassom) e sensores passivos (e.g., câmera digital). De Cristóforis et al. (2013) argumentam que além da viabilidade financeira, a utilização de câmeras fornecem dados em maior resolução (imagens digitais), além de possuir um maior alcance de captura de informação. Na literatura, trabalhos como os de Hota et al. (2009), Baumgartner et al. (1994) e Wu et al. (2011) evidenciam diversas abordagens baseadas em análise de imagens para detecção de trajetória.

## 2.1 NAVEGAÇÃO AUTÔNOMA

Para Murphy (2000) a navegação autônoma pode ser organizada em três paradigmas. Ambos os paradigmas devem seguir três primitivas básicas: sentir (*sense*), planejar (*plan*), e atuar (*act*). A seguir segue uma sinopse de cada um dos paradigmas.

### 2.1.1 Paradigma Hierárquico/Deliberativo

É o paradigma que dominou as implementações de 1967 ao fim da década de 1980, sendo considerado historicamente mais antigo. Utiliza uma abordagem *top-down*, ou seja, de cima para baixo, com ênfase na fase de planejamento. Assim, robôs que utilizam este paradigma, se comportam observando o mundo, a partir da observação é realizado o planejamento da próxima ação e, por fim, a ação é executada (MURPHY, 2000).

### 2.1.2 Paradigma Reativo

Segundo Murphy (2000), a redução dos custos para obtenção de *hardware* de computadores junto com o aumento do poder computacional, aliado ao crescente interesse por parte dos pesquisadores por estudos de Inteligência Artificial (IA) inspiradas na biologia e psicologia cognitiva, guiou a criação de um novo paradigma que então desconsiderou a etapa de planejamento existente no paradigma hierárquico, estabelecendo assim, uma abordagem fundamentada em sensação e ação assumindo que a saída de um sensor (*sense*) é a entrada para uma ação (*act*).

O paradigma reativo obteve grande sucesso sendo usado massivamente entre os anos de 1988 e 1992.

### 2.1.3 Paradigma Híbrido

A partir da década de 1990 começou-se a pensar em sistemas capazes de planejar trajetórias ótimas, construir mapas, monitorar o próprio desempenho, entre outras características que demandavam um comportamento deliberativo, o que se tornou um problema com a abordagem do paradigma reativo. Para solucionar o problema foi criado o paradigma híbrido com o objetivo de reunir as principais vantagens do paradigma deliberativo, que é a capacidade de planejar as ações, e paradigma reativo, que é a resposta em tempo real aos estímulos do ambiente. O paradigma híbrido realiza o planejamento a partir da decomposição uma tarefa em várias sub-tarefas, definindo para cada sub-tarefa um comportamento a ser executado como no paradigma reativo (MURPHY, 2000).

## 2.2 NAVEGAÇÃO AUTÔNOMA BASEADA EM PONTO DE FUGA

DeSouza e Kak (2002) dividem a navegação robótica em ambientes internos e ambientes externos apresentando uma revisão literária das principais abordagens utilizadas até então.

### 2.2.1 Ambientes internos

Baumgartner et al. (1994) propôs um sistema de navegação onde a detecção de trajetória é realizada a partir de pistas no decorrer do caminho de um robô. Estas pistas que tem um formato de argola são colocadas em locais discretos no ambiente de forma que podem ser rapidamente detectadas pelo filtro utilizado na técnica desenvolvida, chamado Kalman estendido. Nesta técnica, o caminho onde o sistema de navegação deve ser utilizado tem ser ensinado manualmente para o robô, assim, o filtro de Kalman consegue produzir uma estimativa da posição e da orientação do veículo de forma precisa.

Shi e Samarabandu (2006) sugere um método com vários níveis de processamento para detectar pontos de fuga em um corredor. Primeiro é aplicado o filtro sigma de Lee (1983 apud Shi e Samarabandu 2006) como pré-processamento

para corrigir ruídos, a partir daí são realizados os processamentos principais do método que são divididos em três níveis. O primeiro é um estágio de baixo nível de processamento, onde são extraídas linhas verticais longas baseadas no algoritmo *fast line finder* mostrado em Kahn et al.(1990 apud Shi e Samarabandu 2006). O segundo é o estágio intermediário de processamento que consiste em realizar uma combinação dos segmentos de linhas obtidos na etapa de baixo nível de processamento, utilizando para isso o algoritmo chamado *Subtractive Clustering* que utiliza estatísticas para definir um conjunto de aglomerado de pontos que definem as linhas de um corredor. A terceira etapa é o estágio de alto nível de processamento onde é realizado novamente o estágio de baixo nível de processamento e o resultado é comparado com os corredores previstos no fim do processamento do estágio intermediário e assim, a hipótese de localização da linha do corredor é rejeitada ou aceita. Se o processamento for aceito, é realizada a detecção do ponto de fuga que é obtido pelo algoritmo *Subtractive Clustering* calculado com base no corredor esquerdo e na linha de corredor calculada pela metodologia.

### 2.2.2 Ambientes externos

DE CRISTÓFORIS et al (2013) apresenta um método dividido em seis etapas que é processado com o auxilio de uma GPU (Graphics Processing Unit). A primeira etapa consiste em encontrar uma linha que divide o horizonte da imagem em uma parte superior e outra inferior, assim, é possível descartar a parte de cima da imagem simplificando a quantidade de processamento que deve ser realizado nas etapas subsequentes. Na segunda etapa, o espaço de cores RGB é convertido em um espaço de cores HSV, posteriormente, tanto a versão RGB quanto a versão HSV recebem a aplicação de um filtro da mediana para reduzir ruídos. A terceira é onde há a maior quantidade de processamento, nela é gerado um mapa de segmentos obtido pela segmentação do canal RGB filtrado na segunda etapa, então, é criada uma lista de segmentos baseada no mapa de segmento ou super-pixels (que neste método são regiões com cor e textura com certa uniformidade) e, além da criação desta lista, são calculados a média e a covariância para cada segmento tanto nos espaços de cor RGB quanto HSV. Na quarta etapa é gerada uma máscara binária resultante de um modelo de mistura de gaussianas que utilizam as informações do espaço RGB e HSV que então é comparado com cada super-pixel da imagem, onde

esta comparação é realizada a partir de um critério de classificação, que define se o super-pixel faz parte de um possível caminho ou não. O modelo de misturas de gaussianas é extraído a partir de uma região retangular de interesse que corresponde a uma área a frente do robô, assim, obtém-se uma amostra da aparência da estrada. A quinta etapa consiste em extrair um contorno do que foi classificado como estrada utilizando para completar possíveis espaços entre as regiões um filtro de abertura morfológica. A última etapa do método consiste em calcular pontos médios a partir do contorno obtido no processamento da etapa anterior e, assim, utilizar estes pontos médios para calcular as velocidades angulares e lineares do robô.

## 2.3 DETECÇÃO DE PONTO DE FUGA

Tai et al. (1993) mostra que os primeiros estudos sobre cálculo do ponto de fuga eram totalmente dirigidos a tentar achar pontos de fuga baseando-se na interseção acumulativa de linhas. Tai explica os métodos propostos por Bernard (1982), Magge et al. (1984), e Quan e Mohr (1989) não tratavam ruídos nas imagens, já as técnicas propostas por Collins e Weiss (1990), Liou e Jain (1986), Shigand et al. (1990) e Wang e Tsai (1990) adotaram estratégias que tratavam de alguma forma imagens ruidosas. Tai propôs um algoritmo qualitativo baseado em probabilidade para determinação do ponto de fuga, que difere da forma quantitativa das propostas anteriores.

Seo at al. (2005) divide as técnicas de detecção de ponto de fuga em três categorias. Técnicas baseadas em Esfera de Gauss, técnicas baseadas no Espaço de imagem, e técnicas baseadas em Espaço de Hough.

### 2.3.1 Abordagem baseada em Esfera de Gauss

O método da esfera de Gauss foi proposto por Bernard (1983) e parte do princípio que existe um ponto focal na câmera, um plano que projeta a imagem tridimensional do mundo real e uma esfera onde o que Bernard chama de Polo Norte é o ponto de fuga.

### 2.3.2 Abordagem baseada em Espaço de Imagem

Cantoni et al. (2001) propôs duas técnicas usando o conceito de espaço de imagem. A primeira abordagem utiliza estatística para procurar a curva senoidal correspondente ao ponto de fuga no espaço de parâmetros polar. Já a segunda abordagem utiliza diretamente o plano de imagem, realizando primeiro uma detecção de bordas, seguido de uma normalização e limiarização para eliminar redundância de informação gerando, então uma imagem binária, de onde é extraído um ângulo que será utilizado para desenhar uma linha reta que será acumulada no espaço de parâmetros, finalizando a etapa com a marcação dos candidatos a ponto de fuga.

### 2.3.3 Abordagem baseada em Espaço de Hough

O espaço de Hough foi utilizado em uma metodologia proposta por Tuyteaars et al. (1998) com a chamada Transformada de Hough em Cascata. Tuyteaars aplicou três transformadas de Hough de forma encadeada na imagem onde a primeira transformada de Hough filtra as linhas retas da imagem. A segunda transformada de Hough é aplicada sobre a Transformada de Hough anterior, assim, é formadas novas linhas que correspondem a pontos onde há interseção das linhas da primeira transformada de Hough. Por último aplica-se outra transformada de Hough sobre a imagem com o resultado da 2ª transformada e obtém-se assim uma linha de fuga baseada na interseção dos pontos resultantes da 2ª transformada de Hough.

Asad e Ikram (2012) propõe uma metodologia que utiliza a Transformada de Hough com três etapas onde a primeira se refere a extração de características baseada em detecção de bordas. A segunda etapa, que é a seleção de características é subdividida em outras duas etapas, que são respectivamente o mapeamento de linhas em destaque e, a seleção das linhas relevantes para o método junto com a conexão de linhas desconectadas devido a obstruções e mudanças de iluminação. A terceira etapa é relativa à tomada de decisão, onde o ponto de fuga calculado com base na mediana dos pontos de interseção das linhas obtidas na segunda etapa, possibilitam três ações diferentes: virar para esquerda, virar para direita, ou continuar a trajetória atual.

# 3 METODOLOGIA

A revisão da literatura expôs diversas abordagens para detecção de ponto de fuga em imagens digitais. Esta é a informação central a ser utilizada para guiar o robô em corredores. Nesse sentido, este capítulo descreve como a solução para o problema de navegação autônoma em corredores foi tratada e implementada. As seções subsequentes abordam com mais detalhes a proposta metodológica.

## 3.1 PROCESSO DE NAVEGAÇÃO

Todo processamento relativo à decisão de trajetória do robô será realizado pelo *smartphone*, cuja especificação técnica é descrita na seção 4.1. Para possibilitar o controle do robô, foi necessário construir um sistema capaz de determinar a trajetória correta a ser seguida. A Figura 5 mostra um diagrama esquemático do processo de orientação de trajetória proposto neste trabalho, onde as setas indicam o fluxo do processo.

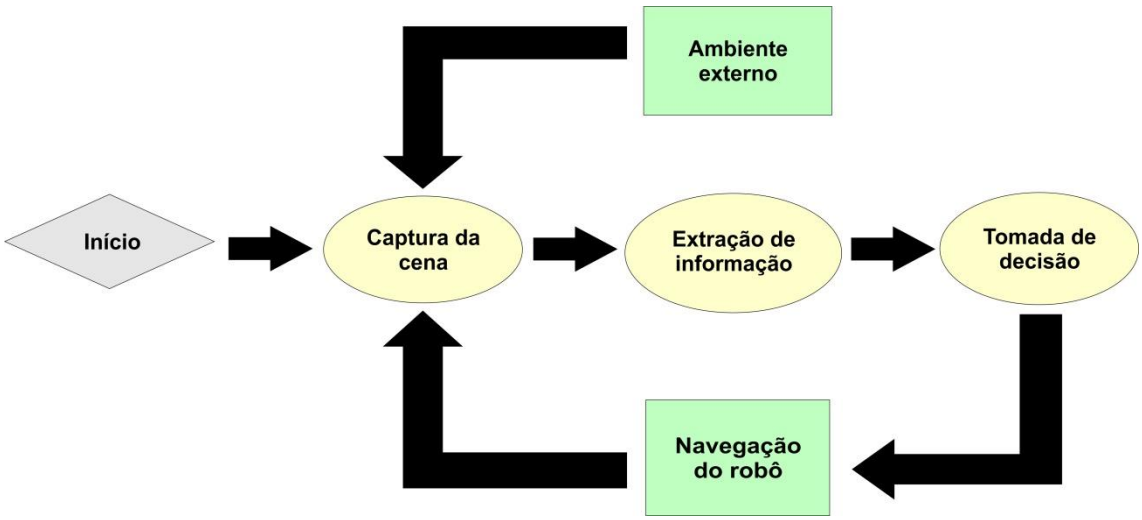


Figura 5: Visão geral do sistema de orientação de trajetória para navegação autônoma.

O procedimento de obtenção da orientação da trajetória pode ser dividido em três etapas que são detalhadas nos tópicos seguintes.

### 3.1.1 Captura da cena

Esta é a etapa inicial do processo de decisão da orientação da trajetória no sistema de navegação autônoma e consiste na captura de uma cena, ou quadro (*frame*), através de uma câmera.

A captura da cena é realizada pelo *software* de navegação no *smartphone*, utilizando uma interface disponibilizada pela *Open Source Computer Vision Library* (OpenCV) que realiza uma ponte (um *bridge*) entre a tecnologia Android e a biblioteca OpenCV.

O OpenCV só disponibiliza a captura de quadros com a configuração da orientação da câmera como paisagem (*landscape*). Na abordagem deste trabalho, a orientação necessária da câmera é a de retrato (*portrait*), por conta de como o *smartphone* é encaixado no protótipo robótico. Assim, para realizar a correção na imagem, é realizado uma operação de transposição de matriz no quadro analisado.

### 3.1.2 Extração de informação

Nesta etapa são utilizadas técnicas de PDI para realizar o tratamento da cena capturada pela câmera. Os passos necessários para realizar a extração do ponto de fuga são explicitados nas próximas sessões.

#### *3.1.2.1 Extração de semirretas*

O processo de extração de padrões de reta na imagem é composto pela extração de bordas de um quadro, seguido de uma aplicação da Transformada de Hough Probabilística (THP) para a obtenção de padrões de reta.

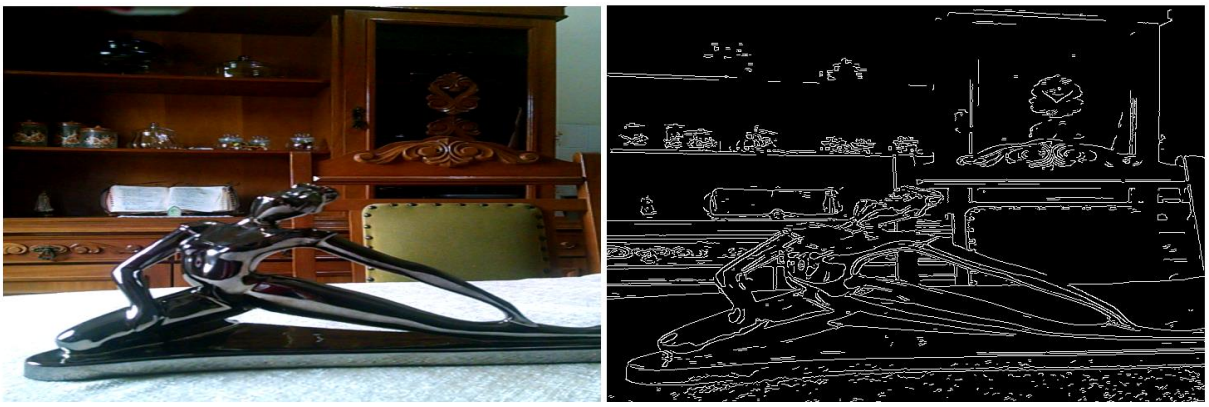

Figura 6: à esquerda um quadro capturado pela câmera do smartphone. À direita a mesma imagem após a aplicação do método de detecção de bordas Canny

Existem diversos métodos para obtenção de bordas em imagens digitais. Nesta abordagem foi escolhido o método Canny, que é considerado um bom método, como mostram Asad e Ikram (2012). O método Canny têm como resultado do processamento uma imagem preta-e-branca, onde os pixels brancos são as bordas detectadas na imagem analisada, como mostra a Figura 6.

A THP utiliza o quadro pós-processado pelo método Canny e determina padrões de semirreta contidos na imagem. Com o objetivo de facilitar a análise das imagens, os pontos calculados pela THP foram deslocados no espaço da imagem para que mais cruzamentos entre retas possam ser detectados visualmente. Isto serviu para visualizar o comportamento de semirretas de comprimento pequeno (que não ocupam todo espaço da imagem), no comprimento total da imagem. As formulações matemáticas de como os novos pontos $P'_0$ e $P'_1$ são calculados estão à mostra na Equação 01 e na Equação 02, respectivamente, onde :

- $x_0$ é a coordenada inicial x do ponto $P_0$
- $y_0$ é a coordenada inicial y do ponto $P_0$
- $x_1$ é a coordenada inicial x do ponto $P_1$
- $y_1$ é a coordenada inicial y do ponto $P_1$
- $\alpha$ é um fator de multiplicação constante
- $P'_0$ é o novo ponto $P_0$ deslocado no plano cartesiano
- $P'_1$ é o novo ponto $P_1$ deslocado no plano cartesiano

$$P'_0 = \left( (x_0 + \alpha * \frac{(x_1 - x_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}), ( y_0 + \alpha * \frac{(y_1 - y_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}) \right) \quad (1)$$

$$P'_1 = \left( (x_0 - \alpha * \frac{(x_1 - x_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}), ( y_0 - \alpha * \frac{(y_1 - y_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}) \right) \quad (2)$$

#### *3.1.2.2 Identificação de interseções entre retas*

Para identificar as interseções entre as retas calculadas pela THP foi utilizada uma abordagem de interpretação geométrica. Primeiro, foi realizada uma filtragem das semirretas usadas para calcular interseções estipulando um limiar (*threshould*) para descartar semirretas paralelamente próximas ao eixo vertical e ao eixo horizontal do plano cartesiano, assim, objetos como portas ou paredes frontais não influenciam no resultado na etapa final do processo.

Toda reta na abordagem deste trabalho possui quatro informações de coordenadas que descrevem os dois pontos necessários para estabelecermos uma semirreta. As informações de uma semirreta *A* e uma semirreta *B* podem ser utilizadas em conjunto com a Equação 03, conhecida como Equação Geral da Reta para estabelecer um ponto de interseção entre *A* e *B*. A seguir descreve-se como estabelecer tal ponto de interseção entre duas semirretas.

$$y = a * x + b \quad (3)$$

Sendo:

- $a_k$ a inclinação de uma reta $r_k$
- $b_k$ o deslocamento vertical da reta $r_k$
- $x_{k0}$ é a coordenada inicial x do ponto $P_0$ da reta $r_k$
- $y_{k0}$ é a coordenada inicial y do ponto $P_0$ da reta $r_k$
- $x_{k1}$ é a coordenada inicial x do ponto $P_1$ da reta $r_k$
- $y_{k1}$ é a coordenada inicial y do ponto $P_1$ da reta $r_k$

Temos que:

$$a_k = \frac{y_{k1} - y_{k0}}{x_{k1} - x_{k0}} \qquad (4)$$

$$b_k = y_{k1} - a_k * x_{k1} \qquad (5)$$

Assim, seja $a_i$ e $a_j$ as inclinações de duas retas $r_i$ e $r_j$, respectivamente, se $|a_i - a_j| \neq 0$, para duas semirretas, podemos fazer:

$$x_{ij} = \frac{b_j - b_i}{a_i - a_j} \qquad (6)$$

$$y_{ij} = a_i x_{ij} + b_i \qquad (7)$$

Onde $x_{ij}$ é a coordenada x e $y_{ij}$ é a coordenada y do ponto de interseção entre $r_i$ e $r_j$.

São calculadas as interseções para cada par de retas extraídas na subseção anterior, desde que a diferença entre o parâmetro de inclinação entre as retas seja maior que 0,1, valor que foi estabelecido depois de vários testes com diferentes ajustes e que apresentou bons resultados para a metodologia proposta.

#### *3.1.2.3 Identificação do ponto de fuga*

A última etapa do processo de orientação de trajetória é a obtenção das coordenadas que determinam o ponto de fuga em um quadro capturado pela câmera.

A solução proposta foi inspirada pelo trabalho de Asad e Ikram (2012) e consiste em realizar uma abordagem estatística para a escolha do ponto de fuga, seleciona-se um ponto mediano levando em consideração todas as interseções de reta calculadas.

O primeiro passo do processo é realizar a ordenação das interseções calculadas na subseção anterior. O critério usado para a ordenação dos pontos é o valor da coordenada do eixo horizontal do quadro analisado. Assim, os pontos são organizados em ordem crescente, onde o menor valor é o ponto mais a esquerda da imagem e o maior valor é o ponto mais à direita da imagem.

Realizada a ordenação, é indicada a coordenada que está localizada no meio dos valores ordenados, sendo que, nos casos onde a quantidade de interseções calculadas é um numero par é escolhida a posição à esquerda da posição que seria considerada o meio (e.g. quando há quatro interseções ordenadas, a posição mediana definida é a dois). Realizada a escolha do ponto de fuga é iniciada a etapa de tomada de decisão do sistema de orientação da trajetória.

### 3.1.3 Tomada de decisão

A partir do ponto de fuga estabelecido na etapa de extração da informação é realizada a tomada de decisão para a orientação da trajetória, baseada na posição do ponto de fuga no eixo horizontal da imagem.

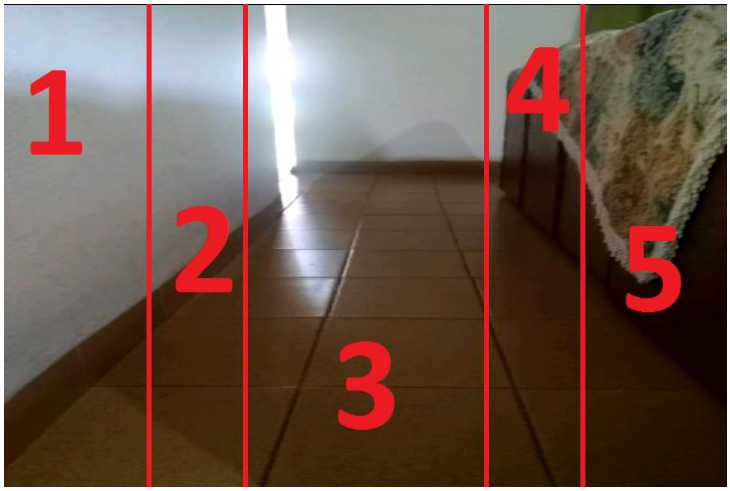


Figura 7: Divisão das regiões de decisão na imagem

Há seis tipos de decisão que o software do sistema de navegação implementado no *smartphone* pode gerar. A Figura 7 mostra o espaço da imagem com quatro linhas verticais que delimitam cinco regiões retangulares. Cada região é responsável por uma decisão de trajetória do sistema de navegação, também sendo possível executar uma decisão de parada do robô, caso não seja possível estabelecer uma trajetória, ou necessário limitar movimentos em um dado momento. As decisões possíveis e suas regiões correspondentes na imagem são:

- Siga em frente (região 3): Quando o ponto de fuga está localizado no centro da imagem, ou próximo do centro, de acordo com a margem de erro, é ordenada uma decisão de seguir em frente (SF) na trajetória atual.

- Vire à esquerda (região 1): Quando o ponto de fuga está em uma posição mais à esquerda da imagem, é executada uma decisão de correção de trajetória, comandando uma curva para a esquerda (CE);
- Vire levemente à esquerda (região 2): Quando o ponto de fuga está à esquerda da imagem, porém a uma distância relativamente pequena da decisão de “Siga em Frente”. É executada uma curva suave para a esquerda (CSE);
- Vire à direita (região 5): Quando o ponto de fuga está em uma posição mais à direita da imagem é executada uma decisão de correção de trajetória, comandando uma curva para a direita (CD);
- Vire levemente à direita (região 4): Quando o ponto de fuga está à direita da imagem, porém a uma distância relativamente pequena da decisão de “Siga em Frente”. É executada uma curva suave para a direita (CSD);
- Pare: Usado quando não é possível estabelecer uma orientação de trajetória, ou quando se deseja encerrar o processo movimento iniciado por alguma das outras decisões possíveis.

#### *3.1.3.1 Decisão heurística*

Nem sempre é possível detectar o ponto de fuga em um quadro. Situações onde só são detectadas semirretas paralelas na imagem impossibilitam a continuidade da navegação.

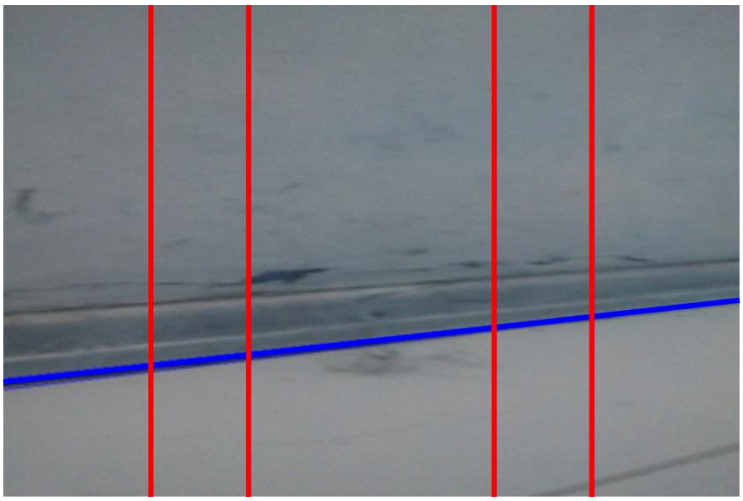

Figura 8: Impossibilidade de realizar o cálculo do ponto do de fuga por não existir interseções de retas. Em azul a borda detectada pelo algoritmo

A Figura 8 mostra uma situação onde não é possível detectar mais um padrão de reta com inclinações diferentes no plano cartesiano para que seja determinado um ponto de fuga. Para solucionar a decisão de trajetória nestes casos foi criada uma heurística de decisão.

A abordagem para a heurística de decisão consiste na subdivisão do espaço da imagem e classificação das semirretas detectadas de acordo com sua localização no espaço da imagem. A Figura 9 mostra a divisão de imagem em quatro quadrantes, que qualificam as retas detectadas e definem a orientação da trajetória.

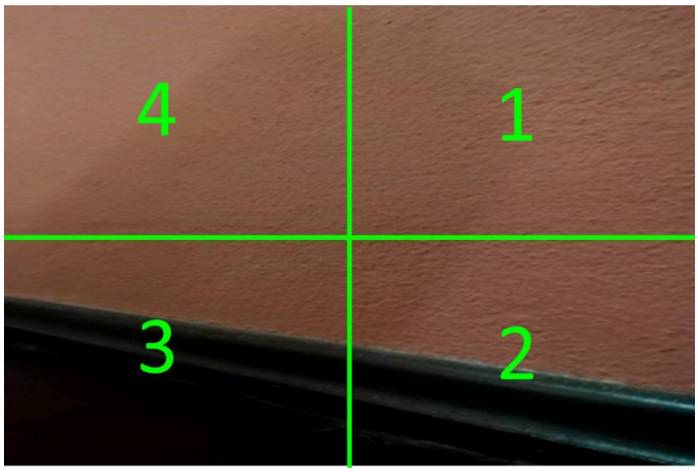


Figura 9: Quadrantes da imagem para decisões heurísticas

Dois casos são tratados pelo método heurístico. No primeiro caso, representado na Figura 10, os pontos que formam a semirreta estão localizados em um mesmo quadrante e a decisão de navegação tomada é para a direita - caso o quadrante seja relativo a parte esquerda da imagem, ou para esquerda - caso o quadrante seja relativo a parte direita da imagem. No segundo caso, como mostra a Figura 11, quando cada um dos dois pontos está em quadrantes diferentes, é calculada a distância euclidiana de cada ponto em relação ao centro da imagem. Assim, a decisão é tomada como no primeiro caso, sendo que o critério de decisão é tomado considerando o ponto mais distante do centro da imagem.

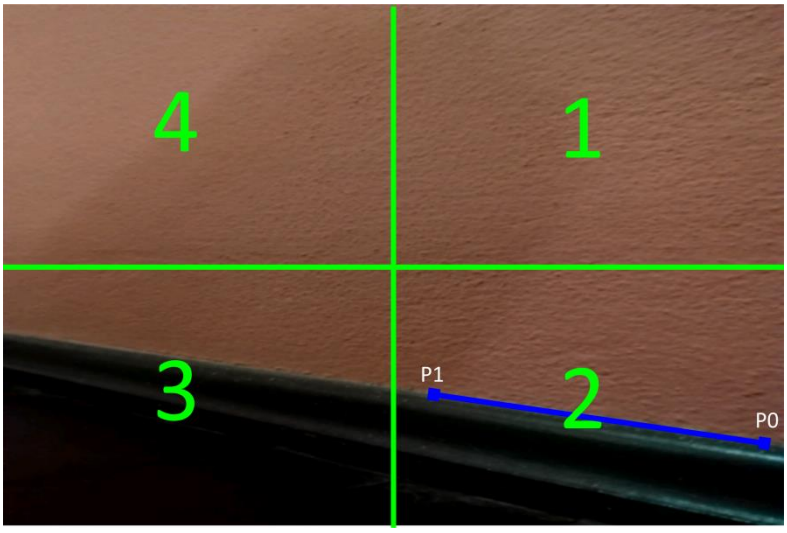


Figura 10: Caso 1 da decisão heurística. Os dois pontos que formam a semirreta estão no quadrante 2, assim, a semirreta é classificada como um tipo do quadrante 2.

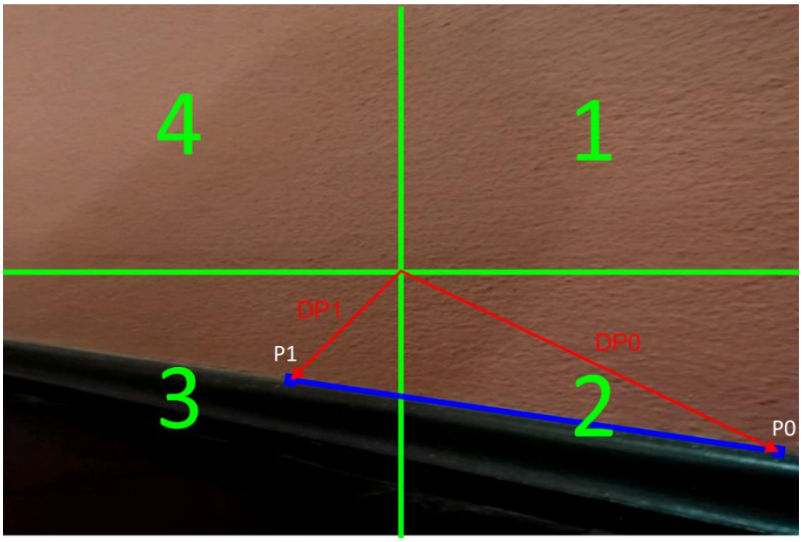


Figura 11: Caso 2 da decisão heurística. Os dois pontos que formam a semirreta estão em quadrantes diferentes. Assim, é calculada a distância de cada ponto em relação ao ponto central da imagem. O ponto que tiver a maior distância em relação ao centro da imagem é o que determina a classificação da reta. Neste exemplo, a semirreta é classificada como um tipo do quadrante 2.

## 3.2 PROTÓTIPO ROBÓTICO

O protótipo robótico é construído com base na plataforma de prototipagem eletrônica Arduino controlado pelo *smartphone* através de uma conexão estabelecida a partir de um módulo *bluetooth*.

### 3.2.1 Software

Cada uma das decisões disponíveis no sistema de orientação de trajetória do *smartphone* é transmitida para o robô através de dois parâmetros:

- Tipo de comando - é um valor de caractere (*flag*) que determina o tipo de movimento que o protótipo deve realizar. Os *flags* estabelecidos são:
  - *'e'* para sinalizar o comando "Vire à esquerda";
  - *'d'* para sinalizar o comando "Vire à direita";
  - *'s'* para sinalizar o comando "Siga em frente";
  - *'p'* para sinalizar o comando "Pare".
- Velocidade - Valor inteiro que varia de 0 a 255 e define respectivamente a intensidade mínima e máxima da corrente que será aplicada nos motores.

O sistema de navegação utiliza de um a dois comandos disponíveis na implementação de *software* do microcontrolador do robô. A seguir será definido como se dá a movimentação do protótipo para cada tipo de comando ordenado pelo sistema de navegação do smartphone:

- Vire à esquerda: Suspende o acionamento da roda esquerda e aciona a roda direita no sentido de rotação para frente.
- Vire à direita: Trava a roda direita e aciona a roda esquerda no sentido de rotação para frente.
- Pare: Suspende o acionamento da roda esquerda e da roda direita.
- Siga em frente: aciona a roda esquerda e a roda direita no sentido de rotação para frente.

### 3.2.2 Hardware

A locomoção do robô é realizada por duas rodas que são conectadas em dois motores de corrente contínua (motores DC – *Direct Current*) com acionamento independente. O acionamento dos motores é realizado por uma ponte H, controlada pelo microcontrolador do Arduino. A Figura 12 mostra um esquema geral da estrutura física do robô onde as setas indicam o fluxo de controle do sistema.

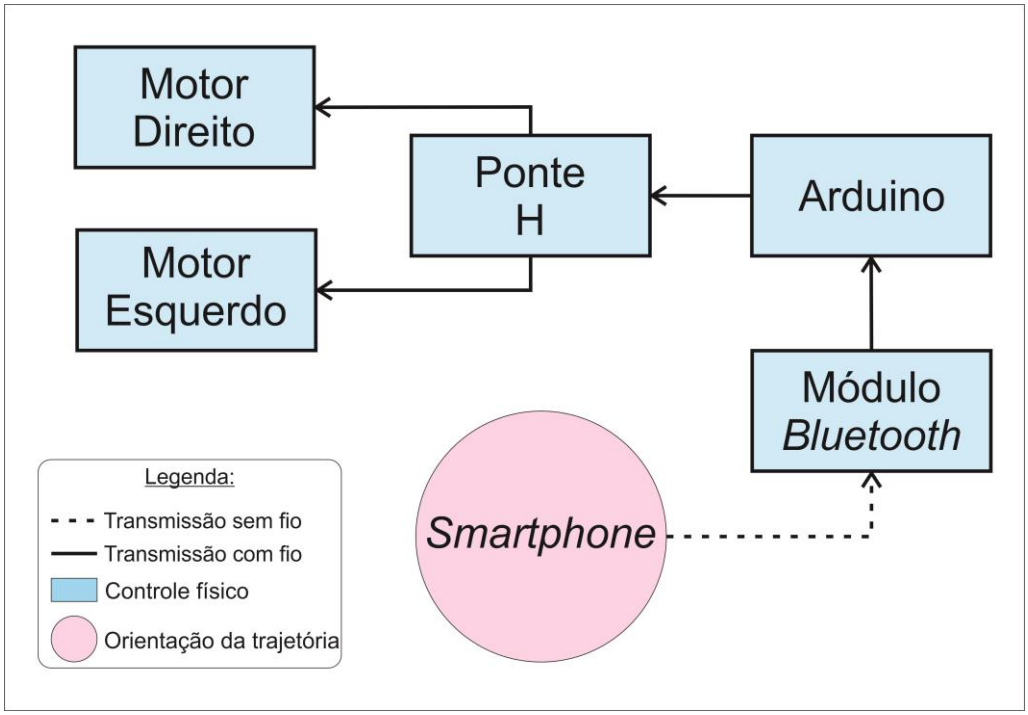


Figura 12: Visão geral dos componentes do protótipo robótico

Na Figura 12 pode-se observar que o sistema para a orientação da trajetória envia um comando de correção do caminho para o robô através da conexão *bluetooth*. A Figura 13 e a Figura 14 mostram a estrutura física do robô e a correspondência entre os componentes do modelo real do robô e a Figura 12.

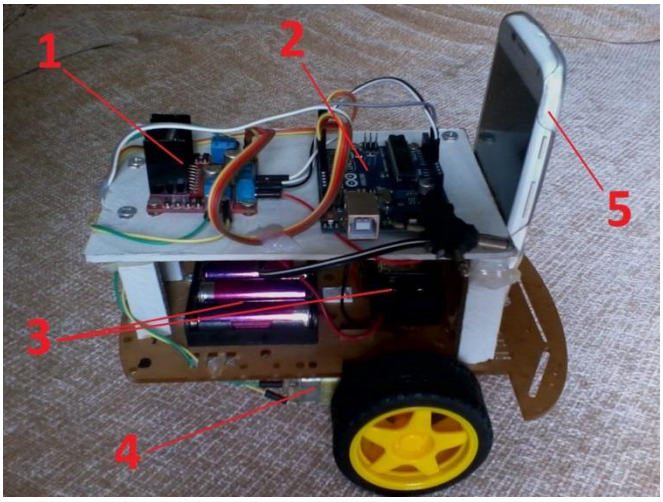


Figura 13: Estrutura física do robô (visão lateral)

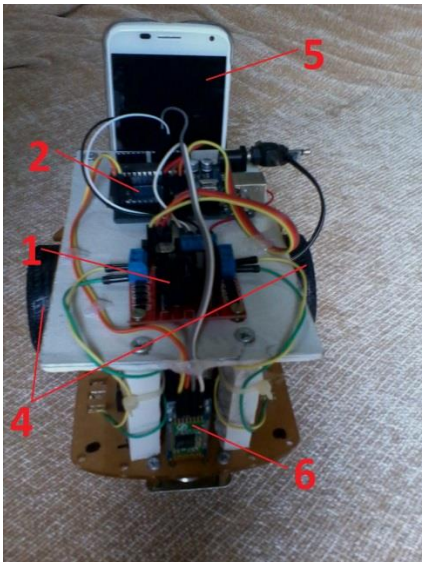


Figura 14: Estrutura físico do robô (visão traseira)

Os componentes enumerados na Figura 13 e Figura 14 são os seguintes:

1. Ponte H
2. Arduino Uno
3. Baterias
4. Conjunto de roda e motor
5. Smartphone
6. Módulo Bluetooth

Segue um resumo a seguir quanto à função de cada componente do controle físico do robô.

#### 3.2.2.1 Módulo *Bluetooth*

O módulo *bluetooth* é o *hardware* que realiza a comunicação do Arduino com o *Smartphone*. A tecnologia *Bluetooth* opera entre 2,4 GHz e 2,485 GHz, podendo ser classificada em três classes onde cada classe têm um alcance específico, sendo a Classe 2 a mais comumente usada, com alcance de 10 metros e com um consumo de potência da ordem de 2,5 mW (BLUETOOTH, 2014).

A escolha da comunicação *bluetooth* é justificada pela sua presença em grande parte dos dispositivos móveis atuais, o que possibilita boa portabilidade entre a parte física do robô e dispositivo responsável pela navegação.

#### 3.2.2.2 Arduino

O Arduino é uma plataforma de prototipagem eletrônica *open-source* criado com a proposta de disponibilizar *hardware* e *software* extensíveis e de fácil manipulação. O modelo utilizado neste projeto, o Arduino Uno R3 (Uno) (UNO, 2014), pode ter a energia fornecida por um cabo USB ou por uma fonte de alimentação externa.

O Uno possui um controlador ATmega328 com seis pinos de entradas analógicas, 14 pinos de entrada/saída digitais, sendo que seis destes pinos podem ser utilizados como saídas analógicas, também chamadas de saídas PWM (*Pulse-Width Modulation*) (ARDUINO, 2014).

A responsabilidade do Uno é controlar os motores responsáveis pela locomoção do robô, sendo todo processamento da orientação da trajetória realizado pelo dispositivo móvel, que envia as ações a serem tomadas pelo microcontrolador do Arduino através da conexão *bluetooth*.

O Arduino Uno foi escolhido por conter quantidade suficiente de entradas e saídas (analógicas e digitais) para o projeto do robô, sendo possível também a expansão futura das funcionalidades do protótipo por conta das portas que não serão utilizadas neste trabalho.

#### 3.2.2.3 Ponte H

A Ponte H é um circuito eletrônico que permite o controle de motores DC por um microcontrolador como o Arduino, por exemplo (FILIPEFLOP, 2014). Sua função é fornecer corrente suficiente para o correto acionamento dos motores, ao passo que a corrente necessária em um motor DC é normalmente maior que a corrente máxima disponível em uma saída PWM do Uno. Se um motor é alimentado diretamente pelo Uno, há a possibilidade do Arduino ser danificado por conta do excesso de corrente elétrica necessário na saída PWM.

Uma Ponte H pode atender diversos motores DC ao mesmo tempo e utiliza para o controle de cada um dos motores um sinal de entrada digital que realiza o controle do sentido da corrente elétrica, e uma entrada PWM utilizada para definir a

intensidade da corrente elétrica aplicada ao motor. Como a alimentação da Ponte H é independente da alimentação do Uno, é possível fornecer corrente suficiente para o correto funcionamento do motor, deixando para o Uno a responsabilidade de enviar um sinal digital e um analógico para o controle da direção e velocidade do motor, respectivamente.

#### 3.2.2.4 Motores DC

Os motores de corrente contínua são os atuadores do robô neste trabalho. A alimentação, assim como a velocidade e sentido de rotação, é aplicada pela Ponte H, que recebe do Arduino Uno um sinal digital para determinar o sentido de rotação e um sinal PWM para determinar a intensidade da corrente elétrica aplicada ao motor.

Geralmente um motor DC (HONDA, 2006) tem uma alta velocidade de rotação e baixo torque, por este motivo a utilização de uma caixa de redução, que é popularmente usada em veículos automotores, é necessária para que o torque seja suficiente para mover a estrutura do robô e a velocidade do motor seja reduzida.

## 3.3 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do *software* foram utilizados diferentes recursos tecnológicos. Abaixo seguem as principais tecnologias que foram utilizadas:

- Java: linguagem de programação fundamentada no paradigma de linguagem Orientada a Objetos (OO) necessária para criar o *software* de controle da trajetória, ao passo que esta é a linguagem de programação utilizada para o desenvolvimento de aplicações nativas no sistema operacional Android (JAVA, 2014).
- Android SDK: O Android *Software Development Kit* (SDK) oferece uma *Application Programming Interface* (API) com várias bibliotecas e ferramentas para o desenvolvimento de aplicativos para dispositivos móveis que utilizam o sistema operacional Android. A parte da implementação relacionada ao uso de recursos do *smartphone* utiliza esta ferramenta (ANDROID, 2014).
- Eclipse IDE: Para o desenvolvimento do *software* de controle de trajetória é utilizado o Eclipse *Integrated Development Environment* (IDE), que é uma ferramenta de desenvolvimento de software com suportes a linguagem de

programação Java, e ao Android SDK através da utilização *plugin Android Developer Tools* (ADT) *Bundle* (DEVELOPER, 2014).

- OpenCV: Para a manipulação de imagens é utilizada a biblioteca *Open Source Computer Vision* (OpenCV). Esta biblioteca possui compatibilidade com o sistema operacional Android e tem diversos algoritmos otimizados para diversos tipos de processamentos de imagens. O objetivo do uso da biblioteca é facilitar a implementação dos processamentos de imagem necessários para a obtenção da orientação da trajetória na navegação do robô (OPENCV, 2014).
- AVR C: É a linguagem utilizada para o desenvolvimento de *software* que é inserido no microcontrolador do Arduino. É uma linguagem de código aberto que pode ser expandida através de bibliotecas escritas na linguagem C++ (ARDUINO, 2014).
- Arduino IDE: Ambiente de desenvolvimento utilizado para a criação de *software* para o microcontrolador do Arduino. O IDE possibilita a escrita do algoritmo de controle no microcontrolador do robô (ARDUINO, 2014).

# 4 RESULTADOS EXPERIMENTAIS

Este capítulo mostra nas seções subsequentes como foram realizados os experimentos, quais critérios foram considerados nos resultados, quais os valores dos parâmetros da metodologia, e quais foram os resultados foram obtidos.

## 4.1 DESCRIÇÃO GERAL DOS EXPERIMENTOS

Para realizar os experimentos, o protótipo robótico foi posicionado no centro do corredor em uma posição paralela às paredes laterais. Não foi considerado no experimento o controle de iluminação, aderência do piso, ou qualquer outro fator relacionado às características do ambiente externo.

Para aferir a distância do corredor e a distância percorrida pelo robô foi utilizada uma trena com três metros de comprimento. Nos casos onde a medida extrapolou este comprimento, foram marcados pontos de três em três metros no chão para facilitar a medida tomando como referência estes pontos demarcados.

Os resultados apresentados neste capítulo foram obtidos com a realização da experimentação da metodologia desenvolvida em dois corredores distintos, para possibilidade de comparação. As características dos locais onde foram feitos os experimentos são mostrados na Tabela 1e Figura 15 para o primeiro corredor (Corredor 1), e na Tabela 2 e Figura 16 para o segundo corredor (Corredor 2) .

O primeiro experimento foi realizado no Corredor 1. A Tabela 1 descreve as características do corredor que é mostrado na Figura 15. Os testes neste corredor foram realizados no período noturno, assim, durante o experimento, a fonte de iluminação foi exclusivamente artificial. O piso do corredor tem pouca aderência, o que demandou um ajuste na velocidade do protótipo robótico, que é detalhada na seção de Configuração dos Parâmetros.

Tabela 1: Características do Corredor 1

| Dados do corredor | |
|---|---|
| **Comprimento:** | 318 cm |
| **Largura:** | 80 cm |
| **Tipo de iluminação:** | Artificial |

Figura 15: Foto do Corredor 1. Na base da imagem uma linha horizontal cinza indicando o ponto de partida do robô

No Corredor 2 existem alguns obstáculos (cadeiras, lixeiras e pessoas), como mostra a Figura 16.

Tabela 2: Características do Corredor 2

| Dados do corredor | |
|---|---|
| **Comprimento:** | 1400 cm |
| **Lagura:** | 295 cm |
| **Tipo de iluminação:** | Artificial e natural |

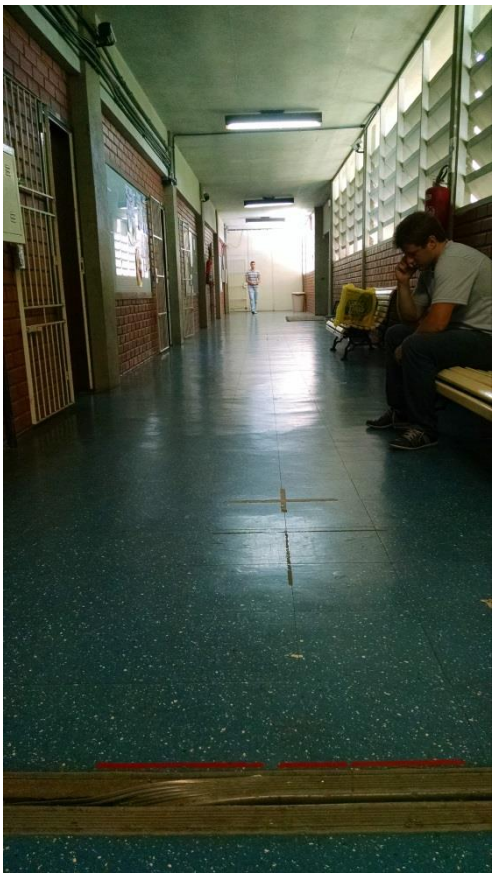

Figura 16: Foto do Corredor 2. Na base da imagem uma linha horizontal vermelha indicando o ponto de partida do robô.

Para o processamento da metodologia, foi utilizado um *smartphone* modelo *Moto X XT 1058* com processador de 1700 MHz *dual core*, memória RAM de 2GB LPDDR2, bluetooth versão 4.0, e câmera de 10,5 Mega Pixels. O *smartphone* é utilizado para realização da a captura das cenas e para o processamento da navegação do robô. A captura da imagem é realizada com uma resolução de 720x480 *pixels*. Durante a experimentação, não foram interrompidos nenhum serviço ativo no smartphone, seja do Sistema Operacional ou de software de terceiros.

Para realizar a análise off-line dos dados, foi desenvolvida uma aplicação para permitir que, para cada frame que um ponto de fuga tenha sido calculado, fosse possível estabelecer um ponto de fuga ideal através da análise humana.

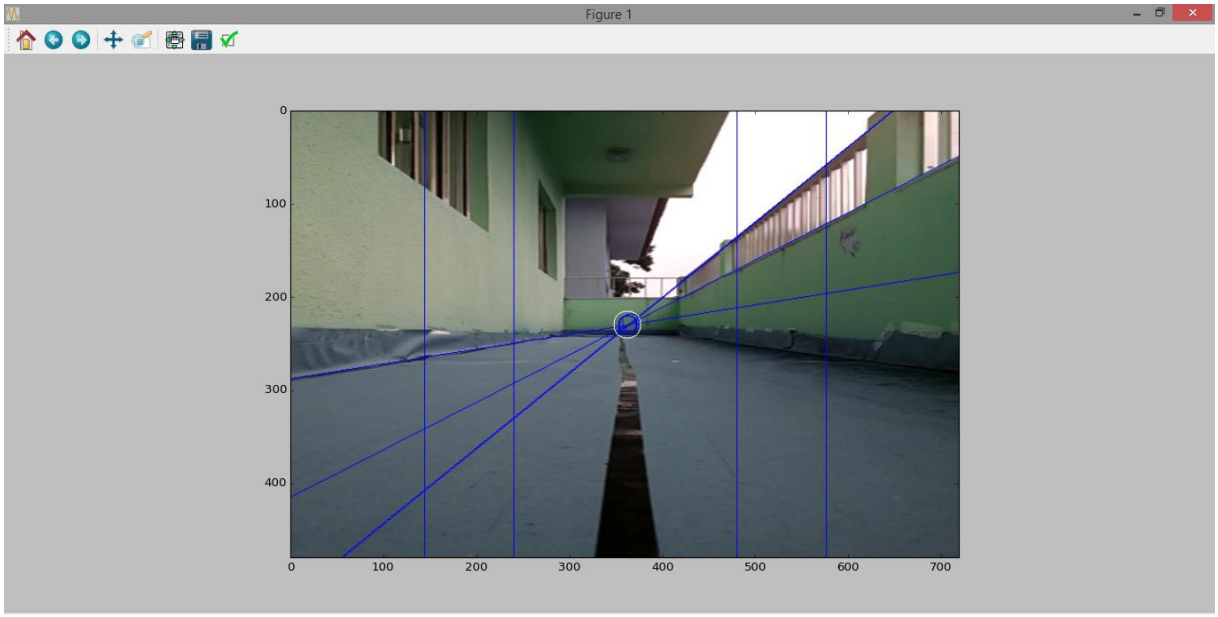


Figura 17: Software desenvolvido para realização da análise off-line

## 4.2 CONFIGURAÇÃO DOS PARÂMETROS

Para a calibração dos parâmetros utilizados na etapa de detecção de bordas, execução da Transformada de Hough, tempo de acionamento dos motores, e parâmetros de decisão, foram executados testes no Corredor 01, especificado na seção anterior. Foram realizados vários experimentos variando o valor dos parâmetros até que se obteve uma configuração satisfatória de toda solução desenvolvida.

Os parâmetros do sistema foram ajustados de forma sequencial, de acordo com as etapas da metodologia. As subseções a seguir mostram como foi realizada a calibração em cada uma destas etapas.

### 4.2.1 Ajustes na detecção de bordas

Primeiro, foi ajustado o valor do método de detecção de bordas Canny até que a filtragem dos ruídos e das bordas de cada frame fosse considerada razoável. A Tabela 3 mostra um resumo dos parâmetros e de seus respectivos valores utilizados na solução deste trabalho.

Tabela 3: Parâmetros para o método Canny

| Parâmetro | Descrição | Valor |
|---|---|---|
| **CANNY_LOW** | Determina um limite inferior pro valor do gradiente do *pixel*. Valores abaixo do definido neste parâmetro são descartados. | 150 |
| **CANNY_HIGH** | Se o gradiente do pixel é maior que o definido neste parâmetro, o pixel é aceito como uma borda. | 200 |
| **CANNY_KERNEL** | Tamanho do núcleo de Sobel utilizado | 3 |
| **CANNY_GRADIENT** | Valor que determina o uso da Norma para cálculo de gradiente. | False |

### 4.2.2 Ajustes na detecção de semirretas

Após o ajuste da detecção de bordas, foram ajustados os parâmetros da Transformada de Hough probabilística, que precisa, entre outros parâmetros, de uma imagem (ou frame) binarizada, calculada neste trabalho com o método Canny. As informações relativas aos parâmetros da THP são mostrados na Tabela 4.

Tabela 4: Parâmetros para a Transformada de Hough Probabilística

| Parâmetro | Descrição | Valor |
|---|---|---|
| **HOUGH_RHO** | Resolução da distância do parâmetro Rho da THP | 1 |
| **Hough_theta** | Resolução do ângulo parâmetro Theta usado na THP | π/180 |
| **HOUGH_THRESHOLD** | Limiar usado para definir quais linhas serão retornadas. Só linhas com mais votos que o valor estabelecido são retornadas pelo método | 140 |
| **HOUGH_MINLINELENGTH** | Tamanho mínimo que uma linha deve ter | 75 |
| **HOUGH_MAXLINEGAP** | Diferença máxima entre dois pontos para permitir uma ligação entre eles | 40 |

### 4.2.3 Ajustes de tempo

Na seção 3.1.3, foram definidas cinco regiões que estabelecem cinco decisões diferentes para a DTBPF. Para os casos onde é necessário realizar uma curva para a correção da trajetória, é necessário estabelecer um tempo de acionamento dos motores. Como existe quatro decisões para realizar curvas, com dois graus de intensidade distintos (curva ou curva suave), é necessário estabelecer dois tempos de atuação, um para as decisões CE e CD e outro para as decisões CSD e CSE. Após várias experimentações, os valores da Tabela 5 mostraram-se satisfatórios para a metodologia.

Tabela 5: Parâmetros de tempo

| Parâmetro | Descrição | Valor |
|---|---|---|
| **DELAY_INSTRUCAO** | Tempo de espera estipulado entre duas instruções | 5000 |
| **SLEEP_INSTRUCAO_BIG** | Usado para definir o tempo de acionamento do motor para as decisões de CD e CE | 220 |
| **SLEEP_INSTRUCAO_SMALL** | Usado para definir o tempo de acionamento do motor para as decisões de CSD e CSE | 150 |
| **GAP_FIM_NAVEGACAO** | Usado para finalizar a navegação automaticamente | 10000 |

### 4.2.4 Ajustes de limiares e multiplicadores

Foram inseridos alguns limiares para restringir o processamento de algumas etapas, assim como evitar problemas de erros em cálculos matemáticos (p. ex. divisões por zero). A descrição de cada limiar e multiplicador, assim como seus valores, são especificados na Tabela 6 e Tabela 7,respectivamente.

Tabela 6: Parâmetros limiares

| Parâmetro | Descrição | Valor |
|---|---|---|
| **TOLERANCIA_PLOTALINHA** | Usado para evitar que sejam inseridas semirretas horizontais ou verticais para o cálculo do ponto de fuga | 30 |
| **DIFERENCA_EIXOS** | Usada como limiar para calcular a inclinação de retas | 0,5 |
| **MIN_LINHAS_DECISAO** | Estabelece quantas linhas devem haver, no mínimo, para executar a heurística | 1 |

Tabela 7: Multiplicadores

| Parâmetro | Descrição | Valor |
|---|---|---|
| **VELOCIDADE** | Usado para determinar a velocidade que o robô usa para a locomoção, é um sinal analógico que varia de 0 a 255 | 100 |
| **ALFA** | Multiplicador usado para deslocar os pontos no plano cartesiano e permitir a plotagem das semirretas calculadas pela THP | 1000 |

## 4.3 CRITÉRIOS DE VALIDAÇÃO

Para aferir a qualidade da metodologia foi considerado um conjunto de métricas detalhadas nas seções seguintes.

### 4.3.1 Taxa de acerto de decisão (TaAD)

Este indicativo mostra um percentual de quanto o processo de orientação de trajetória acertou na tomada de decisão. O valor desta métrica é definido como a razão entre o número de decisões acertadas pelo sistema de navegação sobre o total de decisões tomadas pelo sistema de navegação durante a execução método. Três decisões válidas são consideradas: Vire à esquerda, vire à direita ou, siga em frente.

Para determinar se o sistema de orientação tomou uma decisão correta em um dado momento da navegação do protótipo robótico, foi realizada uma análise off-line dos frames persistidos durante o processo de navegação. Assim, foi comparado se, no momento em que foi tomada uma decisão, o algoritmo decide pela ação que seria tomada por um humano no momento analisado. O valor possível para cada decisão tomada é Falso, caso a decisão do algoritmo seja diferente em relação a uma decisão humana em um dado momento, ou, Verdade, caso a decisão seja a mesma que um humano tomaria no quadro analisado. Matematicamente, o cálculo é realizado conforme a Fórmula 8.

$$TaAD = \frac{Quantidade\ de\ decisões\ válidas}{Total\ de\ decisões\ tomadas} \times 100 \qquad (8)$$

### 4.3.2 Diferença Horizontal (DH)

Medida que mostra o quanto um ponto de fuga calculado pelo método difere do ponto de fuga ideal. Um ponto de fuga ideal é determinado pela análise off-line. Para cada frame onde é possível determinar um ponto de fuga através da percepção humana, é determinado um ponto ideal de fuga.

A DH é a diferença entre o valor do eixo horizontal do ponto de fuga calculado pela técnica e o ponto de fuga ideal. Neste trabalho, a diferença vertical não foi considerada, pois a DTBPF considera somente o deslocamento horizontal para

determinar a orientação da trajetória. Assim, a DH é uma medida porcentual, definida na Fórmula 9.

$$DH = \frac{|x_{método} - x_{ideal}|}{comprimento\ do\ quadro} \times 100 \qquad (9)$$

Onde os seguintes dados são utilizados para o cálculo da métrica:

- $x_{método}$ = valor do ponto de fuga calculado pelo método no eixo horizontal.
- $x_{ideal}$ = valor do ponto de fuga ideal determinado na análise *off-line*
- comprimento do *frame* = comprimento total, em *pixels*, do *frame* analisado

## 4.4 ANÁLISE DOS RESULTADOS

Durante os testes foram persistidos todos os quadros capturados pela câmera, com as linhas calculadas pela THP, interseções, ponto de fuga, e limites das regiões de navegação sobrescritos para a realização da análise *off-line*.

### 4.4.1 Experimento 01

Neste experimento há uma porta colorida no final do corredor com várias figuras desenhadas que interferem no resultado do algoritmo. Pois, quando o protótipo robótico se aproxima do fim do corredor os desenhos da porta formam retas que se interceptam. Assim, os desenhos contidos na porta acabam influenciando decisão do algoritmo de trajetória.

O desempenho da estratégia de controle de trajetória, para o Corredor 1, é analisado com base no gráfico da Figura 18 e na Tabela 8

O gráfico em questão apresenta uma comparação da detecção do ponto de fuga realizada pelo sistema com a detecção manual (*ground-truth*) ao longo dos quadros. Considerou-se na análise apenas a coordenada horizontal dos pontos (eixo y do gráfico) e os quadros para os quais o sistema conseguiu detectar pontos de fuga dentro dos limites da imagem (eixo x do gráfico). Estão representados no gráfico, também, os limites de decisão esquerdo e direito.

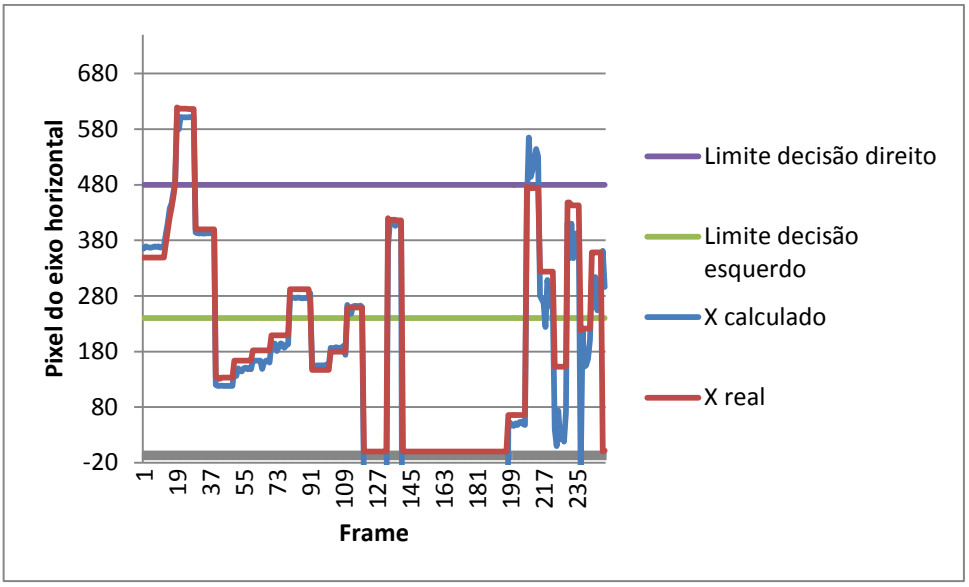


Figura 18: Execução do experimento no Corredor 1

De um modo geral, a curva que representa o desempenho do sistema proposto tende a acompanhar a curva que representa a detecção manual. No quadro 19, o limite de decisão direito foi ultrapassado. Esse fato indica que o robô está orientado para o outro lado. A Figura 19 mostra a sequência de quadros entre o ponto de fuga calculado entre o quadro 18 e o quadro 19, onde é possível ver a extrapolação do limite da decisão SF para a decisão CD.

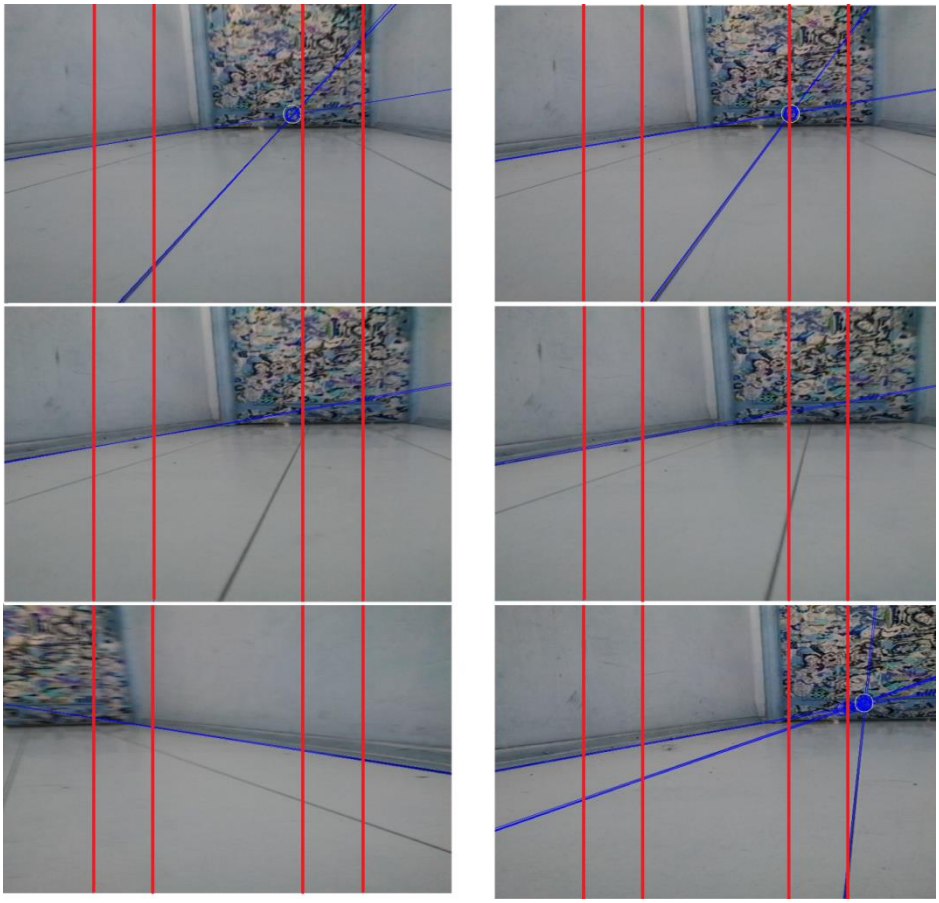

Figura 19: Sequência capturada entre o quadro 18 e o quadro 19 relativo ao gráfico da Figura 18

O ponto de fuga detectado pelo algoritmo é próxima da detecção do ponto de fuga ideal, porém, com o passar o protótipo robótica se aproxima do fim do corredor e a porta  no final do percurso faz com que o algoritmo insira interseções que influenciam no cálculo do ponto de fuga.

Em resumo, a mostra os resultados obtidos pelo método no Corredor 1. Neste experimento o fato da porta no fim do corredor influenciar na detecção do ponto de fuga faz com que a diferença horizontal relativa média tenha uma elevação no seu valor, que se acentua ao passo que o protótipo robótico se aproxima do final do percurso.

Tabela 8: Resumo da execução da metodologia no Corredor 1

| Execução | Tempo de navegação (s) | Distância percorrida | Quantidade de decisões | Taxa de acerto de decisão | Diferença horizontal relativa média |
|---|---|---|---|---|---|
| **1** | 163 | 100% | 24 | 95,83% | 13,40% |
| **2** | 135 | 99% | 22 | 81,92% | 9,03% |
| **3** | 122 | 100% | 16 | 81,25% | 4,03% |
| **4** | 30 | 58% | 5 | 60,00% | 4,58% |
| **5** | 156 | 90% | 37 | 72,97% | 3,91% |
| **6** | 213 | 100% | 38 | 83,78% | 7,61% |
| **7** | 173 | 100% | 36 | 81,82% | 4,47% |
| **8** | 87 | 90% | 30 | 70,00% | 8,73% |
| **9** | 73 | 100% | 10 | 80,00% | 7,41% |
| **10** | 158 | 98% | 28 | 92,86% | 3,98% |
| **Média:** | | **94%** | **25** | **80,04%** | **6,72%** |

### 4.4.2 Experimento 02

No experimento 2 o comportamento do ponto de fuga real é semelhante ao comportamento do ponto de fuga ideal (**Erro! Fonte de referência não encontrada.**). Isto mostra que o método é capaz de detectar a trajetória de forma satisfatória por conta da implementação dos limiares para a detecção de retas explicitadas na seção 4.1 que excluem semirretas verticais e semirretas horizontais do processo de detecção de interseções, mesmo em corredores que possuem alguns obstáculos.

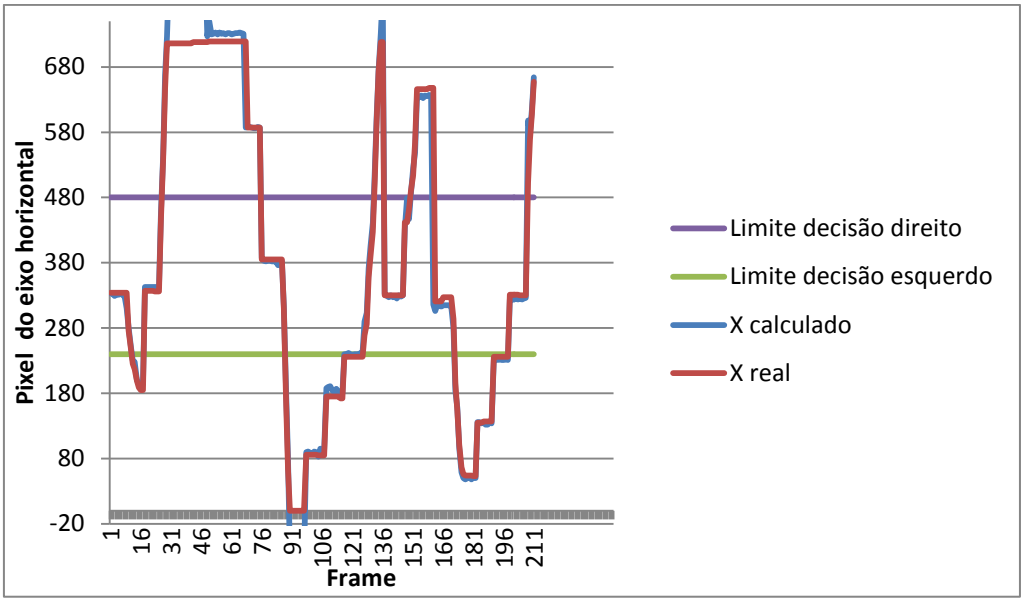


Figura 20: Execução do experimento do Corredor 2

Os resultados obtidos no Corredor 2 são apresentados na Tabela 9 Pode-se observar que a taxa de acerto de decisão possui bons resultados, com execuções que atingiram 100% de acertos nas decisões tomadas pela metodologia.

Em contrapartida, a distância percorrida foi, em média, de 53% do percurso. O problema da distância percorrida se deu por dois motivos que aconteceram em execuções distintas. O primeiro motivo é a perca arbitrária da conexão *bluetooth* entre o *smartphone* e o módulo *bluetooth* do Arduino que em alguns casos reporta uma de tempo de espera (*timeout*) para a conexão. O segundo motivo é relacionado à variação da iluminação por conta das janelas contidas no corredor e da incidência dos raios de sol de forma variável (p. ex. quando o tempo está claro e, tempo depois uma nuvem encobre o sol). Neste caso, a variação de iluminação interfere no ajuste do foco da câmera que é feito de forma automática através do Sistema Operacional do *smartphone*. Com a perda de foco, o método não consegue estabelecer bordas na imagem e, assim, o aplicativo de desenvolvido é encerrado por falta de detecção de pontos de fuga.

Tabela 9: Resumo da execução da metodologia no Corredor 2

| Execução | Tempo de navegação (s) | Distância percorrida | Quantidade de decisões | Taxa de acerto de decisão | Diferença horizontal relativa média |
|---|---|---|---|---|---|
| **1** | 301 | 61% | 37 | 100,00% | 1,10% |
| **2** | 172 | 40% | 25 | 96,00% | 1,84% |
| **3** | 59 | 32% | 7 | 100,00% | 1,03% |
| **4** | 338 | 72% | 35 | 97,14% | 4,49% |
| **5** | 151 | 52% | 20 | 100,00% | 0,87% |
| **6** | 143 | 53% | 18 | 94,44% | 1,82% |
| **7** | 147 | 34% | 19 | 89,47% | 1,72% |
| **8** | 110 | 69% | 12 | 92,31% | 0,63% |
| **9** | 160 | 65% | 18 | 94,74% | 0,97% |
| **10** | 100 | 50% | 12 | 84,62% | 2,02% |
| Média: | | **53%** | **20** | **94,87%** | **1,65%** |

# 5 CONCLUSÃO

Este trabalho mostrou uma metodologia que utiliza um controle de trajetória baseada em ponto de fuga para a navegação de protótipos robóticos. Os resultados apresentados mostram que, com uma boa parametrização dos métodos necessários para realizar a interpretação geométrica dos quadros captados pela câmera, *smartphones* podem ser usados para solucionar problemas que envolvam detectar trajetórias em ambientes internos e assistir sistemas ou indivíduos nos casos onde é necessário ter conhecimento deste tipo de informação.

Neste trabalho foi possível identificar várias necessidades e oportunidades para trabalhos futuros. As principais percepções sobre o que pode ser desenvolvido podem ser listados da seguinte forma:

- Implementar novas interfaces entre o OpenCV e o Android para facilitar a manipulação de características e funcionalidades de smartphones;
- Realizar o cálculo de pontos de fuga com a utilização de métodos de otimização;
- Realizar partes das etapas de PDI com alguma linguagem mais rápida que a linguagem nativa do Android, como, por exemplo, C++;
- Desenvolver no sistema de navegação o reconhecimento de objetos;
- Desenvolver no método um tratamento da posição no eixo vertical para possibilitar o uso da metodologia em situações onde a altura é relevante, tais como na navegação de veículos aéreos não tripulados;
- Desenvolver métodos para calcular o ponto de fuga levando em consideração o histórico de pontos de fuga calculados anteriormente;
- Desenvolver estudos para propor heurística mais refinadas para calcular a trajetória nos casos onde não é possível calcular um ponto de fuga em um quadro.
- Melhorar a comunicação sem fio entre *smartphone*s com SO Android e o módulo *bluetooth* do Arduino

É possível concluir que a utilização da Transformada de Hough Probabilística em um quadro com bordas detectadas pelo método Canny é eficaz para extração de

padrões de reta, desde que os parâmetros para detecção de bordas e para extração de linhas estejam bem ajustados.

Também é possível concluir que *smartphones* suportam o processamento de imagens, ainda que com restrição de capacidade de processamento, quando utilizado a interface do OpenCV para tecnologia Android combinado com a linguagem nativa do *smartphone*.

# 6 REFERÊNCIAS


AMAZON. **Amazon Prime Air**. Amazon. 2014. Disponível em: <http://www.amazon.com/b?node=8037720011>. Acesso em: 14/12/2014.

ANDROID. **Get the Android SDK**. Android. 2014. Disponível em: <http://developer.android.com/sdk/index.html>. Acessado em: 10 fev. 2014.

ARDUINO. **Arduino – HomePage**. Arduino. 2014. Disponível em: <http://arduino.cc/>. Acessado em: 09 fev. 2014.

ASAD, Muhammad; IKRAM, Waseem. Smartphone based guidance system for visually impaired person. In: **Image Processing Theory, Tools and Applications (IPTA), 2012 3rd International Conference on**. IEEE, p. 442-447, 2012.

BARNARD, Stephen T. Interpreting perspective images. **Artificial intelligence**, v. 21, n. 4, p. 435-462, 1983.

BAUMGARTNER, Eric T.; SKAAR, Steven B. An autonomous vision-based mobile robot. **Automatic Control, IEEE Transactions on**, v. 39, n. 3, p. 493-502, 1994.

BILLS, Cooper; CHEN, Joyce; SAXENA, Ashutosh. Autonomous MAV flight in indoor environments using single image perspective cues. In: **Robotics and automation (ICRA), 2011 IEEE international conference on**. IEEE, p. 5776-5783, 2011.

BLUETOOTH. **A Look at the Basics of Bluetooth Technology**. Bluetooth. 2014. Disponível em: <http://www.bluetooth.com/Pages/Basics.aspx>. Acesso em: 09 fev. 2014.

CANTONI, Virginio et al. Vanishing point detection: representation analysis and new approaches. In: **Image Analysis and Processing, 2001. Proceedings. 11th International Conference on**. IEEE, p. 90-94, 2001.

COLLINS, Robert T.; WEISS, Richard S. Vanishing point calculation as a statistical inference on the unit sphere. In: **Computer Vision, 1990. Proceedings, Third International Conference on**. IEEE, 1990. p. 400-403.

DE CRISTÓFORIS, Pablo et al. Real-time monocular image-based path detection. **Journal of Real-Time Image Processing**, 2013. p. 1-14.

DESOUZA, Guilherme N.; KAK, Avinash C. Vision for mobile robot navigation: A survey. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 24, n. 2, p. 237-267, 2002.

DEVELOPER, Android. **Android Developer Tools**. Android. 2014. Disponível em: <http://developer.android.com/tools/help/adt.html>. Acessado em: 17 de dezembro de 2014.

EBRAHIMPOUR, R. et al. Vanishing point detection in corridors: using Hough transform and K-means clustering. **Computer Vision, IET**, v. 6, n. 1, p. 40-51, 2012

FARAGASSO, Angela et al. Vision-based corridor navigation for humanoid robots. In: **Robotics and Automation (ICRA), 2013 IEEE International Conference on**. IEEE, p. 3190-3195, 2013.

FILIPEFLOP, Blog. **Motor DC com Driver Ponte H L298N**. Filipeflop. 2014. Disponível em: <http://blog.filipeflop.com/motores-e-servos/motor-dc-arduino-ponte-h-l298n.html>. Acessado em: 15 de dezembro de 2014.

GROSS, Doug. Amazon's drone delivery: How would it work?. **CNN Tech**, 2 de dezembro de 2013,. Disponível em: <http://www.cnn.com/2013/12/02/tech/innovation/amazon-drones-questions/>. Acesso em: 14/12/2014.

HONDA, Flavio. **Motores de Corrente Contínua. Guia Rápido para uma Especificação Precisa**. Siemens. 2006. Disponível em: <http://www.mariloureiro.net/tecnica/electrif/Motores_CC_ind1.pdf>. Acessado em: 15 de dezembro de 2006.

HOTA, Rudra N. et al. A Simple and Efficient Lane Detection using Clustering and Weighted Regression. In: **COMAD**. 2009.

JAVA. **O que é a tecnologia Java e por que é necessária?**. Java. 2014. Disponível em: <http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acessado em: 10 fev. 2014

KAISER, M. K.; GANS, N. R.; DIXON, W. E. Vision-based estimation for guidance, navigation, and control of an aerial vehicle. **Aerospace and Electronic Systems, IEEE Transactions on**, v. 46, n. 3, p. 1064-1077, 2010.

LIOU, Shih-Ping; JAIN, Ramesh C. Road following using vanishing points. **Computer vision, graphics, and image processing**, v. 39, n. 1, p. 116-130, 1987.

MAGEE, Michael J.; AGGARWAL, Jake K. Determining vanishing points from perspective images. **Computer Vision, Graphics, and Image Processing**, v. 26, n. 2, p. 256-267, 1984.

MURPHY, Robin. **Introduction to AI robotics**. The MIT press, 2000. p. 6-10.

OPENCV. **About**. Opencv. 2014. Disponível em: <http://opencv.org/about.html>. Acessado em: 10 fev. 2014.

PLAY, Google. **Google Play**. Google. 2014. Disponível em: <https://play.google.com/store>. Acessado em: 14 de dezembro de 2014.

QUAN, Long; MOHR, Roger. Determining perspective structures using hierarchical Hough transform. **Pattern Recognition Letters**, v. 9, n. 4, p. 279-286, 1989.

REN, Zhiquan; WANG, Chun; HE, Jibo. Vehicle detection using Android smartphones. In: **7th International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design. Bolton Landing, New York**. 2013.

SANTOSH, D.; ACHAR, Supreeth; JAWAHAR, C. V. Autonomous image-based exploration for mobile robot navigation. In: **Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on**. IEEE, 2008. p. 2717-2722.

SEO, Kyung-Seok; LEE, Jung-Hwa; CHOI, Heung-Moon. An efficient detection of vanishing points using inverted coordinates image space. **Pattern Recognition Letters**, v. 27, n. 2, p. 102-108, 2006.

SHI, Wenxia; SAMARABANDU, Jagath. Corridor line detection for vision based indoor robot navigation. In: **Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on**. IEEE, p. 1988-1991, 2006.

SHIGANG, Li; TSUJI, Saburo; IMAI, Masakazu. Determining of camera rotation from vanishing points of lines on horizontal planes. In: **Computer Vision, 1990. Proceedings, Third International Conference on**. IEEE, 1990. p. 499-502.

SINGHAL, Amit. Issues in autonomous mobile robot navigation. **Computer Science Dept, U. of Rochester**, 1997.

TAI, A. et al. Vanishing point detection. **Image and Vision Computing**, v. 11, n. 4, p. 240-245, 1993.

TUYTELAARS, Tinne et al. The cascaded Hough transform as an aid in aerial image interpretation. In: **Computer Vision, 1998. Sixth International Conference on**. IEEE, p. 67-72, 1998.

UNO, Arduino. **Arduino – ArduinoBoardUno**. Arduino. 2014. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>. Acessado em: 14/12/2014

WANG, Ling-Ling; TSAI, Wen-Hsiang. Computing camera parameters using vanishing-line information from a rectangular parallelepiped. **Machine Vision and Applications**, v. 3, n. 3, p. 129-141, 1990.

WU, Gang et al. Walking Goal Line Detection Based on Machine Vision on Harvesting Robot. In: **Circuits, Communications and System (PACCS), 2011 Third Pacific-Asia Conference on**. IEEE, 2011. p. 1-4.