UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTIMATIVA DE ORIENTAÇÃO COM UMA BÚSSOLA VISUAL BASEADA EM CORES

DISSERTAÇÃO DE MESTRADO

Fabrício Tonetto Londero

Santa Maria, RS, Brasil

2016

ESTIMATIVA DE ORIENTAÇÃO COM UMA BÚSSOLA VISUAL BASEADA EM CORES

Fabrício Tonetto Londero

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof. Dr. Giovani Rubert Librelotto

Santa Maria, RS, Brasil

2016

Londero, Fabrício Tonetto

Estimativa de Orientação com uma Bússola Visual Baseada em Cores / por Fabrício Tonetto Londero. – 2016.
80 f.: il.; 30 cm.

Orientador: Giovani Rubert Librelotto
Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação, RS, 2016.

1. Bússola Visual. 2. Orientação Visual. 3. Orientação baseada em cores. I. Librelotto, Giovani Rubert. II. Título.

E-mail: fabriciotonettolondero@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

ESTIMATIVA DE ORIENTAÇÃO COM UMA BÚSSOLA VISUAL BASEADA EM CORES

elaborada por
Fabrício Tonetto Londero

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Giovani Rubert Librelotto, Dr.
(Presidente/Orientador)

Eduardo Kessler Piveta, Dr. (UFSM)

Reiner Franthesco Perozzo, Dr. (UNIFRA)

Santa Maria, 21 de Dezembro de 2016.

À minha família

AGRADECIMENTOS

Agradeço ao meu orientador, professor Giovani, que sempre esteve presente para me auxiliar na execução da pesquisa.

Agradecer ao professor Rodrigo da Silva Guerra, peça importante para o desenvolvimento deste trabalho.

E também à minha família e aos amigos pelo apoio e compreensão, assim como a Equipe Taura Bots.

"Se você não tiver seu próprio plano de vida, é provável que caia no plano de alguma outra pessoa. E adivinha o que eles planejaram para você? Não muito."
Jim Rohn

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

ESTIMATIVA DE ORIENTAÇÃO COM UMA BÚSSOLA VISUAL BASEADA EM CORES

AUTOR: FABRÍCIO TONETTO LONDERO
ORIENTADOR: GIOVANI RUBERT LIBRELOTTO
Local da Defesa e Data: Santa Maria, 21 de Dezembro de 2016.

Os seres humanos e animais fazem uso de muitos sentidos para interagir com o ambiente a sua volta. Os computadores, por sua vez, interagem por meio de dispositivos de entrada e saída, tais como caixas de som, teclados e monitores, tendo suas interações muito mais limitações. Mas, atualmente, dispositivos como câmeras e microfones foram adicionado à computadores, aumentando sua interatividade. Com este avanço, surgem robôs autônomos equipados com sensores, tais como de som, de tato e visão, este último por intermédio de uma ou de mais câmeras. Um robô, para ser considerado autônomo, deve tomar decisões sem a intervenção humana e, para possuir excelência no que faz, deve ser munido de uma boa forma de orientação. Este trabalho apresenta uma alternativa de bússola visual para estipular a orientação de robôs e veículos autônomos, para que estes consigam se locomover em um cenário (ambiente) e efetuar o trabalho no qual foram construídos para desempenhar. A abordagem proposta trabalha com perspectivas 360º do ambientes, no qual se extrai informações das trocas de cores que ocorrem. Com as contagens de trocas de cores, o processo é repetido com imagens recebidas dos robôs ou veículos em movimento, e as trocas de cores desta imagem com as armazenadas da perspectiva 360º. O resultado da comparação de troca de cores é utilizado para estimar o grau de similaridade entre as imagens e assim, apresentar o ângulo no qual a imagem do robô em movimentação está presente na perspectiva 360º. Os robôs ou veículos autônomos podem usar deste valor apresentado nas suas tomadas de decisão.

Palavras-chave: Bússola Visual. Orientação Visual. Orientação baseada em cores.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Computer Science
Federal University of Santa Maria

ESTIMATES GUIDANCE WITH A VISUAL COMPASS BASED IN COLOR

AUTHOR: FABRÍCIO TONETTO LONDERO
ADVISOR: GIOVANI RUBERT LIBRELOTTO

Defense Place and Date: Santa Maria, December 21th, 2016.

Humans and animals make use of many senses to interact with the environment around them. Computers, in turn, interact through input and output devices such as speakers, keyboards, and monitors, and their interactions have many more limitations. But currently, devices like cameras and microphones have been added to computers, increasing their interactivity. With this advancement, there are autonomous robots equipped with sensors, such as sound, touch and vision, the latter through one or more cameras. A robot, to be considered autonomous, must make decisions without human intervention and, in order to have excellence in what it does, must be provided with a good form of guidance. This work presents an alternative visual compass to stipulate the orientation of robots and autonomous vehicles, so that they can get around in a scenario (environment) and do the work in which they were built to perform. The proposed approach works with 360º perspective of environments, in which information is extracted from the color changes that occur. With color change counts, the process is repeated with images received from the robots or vehicles in motion, and the color changes of this image with those stored from the 360º perspective. The result of the color change comparison is used to estimate the degree of similarity between the images and thus to present the angle at which the image of the moving robot is present in the 360º perspective. Robots or autonomous vehicles can use this value presented in their decision making.

Keywords: Visual compass. Visual orientation. Color-based Guidance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE APÊNDICES

LISTA DE ABREVIATURAS E SIGLAS

3D	Espaço tridimensional
BPMN	*Business Process Model and Notation*
CBR	Competição Brasileira de Robótica
CPU	*Central Processing Unit*
FIFA	Federação Internacional de Futebol (*Fédération Internationale de Football Association*)
HSI	*hue (matiz), saturation (saturação) and intensity (intensidade)*
HSL	*hue (matiz), saturation (saturação) and lightness (luminosidade)*
HSV	*hue (matiz), saturation (saturação) and value (valor)*
IA	Inteligência Artificial
OPENCV	*Open Source Computer Vision Library*
PIXEL	*Picture Element*
RGB	Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*)
SLAM	Localização e Mapeamento Simultâneos
TAURA	Tecnologia em Automação e Robótica Aplicada

SUMÁRIO

1 INTRODUÇÃO

Segundo Kenneth Dawson-Howe (2014), a percepção é essencial para que qualquer entidade interaja de forma significativa com o seu ambiente. Os seres humanos usam muitos sentidos (tais como visão, audição, tato e olfato) para perceber o mundo. Por sua vez, os computadores só podiam receber uma entrada através de dispositivos de entrada simples, tais como teclados e mouses, ou por meio de canais de comunicação com e sem fio. No entanto, nos últimos anos, câmeras e microfones foram adicionados como peças padrão de computadores e dispositivos móveis, aumentando seus dispositivos de interação, de amplificação de seus sentidos com o ambiente.

A visão computacional possibilita automatizar imagem ou vídeo e ainda ampliar o nosso entendimento sobre máquinas. Kenneth ainda afirma que as técnicas utilizadas para automatizar tarefas que vão desde a inspeção e compreensão de vídeo possibilitam desenvolver robôs autônomos para que eles possam melhorar sua interação com ambientes projetado para seres humanos.

Um dos desafios da computação é fazer com que veículos ou robôs autônomos possam se orientar dentro de um contexto (ambiente). Um exemplo disso é a RoboCup, principal competição de futebol de robôs do mundo, tem como objetivo ampliar as dificuldades a cada ano, para que, em 2050, haja um time de robôs humanoides apto a jogar contra os humanos vencedores da Copa do Mundo FIFA de futebol (ROBOCUP LEAGUE TECHNICAL COMMITTEE, 2015). Por exemplo, nos últimos anos, as traves passaram a não possuir mais distinção e os lados do campo são simétricos (ROBOCUP HUMANOID LEAGUE TECHNICAL COMMITTEE, 2015), as equipes necessitam elaborar uma nova forma para que os robôs se orientem dentro do campo e, principalmente, não marquem gols contra.

Baseado no que foi apresentado e em uma carência de implementação acerca da orientação visual para a equipe Taura Bots (UFSM) de futebol de robôs, surgiu o interesse no estudo e elaboração deste trabalho sobre bússola visual para a orientação de veículos e robôs autônomos. Foi elaborado um levantamento sobre os trabalhos relacionados ao tema de orientação baseada em visão computacional, com o intuito de dar continuidade aos trabalhos existentes, evitando repetir os mesmos erros e corrigir problemas deixados em aberto.

Para o futebol de robôs, somente a câmera pode ser utilizada para estipular a orientação, sendo este o problema inicial deste trabalho. Constatou-se que uma bússola visual possui uma

gama maior de aplicabilidade, ou seja, não se limitando ao futebol de robôs. Existem outras alternativas para estipular a orientação, normalmente baseado em odometria, que acumula erros com o tempo, sendo esta uma solução confiável em um curto período de tempo. A bússola visual surge como uma alternativa para aplicações que fazem uso de odometria, para aumentar o grau de certeza quanto a uma orientação e/ou substituir a odometria após grande acumulo de erros (STURM; VISSER, 2009).

A abordagem de bússola visual proposta neste trabalho tem como objetivo, ser livre de contexto e de marcações feitas nos ambientes no qual os veículos ou robôs serão inseridos, diferente do apresentado nos trabalhos relacionados, sendo necessário apenas efetuar uma calibração prévia do cenário, que consiste em montar uma perspectiva 360º do ambiente. As imagens extraídas passam por algumas etapas de processamento de imagens, no qual tornam as imagens mais homogêneas, livre de ruídos. Dentre estas etapas, vale destacar a erosão e a dilatação, combinadas em uma transformação morfológica de abertura, seguida de uma redução de cores, resultando em uma imagem com apenas oito cores. Na sequência, é feita uma extração das áreas de interesse. Destas áreas de interesses, colunas de aproximadamente 10 pixeis de largura, se faz uma contagem das trocas de cores e com estas informações, se monta uma matriz quadrada de troca de cores.

Esses passos são reexecutados nas imagens provenientes dos veículos ou robôs autônomos, também resultando em matrizes quadradas e comparadas com as matrizes da perspectiva 360º do ambiente. Essa comparação entre as matrizes é efetuada com o cálculo de similaridade de cosseno, que retorna um valor entre 0 e 1, onde quando mais próximo a 1, mais similares são as matrizes comparadas. Com base no que foi apresentado, estipula-se a orientação dos veículos ou robôs autônomos no ambiente.

No Capítulo 2 deste trabalho, são apresentados conceitos fundamentais para a leitura deste trabalho, de maneira objetiva, tais como: processamento de imagens, visão computacional e cálculo de similaridade, e subcapítulos que apresentem teorias acerca das técnicas de processamento de imagens. O Capitulo 3 é composto pela metodologia empregada no desenvolvimento deste protótipo de bússola visual. Os testes efetuados para apresentar os pontos fracos e fortes da abordagem aparecem no Capítulo 4, seguido pela discussão dos resultados no Capítulo 5. No Capítulo 6, sçao trazidos os trabalhos relacionados ao tema de estimativa de orientação baseado em bússola visual. O Capítulo 7 aborda as considerações finais sobre o tema, relatando desafios, dificuldades e trabalhos futuros.

1.1 Objetivos Gerais

Desenvolver uma abordagem de bússola visual capaz de estipular a orientação dentro do cenário empregado, para ser utilizada em veículos re robôs autônomos.

1.2 Objetivos Específicos

- Funcionar de maneira genérica, livre de contexto;
- Funcionar sem a utilização de marcações no cenário;
- Funcionar fazendo uso apenas de uma ou mais câmeras.

2 ROBÓTICA

A robótica é uma das principais aplicações no qual uma bússola visual é necessária; em grande destaque às competições de futebol de robôs, no qual é o foco da grande maioria dos trabalhos correlatos. Este capítulo aborda a robótica e a Robocup, conceitos que culminaram com a motivação deste trabalho; assim como a Equipe Taura Bots.

Segundo Ullrich (1987), a robótica consiste no estudo de montagem e programação de robôs, e é considerada uma área de estudo relativamente recente, na qual é constituída de sistemas compostos por máquinas automáticas e controladas por circuitos integrados programáveis. As indústrias são grandes motivadoras e beneficiadas nos avanços tecnológicos nas áreas de eletrônica, mecânica e computação; na qual a construção de robôs cada vez mais avançados se faz possível (ROMANO, 2002). Para Ullrich (1987), a robótica pode ser definida como a ciência dos sistemas que interagem com o mundo real com ou sem intervenção humana e um robô é considerado um equipamento multifuncional e reprogramável, projetado para movimentar peças e materiais, além de efetuar outras ações especializadas tais que podem promover bem-estar, conforto, saúde e educação aos seres humanos.

Um destaque recente da robótica é a sua utilização como ferramenta educacional interdisciplinar. SegundoSantos, Nascimento e Bezerra (2010), a robótica faz com que alunos se tornem mais questionadores e capazes de efetuar relacionamentos entre diferentes tipos de conhecimentos de forma a melhorar a tomada de decisão e a solução de problemas; fazendo assim, com que os alunos ultrapassem os limites atribuídos a cada disciplina individualmente.

A gama de aplicabilidade de robôs é muito extensa. Além do que já foi abordado, existem robôs fundamentais na área da saúde (VALERO et al., 2011), onde robôs auxiliam médicos em cirurgias; na segurança no qual existem robôs que desarmam bombas, não colocando a vida de seres humanos em risco e outros utilizados como batedores que auxiliam no reconhecimento de território inimigo (ROMANO, 2002).

Dentre os tipos de robôs existentes, existem os robôs humanoides, que são os robôs que lembram seres humanos, ou seja, possuem duas pernas, dois braços, tronco e cabeça. Estes robôs se destacam por superar problemas de locomoção sofridos por robôs sobre rodas por exemplo; que possuem grande dificuldade em terrenos irregulares, acidentados e escadas. Sendo este, um dos robôs mais complexos de se trabalhar (FUGALI; LIBRELOTTO, 2016). Segundo Akhtaruzzaman e Shafie (2010), grande parcela da complexidade de se trabalhar com

robôs humanoides ocorre devido a simulação de tornozelos, joelhos, quadril, ombros, cotovelos e principalmente mãos, o que envolve a utilização de inúmeros atuadores para controlar o grau de liberdade necessária. Os autores ainda afirmam que os robôs humanoides têm recebido grande carga de atenção nos últimos anos, existindo grande número de projetos em andamento, sendo fundamental sua aplicabilidade no futuro, tais como robôs *homecare*, que podem cuidar de idosos, crianças e doentes; assim como robôs que auxiliam nas atividades domésticas.

Dentre os principais modelos de robôs humanoides, destaca-se o DarWin-Op, acrônimo de *Dynamic Anthropomorphic Robot with Intelligence–Open Platform* (HA et al., 2011). O DarWin-Op é um robô de plataforma aberta, onde qualquer pessoa ou instituição pode fazer o *download* do projeto e construir seu próprio DarWin-Op, assim como efetuar as alterações no projeto que julgar interessante.

2.1 *RoboCup*

No ano de 1997, o computador *Deep Blue* venceu uma partida de xadrez de Garry Kasparov, campeão mundial de xadrez. Sendo este, um marco para a inteligência artificial, que assumia o xadrez como principal desafio desde 1950. A partir deste momento, novos desafio precisavam ser propostos, foi então que se deu a *RoboCup* (MONTENEGRO, 2015).

Com o intuito de acarretar evolução e popularização da robótica, foi criada a *RoboCup*, uma competição internacional de robótica dividida em várias categorias de futebol de robôs e robôs de resgate. Dentre os principais objetivos, está o incentivo às pesquisas na área da robótica, visão computacional e inteligência artificial. A escolha do futebol se deu por ser o esporte mais popular do mundo, sendo assim, mais chamativo ao público, sendo este de participantes, espectadores e patrocinadores. A *RoboCup* propõe como objetivo que, em 2050, os robôs humanoides vencedores da *RoboCup* sejam capazes de jogar contra a seleção vencedora da copa do mundo FIFA (KITANO et al., 1998). Este objetivo parece ser difícil de ser alcançado, mas a cada ano a comissão organizadora altera algumas das regras da competição a fim de aumentar o desafio, fazendo com que as equipes tenham que trabalhar em algo novo a cada edição da competição (MONTENEGRO, 2015).

Segundo Gerndt (2015) e Baltes (2014); o projeto, a construção e a programação de um robô humanoide autônomo atualmente é uma tarefa difícil, e ser capaz de jogar futebol é um grande problema, mas apresenta uma constante evolução. Os autores ainda informam que o projeto da *RoboCup* é realmente ambicioso, mas desde que foi anunciado e teve sua primeira

edição no ano de 2002, em Fukuoka, Japão; teve grande participação de equipes/universidades de todo o mundo.

A participação da UFSM na *RoboCup* se dá por parte da equipe Taura Bots, que participa de duas modalidades de futebol de robôs humanoides: a categoria *Kid-Size*. Esta categoria contempla robôs de 40 a 90 centímetros; e desde 2016, participa da modalidade *Teen-Size*, por sua vez, esta categoria contempla robôs de 90 a 120 centímetros (ANDERS et al., 2016).

3 VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGENS

Muito se discute acerca dos conceitos que separam a visão computacional do processamento de imagens, não sendo ainda bem definidos. Caberia ao processamento de imagens, efetuar os devidos procedimentos em uma imagem, para torná-la adequada para a sua real utilização e/ou gerar uma nova imagem. Por sua vez, a visão computacional, que também recebe uma imagem como entrada, gera uma saída que seria a interpretação desta entrada, sendo total ou parcial (MARENGONI; STRINGHINI, 2009); ou ainda, segundo Shapiro e Stockman (2001), a visão computacional possui como objetivo a tomada de decisão acerca de características e informações extraídas de imagens.

Para Gonzalez et al. (GONZALEZ; WOODS; EDDINS, 2004), a área de processamento de imagens tem como característica o extenso experimento de soluções em um problema, pois uma solução de processamento para uma imagem, pode não ser o adequado para outra imagem, por mais similares que elas sejam. Ou seja, é necessário grande esforço em aplicar diferentes níveis de ensaios e experimentos, para então chegar numa solução que se pode considerar aceitável.

A visão computacional pode ser definida, segundo Bradsky e Kaebler (2008), como a transformação de dados de uma câmera fotográfica ou de vídeo em uma decisão ou uma nova forma de representação, com o objetivo de atingir um objetivo prévio. Esta transformação pode ser uma conversão de uma imagem colorida para uma imagem em tons de cinza, como a remoção de movimento de uma câmera a partir de sequência de imagens. Os autores salientam da importância de informações prévias, tais como, a existência de alguma pessoa na cena, a distância entre a câmera e algum objeto especifico, se a câmera estava em movimentação, para facilitar a transformação anteriormente citada e a precisão da mesma. Embora o ser humano seja uma criatura visual, pensa-se que o trabalho com visão computacional possa ser algo simples, o que é um grande equívoco.

Na Figura 3.1 é possível notar a diferença entre processamento de imagens e visão computacional, onde é apresentada uma imagem original (esquerda) com problemas na iluminação que acarretam em dificuldade na identificação da placa do veículo, e na direita a imagem pós-processamento, onde se pode identificar com mais facilidade a placa do veículo em questão. Com a imagem de saída (pós-processada), se pode aplicar a visão computacional para extrair informações da imagem, como no caso, a placa do veículo (MARENGONI; STRINGHINI,

2009), uma típica aplicação de visão computacional.

Figura 3.1 – Imagem original (esquerda) e a pós-processada (direita), possibilitando à visão computacional de extrair informações (placa do veículo) (MARENGONI; STRINGHINI, 2009)

Fica assim, evidente a interdependência de um estudo comparado ao outro. E, atualmente, são raras as áreas da tecnologia que não façam uso de alguma forma de processamento de imagens. Quando se fala de visão computacional, este estudo ainda é bastante novo, contendo poucos estudos e muito a ser feito. Para demonstrar a complexidade do estudo, pode-se analisar a Figura 3.2, onde existe um veículo, mas a câmera de um computador "enxerga"apenas sua representação em números. No caso, a matriz apresentada é a representação da área de abrangência do espelho retrovisor do lado esquerdo do veículo. Para efetuar correções em uma imagem, deve-se manipular estes números, existindo fórmulas e práticas para efetuar diferentes tipos de alterações, correções, melhorias ou simplesmente preparação de uma imagem. Mesmo com este conhecimento, se torna formalmente impossível a criação de um código para identificar automaticamente, sem intervenção humana, espelhos retrovisores em veículos baseado na leitura e comparações dessas matrizes numéricas (BRADSKI; KAEHLER, 2008).

Para diminuir o grau de dificuldade quanto ao processamento de imagens e aumentar a compreensão básica, se deve tomar conhecimento quanto à procedência das imagens, tais como raio-x e ressonância magnética. Logo, a abrangência do estudo de processamento de imagens não é global, sendo desmembrado para uma melhor geração de resultados e eficiência nos estudos, diminuindo consideravelmente a complexidade por ser estudado de maneira mais específica (GONZALEZ; RICHARD, 2002).

Autores como Orlandini (2012), Gonzalez e Richard (2002), Bradski e Kaehler (2008) e Shapiro e Stockman (2001) destacam a dificuldade de se trabalhar com visão computacional, principalmente ao desenvolver algo genérico, podendo ser considerado algo utópico. Como pode ser visto na Figura 3.3, a porta da direita possui uma área mais luminosa, podendo ser

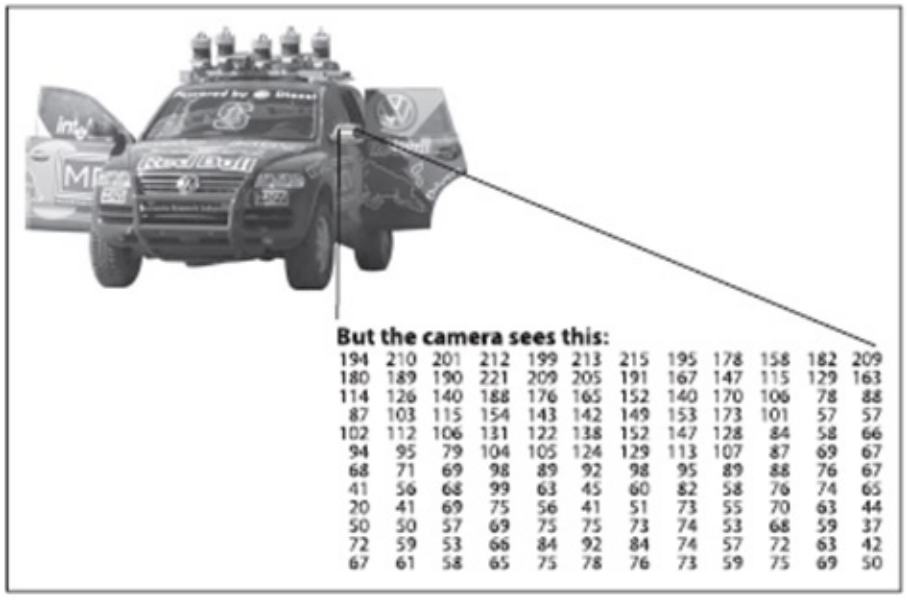


Figura 3.2 – Veículo e como um computador "enxerga"o espelho retrovisor (MARENGONI; STRINGHINI, 2009)

referente a abertura de uma janela do ambiente, ou um reflexo de um veículo que passou em uma rua próxima, mas ainda assim, podemos constatar se tratar de uma porta, e inclusive da mesma porta da esquerda da imagem; mas para um computador, a tarefa não é tão simples.

Figura 3.3 – Reflexo alterando um ambiente (ORLANDINI; MARTINS, 2012)

Devido à grande complexidade de se trabalhar com visão computacional, foram desenvolvidas bibliotecas para auxiliar nas tarefas, pensando em diminuir a dificuldade e tornar a área mais atrativa. Na seção seguinte, trata sobre a OpenCV, uma das mais populares ferramentas de processamento de imagens e visão computacional.

3.1 Imagens Digitais

Na computação, imagens digitais podem ser representadas por uma matriz ou função bidimensional *f(x,y)* onde x e y são coordenadas espaciais, como pode ser visto na Figura 3.2 e 3.4, onde cada posição, denominada Pixel (*Picture Element*), possui valores que representam a cor em si. A altura (quantidade de pixel de altura) e largura (quantidade de pixel de largura) da matriz depende da resolução onde quanto maior a resolução, maior a quantidade de pixels, como pode ser visto na Figura 3.5, onde a mesma imagem foi extraída com resoluções diferentes (SCURI, 1999) (SANTOS, 2012).

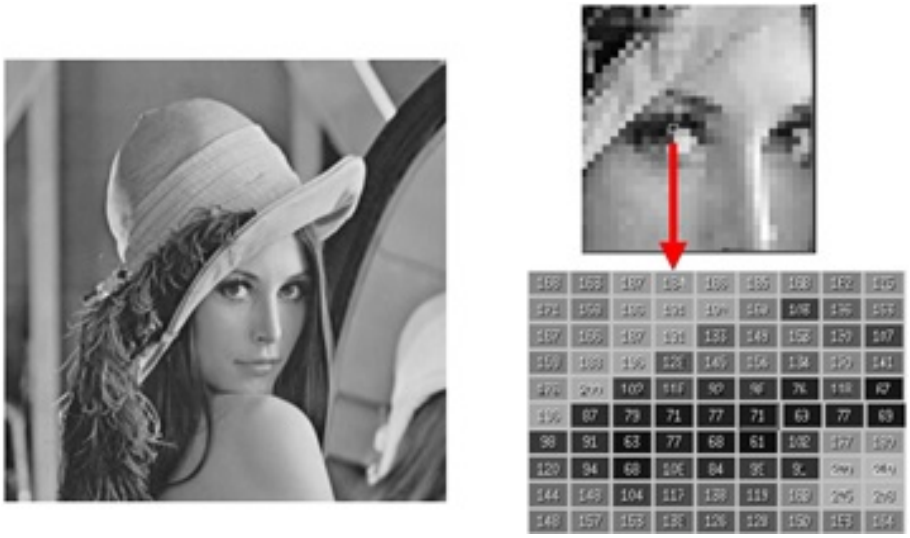


Figura 3.4 – Representação de uma imagem digital

As Figuras 3.4 e 3.5 são exemplos de imagens na chamada grayscale ou seja, imagens em tons de cinza; onde cada pixel armazena apenas um valor inteiro que representa o cinza, que varia de 0 (preto) até 255 (branco).

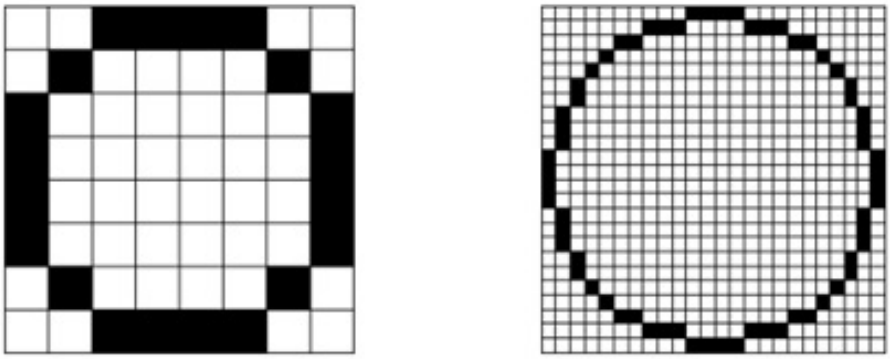


Figura 3.5 – Imagens extraídas com resoluções diferentes

Desta forma, podemos definir imagem como uma matriz onde os índices de suas linhas e colunas identificam o pixel, e seu respectivo valor, que corresponde a cor daquele ponto da imagem (SANTOS, 2012).

3.2 Espaço de Cores

A cor pode ser definida não como uma característica de um objeto, mas sim como uma percepção humana do mesmo. Fato este que faz com que cada indivíduo tenha uma percepção diferente, dependendo de fatores fisiológicos e psicológicos. A distribuição de energia e a luz visível refletida por um objeto, os chamados estímulos da cor, atingem a retina e então o cérebro processa esses estímulos para que se possa enxergar as cores. Ou seja, a cor de um objeto é determinada pela frequência da onda que ele reflete (MELCHIADES; BOSCHI, 1999).

Sendo assim, o olho humano recebe estímulos sobre três canais de cores diferentes para formar uma imagem, surgindo assim o espaço de cores RGB (*red*, *green*, *blue*), no qual a imagem é descrita por três coordenadas: a quantidade de vermelho, verde e azul, como pode ser visto na Figura 3.6.

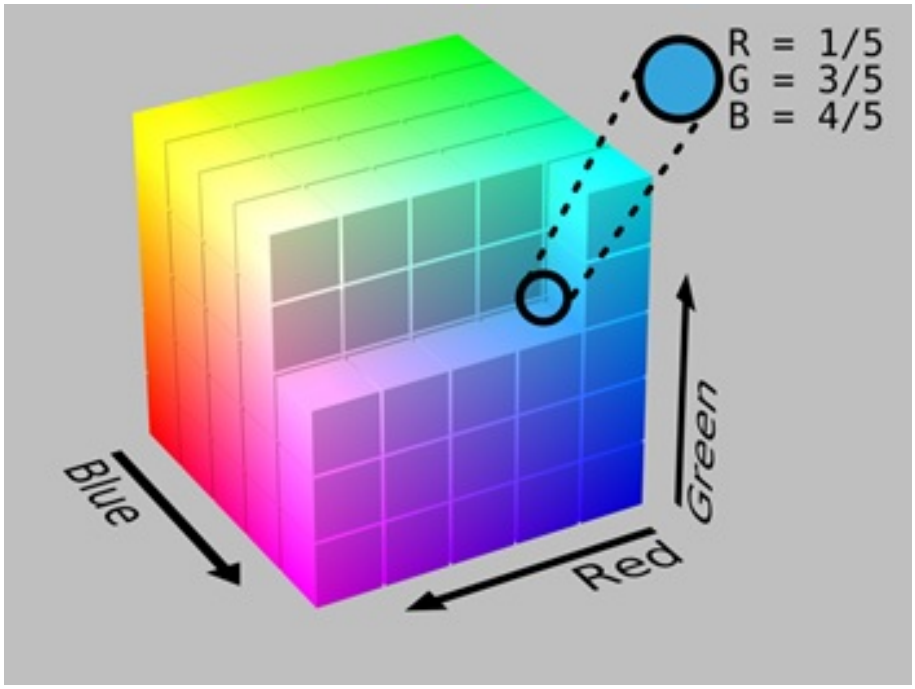


Figura 3.6 – Representação do Sistema de Cores RGB

No espaço de cores RGB, um pixel passa a armazenar um vetor de inteiros de três posições ao invés de um inteiro como é o caso do *grayscale*. A cor visualizada em cada pixel é a combinação das três cores do espaço RGB, também variando entre 0 e 255, como pode ser

visto na Figura 3.6 (SANTOS, 2012).

É muito comum em trabalhos que envolvam processamento de imagens e/ou visão computacional a troca/conversão do espaço de cores, como ocorre no trabalho de Penharbel et. al. (2004), onde foi trocado de RGB para HSI (do inglês *Hue*, *Saturation* e *Intensity*, sendo respectivamente: tonalidade, saturação e intensidade), em virtude da dificuldade de separação de cores (devido à similaridade) no espaço RGB. A representação do espaço de cores HSI passa a ser um cone ou cilindro ao invés de um cubo como ocorre no espaço de cores RGB. Esta representação também é válida para os espaços de cores HSV (do inglês *Hue*, *Saturation* e *Value*, sendo respectivamente: tonalidade, saturação e valor) e HSL (do inglês *Hue*, *Saturation* e *Lightness*, sendo respectivamente: tonalidade, saturação e luminosidade), como pode ser visto na Figura 3.7.

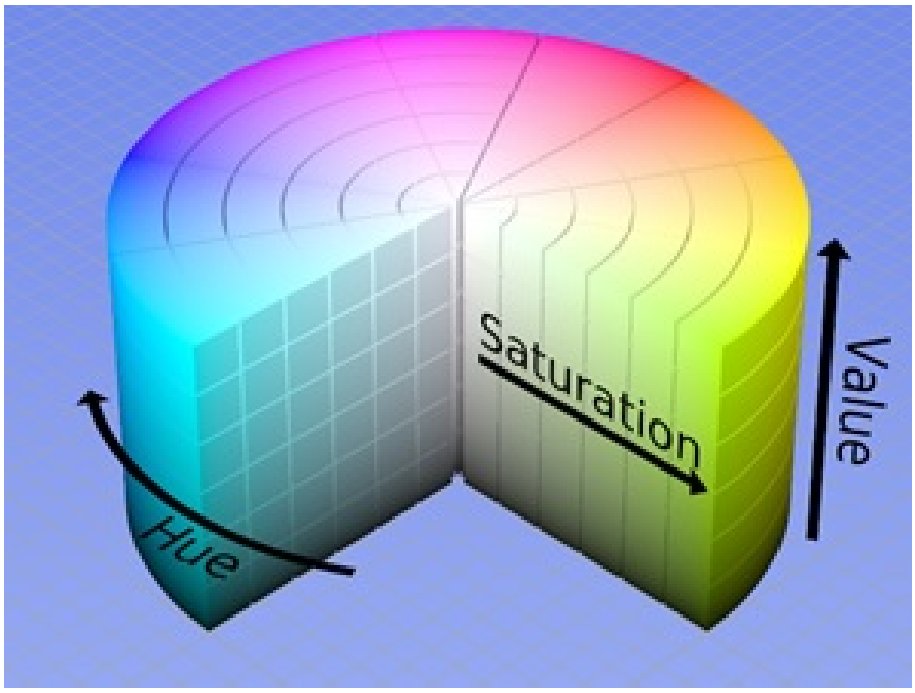


Figura 3.7 – Cilindro para representação do espaço de cores HSV

Um exemplo da facilidade que justifica a escolha da troca do espaço de cores pode ser dado no ato de separar algumas cores, como o amarelo e o azul, que podem ser separadas simplesmente com o parâmetro H, enquanto outras, como o laranja e o vermelho, deverão ser separadas também pela saturação e pela intensidade, pois o parâmetro H de cada uma delas é próximo. Em RGB, este trabalho passa a ser mais trabalhoso (PENHARBEL et al., 2004).

3.3 Transformações Morfológicas

Esta seção aborda o que são transformações morfológicas assim como as principais utilizadas em trabalhos que envolvam processamento de imagens, dentre elas a dilatação, erosão, abertura e fechamento.

Transformações morfológicas são aquelas que, após o seu processamento, resultam na alteração da forma da imagem, mantendo o seu tamanho original. É necessário a definição de um operador a ser aplicado sobre a imagem original, resultando em distorções nos objetos presentes na imagem. Um exemplo de utilização destas transformações é para a remoção de ruídos em imagens, tais como pode ser visto na Figura 3.8 (BRADSKI; KAEHLER, 2008). Ainda existindo outras técnicas para a mesma finalidade de remoção de ruídos.

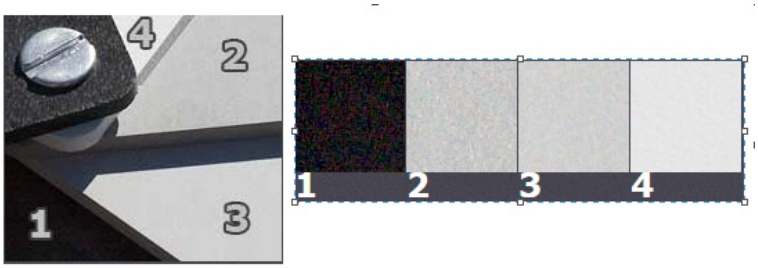


Figura 3.8 – Exemplo de ruído em imagens. Quando mais escuro, maior incidência de ruídos

> "O princípio básico da morfologia matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação através de outro conjunto completamente definido, chamado elemento estruturante. Portanto, a base da morfologia matemática é a teoria de conjuntos. Por exemplo, o conjunto de todos os pixels pretos em uma imagem binária descreve completamente a imagem (uma vez que os demais pontos só podem ser brancos). Em imagens binárias, os conjuntos em questão são membros do espaço inteiro bidimensional Z^2, onde cada elemento do conjunto é um vetor 2-D cujas coordenadas são as coordenadas (x,y) do pixel preto (por convenção) na imagem. Imagens com mais níveis de cinza podem ser representadas por conjuntos cujos elementos estão no espaço Z^3. Neste caso, os vetores têm três elementos, sendo os dois primeiros as coordenadas do pixel e o terceiro seu nível de cinza."(MARQUES FILHO; NETO, 1999)

Este operador à ser definido, também denominado como elemento estruturante, consiste no tamanho e forma no qual o elemento estruturante agirá. A Figura 3.9 possui quatro exemplos de elementos estruturantes de tamanho 7, no qual representa uma imagem de tamanho 7x7 para imagens binárias. Na Figura 3.9(a) denominado *flat*, todos os *pixels* vizinhos ao pixel em questão farão parte da transformação utilizada. Em (b), os pixels que farão parte da transformação formam um X, e em (c) formam um sinal de mais (+) e por último, em (d) formam um círculo (CALXITO, 2003).

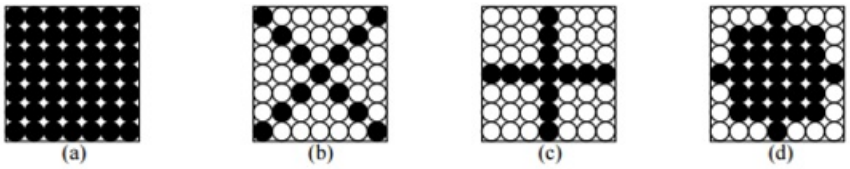


Figura 3.9 – Elementos estruturantes de dimensão 7X7 (CALXITO, 2003)

Além das transformações morfológicas, também existem transformações geométricas, responsáveis por redefinir os espaços dos pontos em uma imagem e podem ser de translação, rotação e escala. Existe também transformações radiométricas no qual o objetivo é redistribuir o nível de cinza preservando as bordas dos objetos (BRADSKI; KAEHLER, 2008).

3.3.1 Dilatação

A dilatação é, ao lado da erosão, as duas principais transformações morfológicas existentes. Após a definição do elemento estruturante, este é aplicado em cada pixel da imagem no qual terá pequenos vãos pequenos ou finos (ruídos) removidos, ou seja, ocorre um enchimento dos objetos das imagens, onde elementos maiores podem "engolir"elementos menores (FACON, 2011).

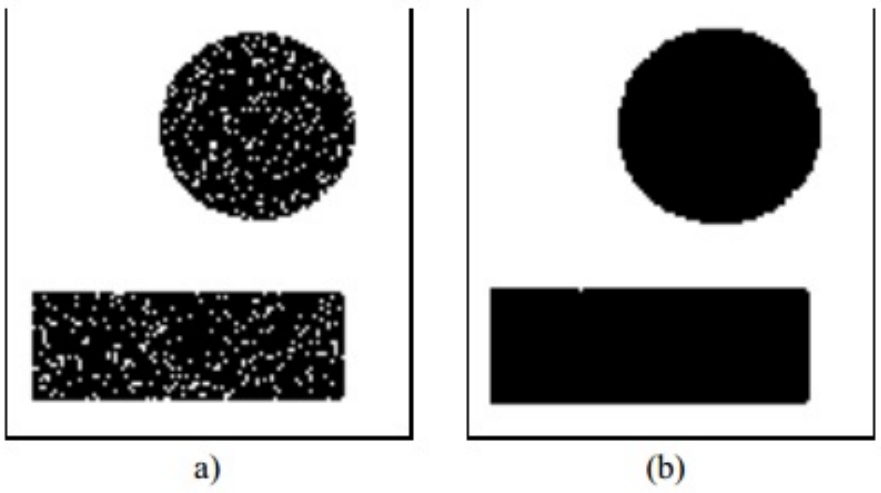


Figura 3.10 – Limpeza de uma imagem ruidosa: (a) Imagem Original, (b) Dilatação (FACON, 2011)

Na Figura 3.10, pode-se observar o resultado de uma dilatação feita utilizando um elemento estruturante, onde os ruídos (pontos brancos) vistos em (a) foram eliminados, resultando na imagem (b).

3.3.2 Erosão

A erosão pode ser facilmente entendida como um processo inverso a dilatação. Ao invés de aumentar os elementos presentes em uma imagem, os elementos são "encolhidos"ou afinados. Esses elementos eliminados não são engolidos por objetos maiores como na dilatação, mas sim, deixam de existir e os elementos maiores terão suas áreas resultante com tamanho reduzido (BRADSKI; KAEHLER, 2008) (FACON, 2011).

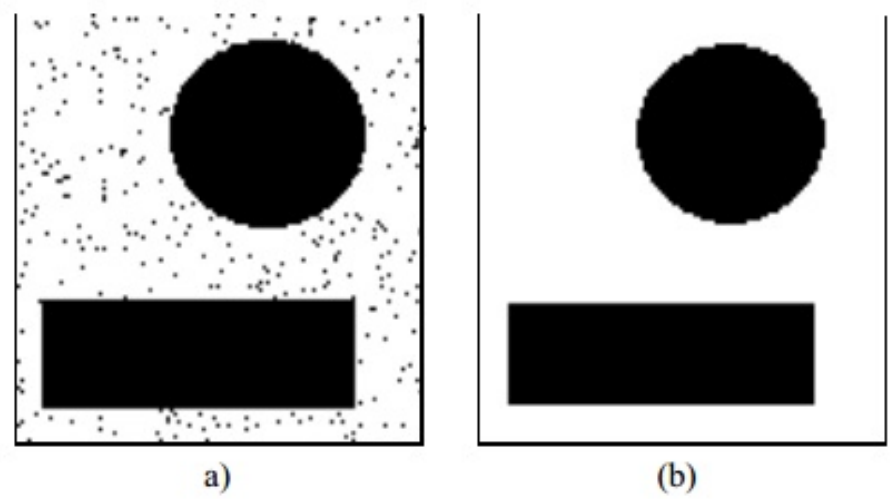


Figura 3.11 – Limpeza de uma imagem ruidosa: (a) Image Original, (b) Erosão dos conjuntos pretos. (FACON, 2011)

Na Figura 3.11, os ruídos representados pelos pequenos pontos pretos (a) na imagem são eliminados após a aplicação da erosão (b).

3.3.3 Abertura e Fechamento

As transformações de abertura e fechamento nada mais são do que combinações sequenciais das já relatadas erosão e dilatação. Por si só, erosão e dilatação alteram o tamanho dos objetos no ato de efetuar a remoção dos ruídos presentes na imagem original. A erosão reduz enquanto que a dilatação aumenta. A partir da combinação de dilatação e erosão é possível manter a forma das áreas dos objetos presentes na imagem original (FACON, 2011) (MARQUES FILHO; NETO, 1999). Dependendo da imagem e da quantidade de clico executados, a foram de um objeto presente na imagem pode ser deformada.

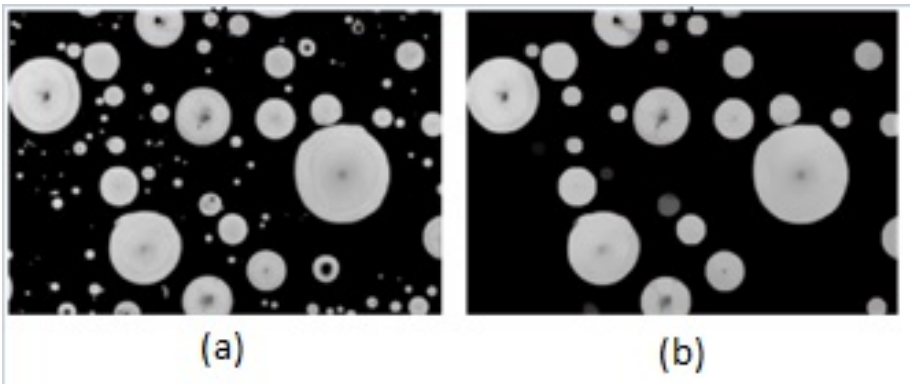


Figura 3.12 – Exemplo de execução de uma transformação de abertura onde (a) é a imagem original e (b) a imagem resultante

A execução de uma erosão seguida por uma dilatação é denominada transformação de abertura. Nesta transformação, os objetos pequenos são eliminados, ocorre a separação dos objetos ligados por poucos pixels e objetos grandes não são afetados. Na Figura 3.12, nota-se o resultado da execução de uma transformação de abertura em (b) no qual os ruídos e os objetos pequenos foram eliminados (CONCI et al., 2004).

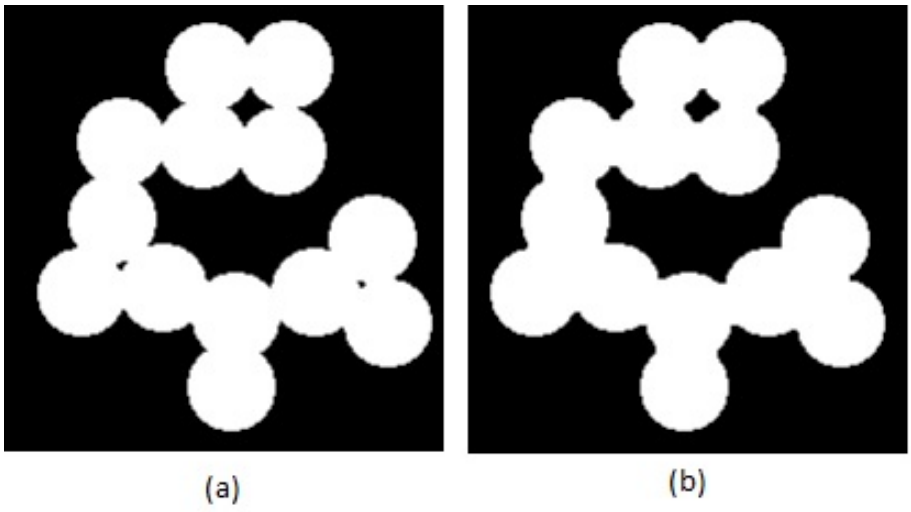


Figura 3.13 – Exemplo de imagem após a execução de um fechamento onde (a) é a imagem original e (b) a imagem resultante

Por sua vez, a transformação de fechamento denomina-se pela execução de uma dilatação seguida por uma erosão. Ao contrário da abertura, ocorre a junção de objetos separados por poucos pixels. Objetos pequenos também são removidos da imagem original. Exemplo de transformação de fechamento pode ser visto na Figura 3.13, onde em (b), objetos com poucos pixels de ligação foram juntados em comparação com a imagem original (a).

3.4 Redução de Cores

Como relatado na Seção 2.2, o ato de identificar mudança de cores pode ser trabalhoso, e muitas vezes requer mudanças no espaço de cores utilizado. Para a utilização do espaço RGB, uma alternativa é a redução de cores de uma imagem. Um trabalho que use contagem e/ou análise de cores, a redução de cores passa a ser uma alternativa para redução do tempo de execução e de erros obtidos devido à fácil divergência de cores no espaço RGB, mesmo quando imperceptível ao olho humano (LAGANIÈRE, 2011). Um código fonte de exemplo pode ser visto abaixo.

```
1
2  void colorReduce(cv::Mat& image, int div = 128)
3  {
4          int nl = image.rows;
5          int nc = image.cols * image.channels();
6
7          for (int j = 0; j < nl; j++)
8          {
9                  uchar* data = image.ptr<uchar>(j);
10
11                 for (int i = 0; i < nc; i++)
12                 {
13                         data[i] = data[i] / div * div + div / 2;
14                 }
15         }
16 }
```

O algoritmo apresentado por Laganière (2011) usa um parâmetro inteiro denominado fator de redução (*div* no código fonte acima), no qual cada um dos três canais, de cada pixel da imagem, tem seu valor divido por este fator de redução e, então, a parte inteira do resultado da divisão é multiplicado pelo fator de redução, obtendo-se o maior múltiplo do fator de redução menor que o valor inicial; à este múltiplo soma-se o fator de redução dividido por 2 e obtém-se então a posição central do intervalo entre dois múltiplos adjacentes do fator de redução. A Figura 3.14 apresenta alguns exemplos da execução do algoritmo de redução de cores para diferentes valores ao fator de redução (*div*).

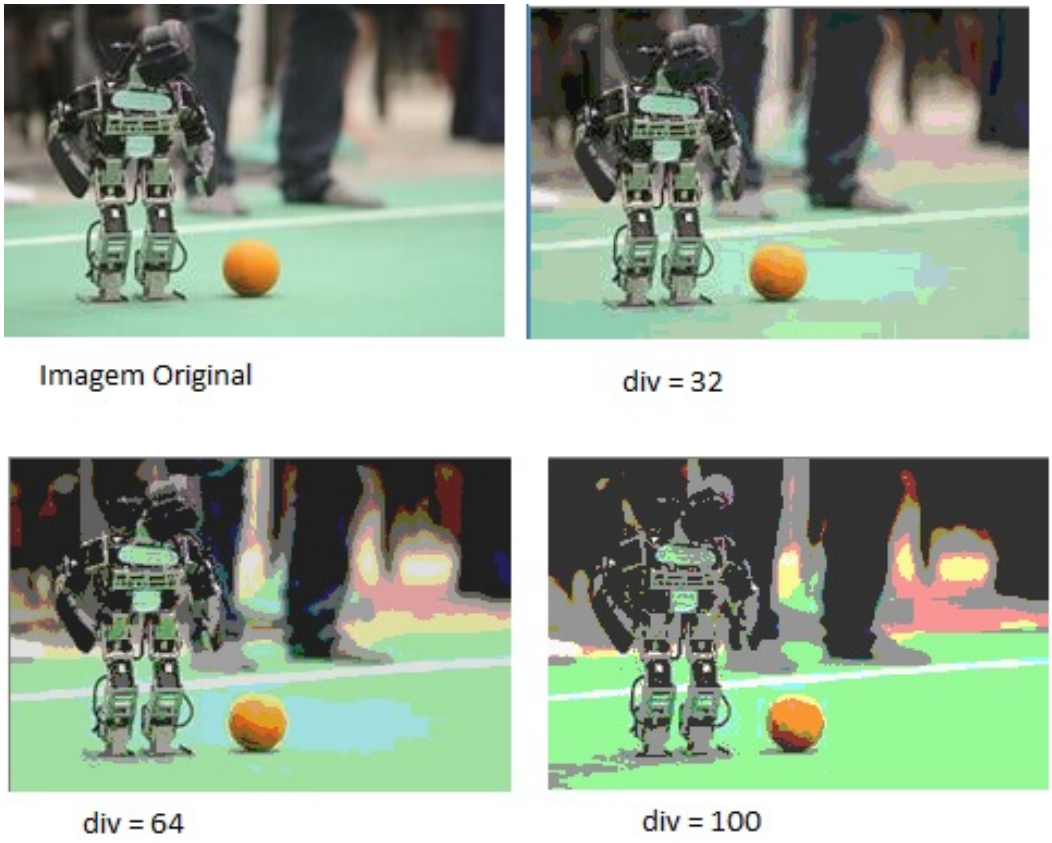


Figura 3.14 – Imagens resultantes após a execução do algoritmo de redução de cores

3.5 Cálculo de Similaridade

Cálculos de similaridade são utilizados para definir o grau de similaridade entre objetos distinto. A grande maioria dos cálculos de similaridades usa vetores que armazenam características extraídas destes objetos, que podem ser imagens por exemplo (SANTOS, 2012).

O cálculo de similaridade de cosseno é utilizado em mineração de dados e comparação de documentos. Existindo palavras em comum, método mais populares como a Distância Euclidiana e os cálculos de coeficiente de correlação de Pearson tornam-se inapropriados (TAN, 2005).

Para se calcular a similaridade de cosseno, deve-se gerar um vetor dos atributos que estão sendo analisados. O vetor é utilizado para encontrar o produto escalar normalizado entre os dois vetores que estão sendo analisados. Então, calcula-se o cosseno do ângulo entre os dois vetores; quanto mais próximo a 1, mais similares são os objetos, consequentemente, quanto mais próximo a -1, mais divergentes são os objetos.

$$similarity(x, y) = cos(\theta) = \frac{x \cdot y}{||x|| * ||y||}$$

Figura 3.15 – Equação matemática da Similaridade de cosseno

Santos (2012) destaca a utilização desta métrica de similaridade em seu trabalho que envolve processamento de imagens. O autor utiliza a métrica de similaridade entre duas imagens, onde é feita a medida utilizando o cosseno do ângulo formado por seus vetores de características (cor). O cosseno do ângulo é independente do tamanho dos vetores de características.

3.6 Sumário do Capítulo

Este capítulo aborda os principais conceitos necessários para uma melhor compreensão da abordagem proposta de bússola visual para veículos autônomos. No início foi apresentado conceitos de visão computacional e processamento de imagens bem como a literatura aborda os limites conceituais de um assunto para com o outro. Incluído nessa temática, conceitos de imagens digitais e cores. As técnicas de transformações morfológicas utilizadas, tais como dilatação, erosão, abertura e fechamento, e dois aspectos importante, a redução de cores e cálculos de similaridades.

O capítulo seguinte descreve a metodologia do presenta trabalho, apresentando os passos para a construção da bússola visual, desde os primeiros passos compostos pela etapa de processamento de imagens e extração das áreas de interesse, bem como é efetuado o cálculo de similaridade destas áreas de interesse para a estipulação da orientação.

4 ESTIMATIVA DE ORIENTAÇÃO BASEADA EM CORES

A abordagem a ser apresentada tem como objetivo ser a mais genérica possível, ou seja, livre de contexto e de marcações no cenário. Também deve ser aplicada, além do futebol de robôs, para auxiliar na estipulação da orientação em sistemas que fazem uso de odometria, corrigindo ou alterando o grau de certeza quanto à uma orientação determinada.

A bússola visual desenvolvida é baseada em cores, no qual se efetua uma análise e armazenamento de informações referente as cores do cenário no qual um veículo ou robô autônomo pode se locomover. Essa ação deve ser refeita sempre que houver alterações no ambiente, tais como variação da iluminação devido ao horário do dia, mudanças climáticas ou disposição de pessoas e/ou objetos. Então, o cenário no qual o indivíduo autônomo se encontra após esta calibração é comparado com as informações armazenadas do cenário para estipular a orientação do indivíduo.

A escolha desta abordagem se deu devido à ser uma solução mais simples comparada a técnica de SLAM (Localização e Mapeamento Simultâneos) por exemplo (HAN; LEE; JI, 2010), no qual é uma técnica utilizada por robôs e veículos autônomos para construir um mapa de um ambiente ao mesmo tempo que se localiza; além de o foco ser diferente. Enquanto o SLAM foca em efetuar um mapeamento e identificar a localização de robôs e veículos, a abordagem apresentada neste trabalho tem foco no estipular a orientação desses robôs e veículos. A escolha de se trabalhar com as cores do ambiente se dá devido à inexistência de padrões dos quais não se pode garantir que façam parte de um cenário e poderiam auxiliar na orientação; sendo assim, esta proposta de bússola visual se torna genérica e livre de contexto.

Quase todos os trabalhos de visão computacional começam com a definição do que é necessário quanto ao processamento de imagens. Primeiramente, foram desenvolvidos alguns filtros de processamento de imagens para a aquisição de uma imagem adequada para se trabalhar, no caso, livre de ruídos e de grande variação de cores, utilizando a biblioteca de visão computacional e processamento de imagens OpenCV desde o primeiro protótipo.

A Figura 4.1 ilustra o processo como um todo, construído na Notação BPMN – *Business Process Model and Notation* (2011) em forma de um diagrama independente de metodologia.

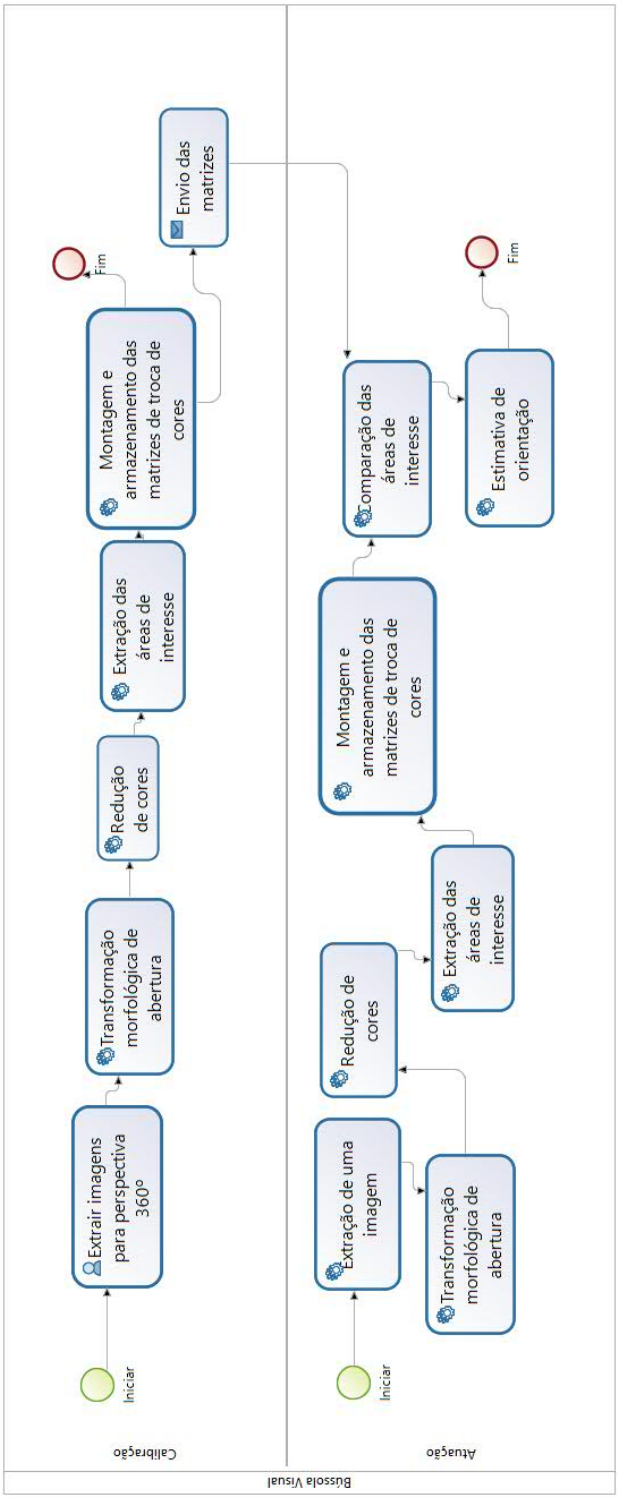


Figura 4.1 – Representação do fluxo de execução da bússola visual, com a etapa de calibração e atuação separadas

O processo foi separado entre as duas principais etapas da bússola visual. A calibração, que consiste em preparar a perspectiva do ambiente; não é regra que as imagens sejam/formem uma perspectiva 360º, mas deve ser informado o ângulo de abertura da imagem para se efetuar os cálculos de maneira correta. A outra etapa denominada atuação, consiste no ato de extrair imagens dos robôs ou veículos autônomos e, após se efetuar as etapas de processamento de imagens descritos à seguir, calcular a similaridade e consequentemente o ângulo destas imagens na perspectiva 360º resultante na etapa de calibração.

4.1 Calibração

Para que a bússola visual funcione, a etapa de calibração deve ser efetuada de maneira atenciosa. O primeiro passo é informar à bússola a imagem a ser utilizada como base. Recomenda-se que esta imagem seja uma perspectiva 360º do ambiente/contexto no qual os robôs ou veículos autônomos serão inseridos. Independente disto, o ângulo de abertura da imagem deve ser informado. Não foi desenvolvido um módulo específico para a extração de imagens, o que permite que isto seja feito com a utilização de uma câmera, celulares e *webcams*, estas embutidas ou não em robôs ou veículos autônomos.

Para a montagem de uma perspectiva 360º de um cenário, pode-se agrupar uma sequência de imagens, e uma etapa fundamental é a definição das técnicas de processamento de imagens a serem utilizadas. Ao ser efetuada de maneira concisa, essa calibração diminui a chance de erros por parte do algoritmo de bússola visual, sendo assim, uma etapa crucial para o funcionamento desta abordagem.

Cada uma das etapas seguintes serão explicadas individualmente nas subseções a seguir, dando inicio à transformação morfológica de abertura e a redução de cores, que fazem parte das técnicas de processamento de imagens seguidas pela extração das áreas de interesse e da montagem das matrizes de cores.

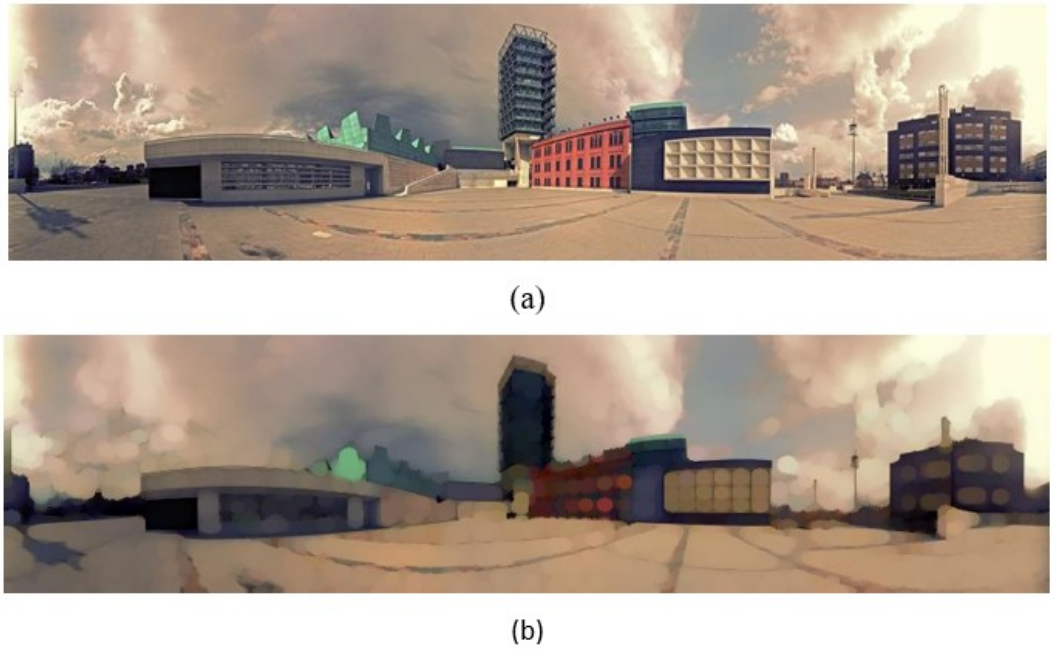
4.1.1 Transformação morfológica de abertura

Para o(s) *frame(s)* retirado(s) , executam-se etapas de processamento de imagens com o objetivo acarretar maior similaridade entre as cores presentes, podendo este ser um passo opcional, aplicado somente para deixar as cores mais homogêneas, ou seja, remover ruídos na imagem. No caso, foi aplicada a transformação morfológica de abertura a qual é obtida pela

erosão da imagem seguida por uma dilatação, o que acarreta na remoção de pequenos ruídos (OPENCV, 2016).

O código da erosão começa com a definição do tamanho da erosão e da definição da estrutura morfológica para então executar a função *erode* da biblioteca OpenCV. O código completo desta etapa pode ser visto no Apêndice B. Não é regra que este deva ser o código utilizado, o importante é efetuar a erosão.

```
1
2 int erosion_size = 2;
3 Mat element = getStructuringElement(cv::MORPH_OPEN,
4                 cv::Size(2 * erosion_size + 1, 2 * erosion_size + 1),
5                 cv::Point(erosion_size, erosion_size));
6 erode(image, image, element);
7 .
```



(a)

(b)

Figura 4.2 – Exemplo de uma imagem original (a) e após execução da transformação morfológica (b)

A cor em uma imagem pode ser representada pelas intensidades dos canais vermelho, verde e azul no sistema de cores RGB. Independente do sistema de cores utilizado, uma pequena alteração em um dos valores (de intensidade de canal, por exemplo), não pode-se considerar como a mesma cor, mesmo que a olho nu se pareçam a mesma.

Apenas a execução de uma transformação morfológica de abertura e até mesmo outros filtros de imagens não se fazem suficientes para a abordagem proposta, então, foi utilizado um algoritmo para redução de cores, com o intuito de corrigir as pequenas alterações nos canais de

cores que se identificam como troca de cor (LAGANIÈRE, 2011).

Figura 4.3 – Variações dos tons de cor vermelha, onde cada pixel para esquerda ou direita caracteriza uma troca de cor

4.1.2 Redução de Cores

Um algoritmo de redução de cores é necessário para este trabalho devido à abordagem trabalhar com identificação de troca de cores de cada coluna. A ausência da etapa de redução de cores resultaria em um trabalho massivo, porque quaisquer pequenas variações de cores implicariam novas mudanças a serem processadas pela abordagem, tal como pode ser visto na Figura 4.3. Um exemplo de código fonte para redução de cores pode ser visto em no Capítulo 2.5 (LAGANIÈRE, 2011). Novamente, não sendo regra a utilização do código apresentado, mas sim aplicar uma redução de cores. O resultado da redução pode ser visto na Figura 4.4.

Figura 4.4 – Imagem resultante após a redução de cores

Pode-se ainda aplicar alguns outros filtros para melhorar a imagem e assim facilitar a identificação de padrões ou segmentação de objetos; julgou-se, contudo, suficiente o que foi aplicado até o momento.

4.1.3 Extração das áreas de interesse

Com o *frame* resultante da etapa de processamento de imagem (transformação morfológica de abertura e redução de cores), separa-se a imagem em colunas de 10 pixels de largura, nas quais são analisados e extraídos os seus histogramas (ocorrência de cada cor na imagem) e onde as trocas de cores ocorrem (KOK; METHENITIS; NUGTEREN, 2013). Esses dados são necessários para análise das colunas dos frames que surgirão para comparação, conforme a

movimentação do indivíduo autônomo. A Figura 4.5 destaca as áreas de interesse. Estas áreas foram analisadas e são armazenadas as trocas de cores que ocorreram em cada uma delas, como por exemplo:

```
64 64 64(cinza) para 192 64 64 (vermelho): 23
192 64 64(vermelho) para 64 64 64 (cinza): 8
64 64 64 (cinza) para 64 64 192 (azul): 14
64 64 192 (azul) para 64 64 64 (cinza): 4
.
```

Cada linha acima traz a contagem de troca de cores, sendo a primeira cor a origem e a segunda o destino, seguida pela quantidade de vezes que esta troca de cor ocorreu.

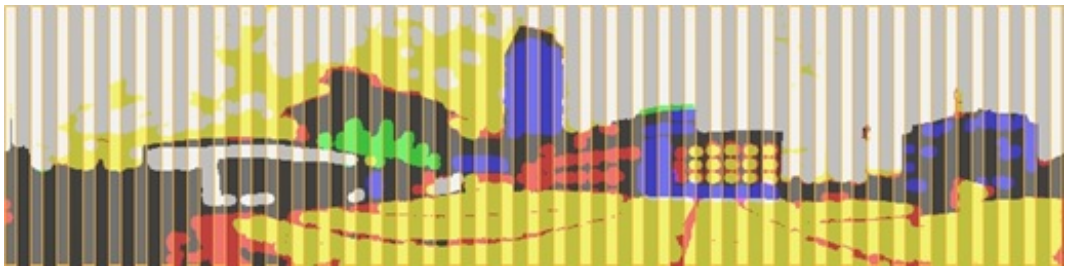


Figura 4.5 – Imagem com destaques nas áreas de interesses (Colunas)

Após ter armazenado as informações de troca de cor para cada uma das áreas de interesse, deve se começar a trabalhar esses dados para se efetuar o cálculo de similaridade. O código fonte utilizado nesta etapa da bússola visual pode ser visto no Apêndice D.

4.1.4 Matrizes de troca de cores

Antes de se efetuar o cálculo de similaridade, deve-se montar uma matriz referente a troca de cores ocorridas nas áreas de interesse destacadas na seção anterior. Neste trabalho foi utilizada a similaridade de cosseno. Para esta métrica, deve-se gerar um vetor dos atributos que estão sendo analisados, no caso da bússola visual, a contagem de trocas de cores que ocorrem em cada coluna. O vetor é utilizado para encontrar o produto escalar normalizado entre os dois vetores que estão sendo analisados, neste caso, um proveniente da bússola visual e outro do indivíduo autônomo durante sua locomoção. Então, calcula-se o cosseno do ângulo entre os dois vetores; quanto mais próximo a 1, maior o grau de similaridade entre os objetos, consequentemente, quanto mais próximo a -1, mais divergentes são os objetos.

O primeiro passo é construir matrizes que correspondessem às trocas de cores que ocorrem em cada coluna (área de interesse) anteriormente separada na qual podem ser vistas na

Figura 4.6, e então aplicar o algoritmo da similaridade de cosseno para efetuar a comparação entre essas matrizes.

	0	0	1	0	4	0	0	0
	0	0	0	25	76	0	0	0
	0	0	0	0	1	0	0	0
	0	27	0	0	2	0	0	0
	0	64	0	4	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Figura 4.6 – Representação de uma matriz de troca de cores

As matrizes foram criadas baseada no número de cores existentes nas imagens, no caso, existem apenas oito cores resultantes do algoritmo de redução de cores. Cada uma dessas cores (amarelo, cinza, vermelho, azul, verde, rosa, branco e ciano) foram definidas como índices para as linhas e colunas da matriz, como pode ser visto na Figura 4.6. O conteúdo das matrizes representa a troca de cores existente entre as cores das linhas com as cores das colunas, por exemplo na Figura 4.6, houve 64 trocas de verde para cinza e 25 trocas de cores da cor cinza para a cor azul.

No Apêndice E é apresentado o código fonte responsável por fazer a contagem da troca de cores. Após este passo, o Apêndice F traz o código fonte responsável de, com o resultado da contagem, montar as matrizes de troca de cores, para então, na etapa de atuação, efetuar os cálculos de similaridade.

Com as matrizes devidamente geradas e armazenadas, a bússola visual encontra-se calibrada e pronta para ser utilizada. Na seção seguinte é abordado a etapa de atuação, no qual a bússola necessita estar calibrada para funcionar.

4.2 Atuação

A etapa de atuação é onde o robô ou veículo autônomo vai se locomover no cenário, e em algum dado momento ele precisará se orientar. Neste momento o robô ou veículo retira e envia para a bússola visual uma nova imagem que será comparada com a perspectiva 360º

montada na etapa de calibração.

As quatro subetapas salientadas na etapa de calibração são reexecutadas para cada imagem recebida pela bússola visual na atuação. Ou seja, para cada imagem enviada pelo robô ou veículo autônomo é feita a etapa de transformação morfológica de abertura, redução de cores, extração das áreas de interesse e a montagem das matrizes de troca de cores. É importante frisar que as etapas podem ser alteradas, novos filtros de processamento de imagens podem ser aplicados conforme a necessidade dos requisitos da aplicação, mas é de suma importância que as etapas aplicadas na calibração da bússola visual sejam exatamente as mesmas aplicadas na etapa de atuação para não acarretar mal funcionamento da abordagem proposta.

Neste ponto, temos as matrizes de troca de cores devidamente armazenadas, tanto para a bússola visual (calibração) como para a etapa de atuação (robô ou veiculo autônomo em movimento no cenário). Com essas informações, analisam-se as matrizes para estipular a probabilidade de similaridade entre elas, a fim de direcionar o indivíduo autônomo para direcionar-se corretamente para o objetivo previamente estipulado.

4.2.1 Cálculo de Similaridade

Após gerar uma matriz para cada coluna (área de interesse), percorrem-se as matrizes da bússola e estas são comparadas utilizando a similaridade de cosseno, uma a uma, com as matrizes dos *frames* teoricamente provenientes do indivíduo autônomo, ou seja, cada matriz da bússola (calibração) é comparada com todas as matrizes dos *frames* (atuação). O código desenvolvido na linguagem de programação C++ referente à Similaridade de Cosseno pode ser visto no Apêndice A.

Com a execução dos cálculos de similaridade, obtém-se valores que correspondem à similaridade de cada uma das matrizes (colunas), e assim, os maiores valores correspondem a área de interesse, ou seja, a direção na qual o indivíduo autônomo deve se direcionar. Mas para isso, apenas uma coluna não pode servir como parâmetro, e sim um aglomerado delas, levando as matrizes vizinhas como parâmetro. Por exemplo, soma-se o valor de similaridade da matriz de índice 6 com as matrizes de índice 4, 5, 7 e 8. Isto se da devido a uma única coluna não apresentar resultado suficiente para estipular a similaridade. O processo é refeito com todas as similaridades resultantes. Vale salientar que esta soma com os valores de duas matrizes para a esquerda e duas para a direita pode ser alterado conforme for constatado a necessidade.

4.2.2 Estimativa de orientação

Identificado o índice da matriz que obter o maior valor de similaridade, então basta efetuar uma simples regra de três para informar para a inteligência artificial (IA) do robô ou veiculo autônomo, para que ângulo ele esta direcionado na perspectiva 360º.

```
1
2 //imageAngle = 360; constante global (angulo da imagem da calibracao)
3 //mostSimilarColumn = 23; parametro (indice da coluna mais similar)
4 //CountColumn = 41; constante global (quantas matrizes existem)
5 double angle = (mostSimilarColumn * imageAngle) / CountColumn;
6 .
```

Nesta etapa fica evidente a importância de informar corretamente o ângulo de abertura da imagem utilizada. Uma informação errônea pode afetar os resultados, orientando incorretamente o robô ou veiculo autônomo. A variável *angle* receberá o resultado, ou seja, o angulo de maior similaridade da imagem recebida na etapa de atuação dentro da imagem da calibração. Esta variável pode ser retornada para a IA de um robô ou veiculo autônimo por exemplo, e auxiliar na tomada de decisão.

O ângulo resultante pode ser combinado com informações nos exemplos abaixo:

- Onde fica a porta de saída?
- Onde fica a garagem?
- Onde fica o gol adversário?

Sabendo-se o ângulo mais similar, e sabendo em que ângulo localiza-se a resposta para alguma das perguntas acima na perspectiva 360º, pode-se combiná-las de maneira à contribuir significativamente para a tomada de decisão.

4.2.3 Ponto de corte da similaridade

É possível que ocorram casos no qual o cálculo de similaridade não encontre valores que representem uma similaridade adequada e consequentemente não retorne um ângulo suficientemente correto. Por exemplo, a similaridade se torna mais plausível com valores mais próximos a 1 para cada coluna, e soma-se os valores de uma coluna com suas duas colunas da esquerda e duas da direita. Logo, quando mais próximo de 5 o somatório das colunas, mais similar é uma determinada região.

Pode-se ocorrer deste somatório resultar que a região mais similar resulte em um valor aproximado de 3 por exemplo. Nestes casos, deve-se ignorar o *frame* recebido e repetir o procedimento com um novo *frame*. Um valor 3 pode sim ser a região mais similar, mas essa margem de erro pode prejudicar a tomada de decisão por parte dos robôs ou veículos autônomos, tais como decidir chutar à gol em futebol de robôs se o placar é favorável ou não e acabar fazendo um gol contra (FUGALI; LIBRELOTTO, 2016), então, fica a cargo de cada aplicação como tratar esse grau de certeza. Recomenda-se utilizar como grau de certeza aceitáveis, valores acima de 4,5.

Como anteriormente relatado, fica a cargo de quem esta desenvolvendo a bússola aplicar as devidas etapas de processamento de imagens. Caso a similaridade mantem-se inferior a 4,5, esta etapa precisa ser revisada ou aperfeiçoada, isso depende se foi desenvolvido de maneira diferente da apresentada neste trabalho ou se foi aplicado em um cenário totalmente diferente dos testes efetuados neste trabalho.

4.2.4 Combinação de bússolas (calibrações)

Mesmo após uma adaptação das técnicas de processamento de imagens não acarretarem em melhoria nos resultados de similaridade, a utilização de mais de uma bússola calibrada, ou seja, mais de uma perspectiva 360º de um mesmo cenário/ambientes, mas em posições diferentes, pode surgir como alternativa. Por exemplo: ao identificar que uma bússola não atingiu o grau aceitável de similaridade (4,5) em uma perspectiva em 360º do ambiente, repete o procedimento de encontrar região de alta similaridade em uma próxima perspectiva. Um exemplo disto é o futebol de robôs, como pode ser visto na Figura 4.7, aplicada ao futebol de robôs, no qual em cada um dos círculos (região amarela), pode-se efetuar uma calibração (montagem da perspectiva em 360º).

É difícil prever quantas perspectivas devem ser criadas, e até mesmo se realmente existe a necessidade de mais de uma perspectiva. Como foi visto, a área de visão computacional é bastante complexa e muito dependente de intervenção humana para cada caso de estudo (GONZALEZ; RICHARD, 2002)(BRADSKI; KAEHLER, 2008) (MARENGONI; STRINGHINI, 2009)(SHAPIRO; STOCKMAN, 2001), tornando assim, árduo calcular a real necessidade ou um valor de contagem genérica de perspectivas para um determinado contexto no qual esta bússola visual pode ser aplicada.

A inserção de muitas perspectivas, em alguns casos, pode se tornar apenas redundante.

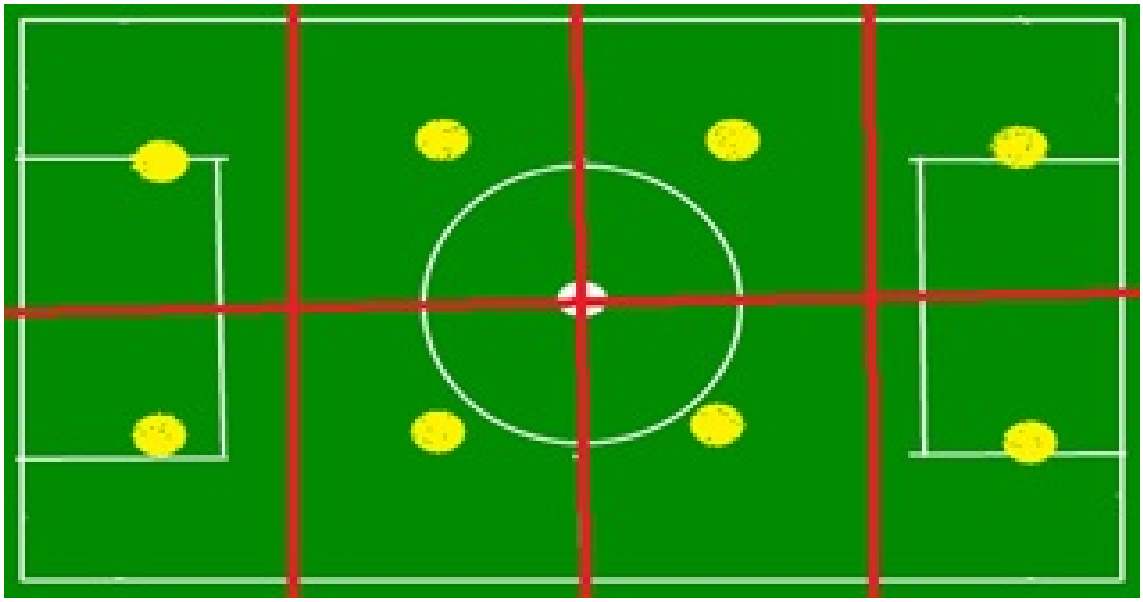


Figura 4.7 – Representação de um campo de futebol dividido em regiões

Em outros, pode servir para aumentar o grau de certeza acerca de uma região específica; e ainda em outros, ser fundamental para o bom funcionamento da bússola. Por exemplo, em um cenário estreito e longo, como um corredor, a montagem de diferentes perspectivas ao longo do percurso torna-se uma boa alternativa. Por sua vez, um cenário pequeno e circular, pode não ser uma boa alternativa, sendo uma perspectiva suficiente. Entretanto, deve-se testar e tratar cada caso individualmente para se estipular a necessidade ou a quantidade de perspectivas para um cenário específico, fazendo assim, com que a bússola funcione de maneira adequada.

4.3 Sumário do capítulo

Este capítulo relatou os passos utilizados na abordagem apresentada para o auxílio da estimativa de orientação de indivíduos autônomos. Foi abordado sobre os passos de processamento de imagens assim como a separação das áreas de interesse e o principal conteúdo da abordagem, o cálculo de similaridade entre essas áreas de interesse, no caso, a similaridade de cosseno.

Os passos descritos foram separados entre os dois módulos da bússola visual desenvolvida neste trabalho, a calibração, que consiste na elaboração da perspectiva 360º de um cenário ou ambiente e a atuação, que consiste no robô ou veículo autônomo em movimentação e com necessidade de se orientar.

Conforme o que foi apresentado, o próximo capítulo aborda uma sequência de testes

com o intuito de apresentar as vantagens e desvantagens da abordagem proposta, estipulando assim, os melhores contextos (ambientes) no qual seu uso possa se tornar mais confiável como aqueles no qual seu uso não é recomendado.

5 ESTUDO DE CASO

Neste capítulo, serão apresentados alguns dos testes efetuados no protótipo desenvolvido, assim como seus pontos fortes, aqueles no qual a confiabilidade é alta e os pontos ou situações no qual a abordagem pode apresentar algumas irregularidades que possam vir a acarretar falta de confiabilidade devido à falta de determinação do ângulo correto assim como o encontro de mais de um ângulo similar.

5.1 Estudo de Caso 1: Fragmento de uma imagem

O primeiro teste apresentado consiste em encontrar similaridade entre a Figura 5.1 e a Figura 5.3, procurando a área mais similar. A Figura 5.1 apresenta a imagem original, e a Figura 5.2 apresenta a imagem resultante após os passos de processamento de imagens previamente apresentados no Capítulo 4.1 na etapa de calibração, além das áreas de interesse devidamente numeradas para melhor entendimento dos resultados. O mesmo foi feito com a imagem a ser analisada/procurada na imagem, na etapa de atuação, no caso, a Figura 5.1.

Figura 5.1 – Imagem original

O resultado esperado é que as colunas destacadas na Figura 5.3 (b) sejam equivalentes respectivamente as colunas 23, 24, 25, 26 e 27 da Figura 5.2; região de origem da Figura 5.1.

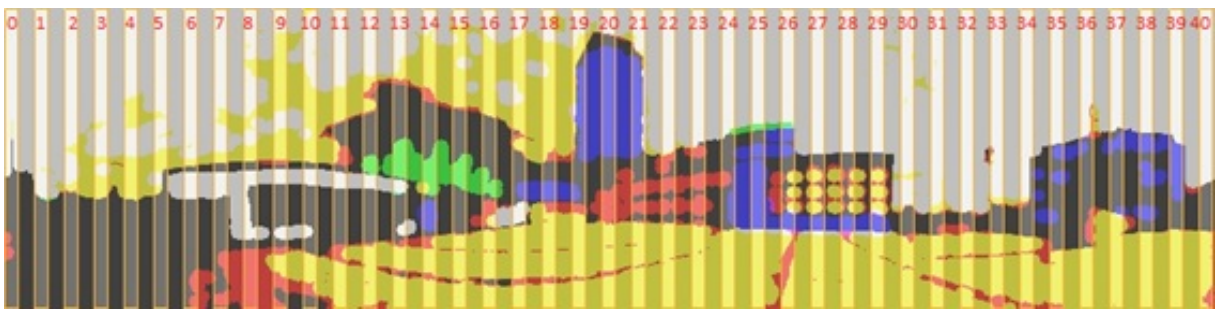


Figura 5.2 – Imagem após processamento e destaque das áreas de interesse numeradas

No protótipo desenvolvido, existe uma série de laços de repetições que percorre cada uma das áreas de interesse da imagem, ou seja, cada uma das matrizes que contém as informações sobre as trocas de cores, e cada uma destas matrizes são comparadas com todas as matrizes do outro *frame* (imagem), tal como descrito no capítulo 4.1.3. O resultado da similaridade de cada uma das colunas, como pode ser visto na Figura 5.4.

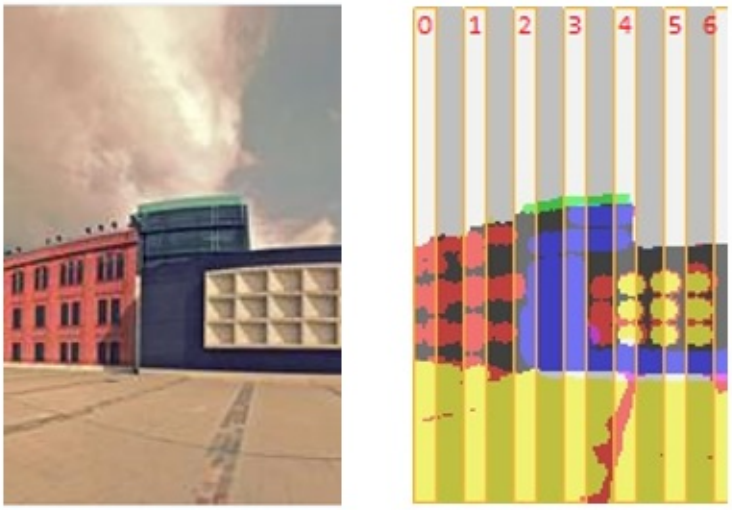


Figura 5.3 – (a) Imagem Original e (b) a imagem após processamento de imagens com as áreas de interesse destacadas

Ainda na Figura 5.4, a área destacada em vermelho apresenta a região de maior similaridade calculada, onde é informado por exemplo, que a área de interesse 23 da primeira imagem (bússola) é equivalente a área de interesse 1 da segunda imagem. Esta equivalência, pelo cálculo de similaridade de cosseno apresentado no capítulo 4.2.1 resultou em 0.998076, sendo bem próximo a 1, onde as áreas seriam totalmente similares ou no caso, as mesmas. O resultado ainda apresenta uma sequência lógica, onde a área 24 é equivalente a área 2, a 25 com a 3 e assim sucessivamente.

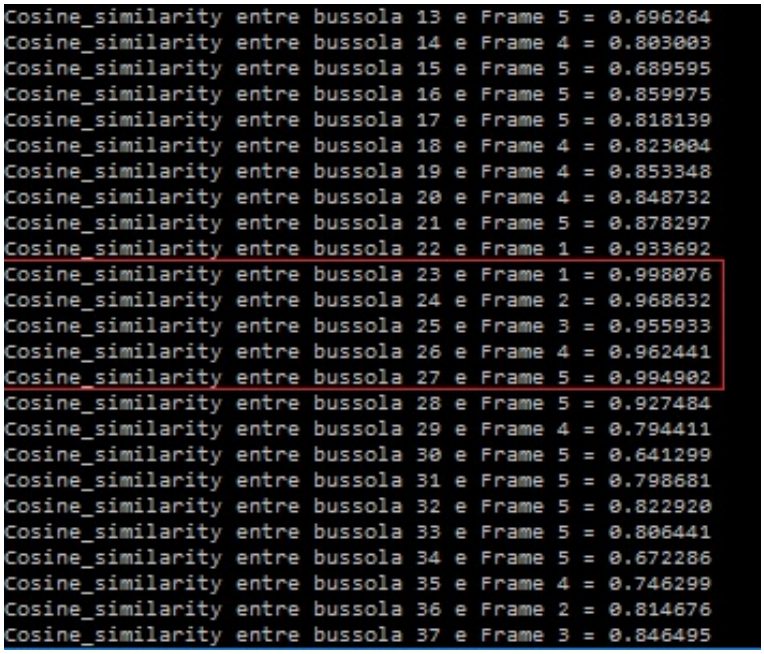


Figura 5.4 – Similaridade resultante entre as áreas de interesse

Após ter a similaridade calculada, soma-se à essa os dois próximos valores das áreas à esquerda e à direita. A soma resultante de maior valor é considerada a região mais similar e consequentemente, calcula-se o ângulo. Sabendo quais as áreas de interesse pertencentes a região similar entre as imagens, calcula-se o ângulo com uma simples regra de três conforme apresentado no capítulo 4.2.2.

Logo, a Figura 5.3 (a) possui maior similaridade no ângulo 201º da Figura 5.1. Informação esta que um robô ou veículo autônomo pode utilizar para estimar sua orientação, adicionando esta informação a sua tomada de decisão, que pode ser algo como ir até algum objetivo e voltar para "casa", ou evitar gols contras no futebol de robôs, que é um problema muito comum e impactante, tal como relatado por Fugali e Librelotto (2016).

5.2 Caso de Estudo 2: Ambiente fechado

O segundo teste apresentado faz uso da imagem em 360º de um ambiente fechado visto na Figura 5.5, no caso o laboratório de Linguagens de Programação e Banco de Dados da UFSM. Para a extração desta imagem, foi utilizado um *smartphone* com um aplicativo de extração de imagens em 360º.

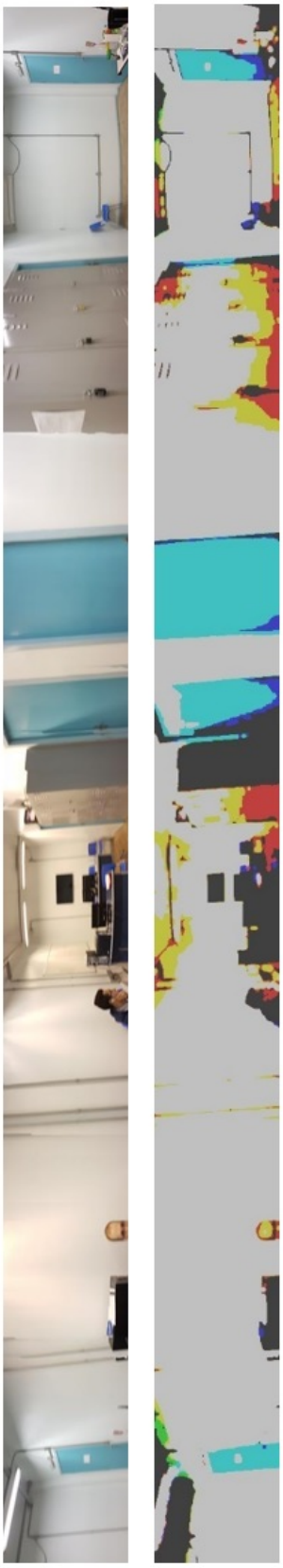


Figura 5.5 – Imagem 360º de um ambiente fechado e sua versão após as etapas de processamento de imagens

O objetivo deste teste é encontrar a Figura 5.6 incluída na Figura 5.5, um teste de grande valor devido à ocorrência de várias regiões similares ao longo de toda a imagem, no caso, a porta azul envolta de um ambiente branco. Após efetuado os mesmos passos de processamento de imagem, extração das áreas de interesse e cálculo de similaridade descritos ao longo do capítulo 4; o algoritmo calculou corretamente o ângulo referente à similaridade entre as imagens.

Figura 5.6 – Imagem à ser procurada

Novamente se obteve resultados satisfatórios, como pode ser visto na Figura 5.7, onde se teve uma sequência lógica entre as matrizes similares. Embora outras regiões da imagem original tiveram valor de similaridade altos (regiões que apresentam portas parecidas envolta do mesmo ambiente branco), a região que envolvam as áreas de interesse 80 até 92 tiveram a similaridade de valor mais expressivo.

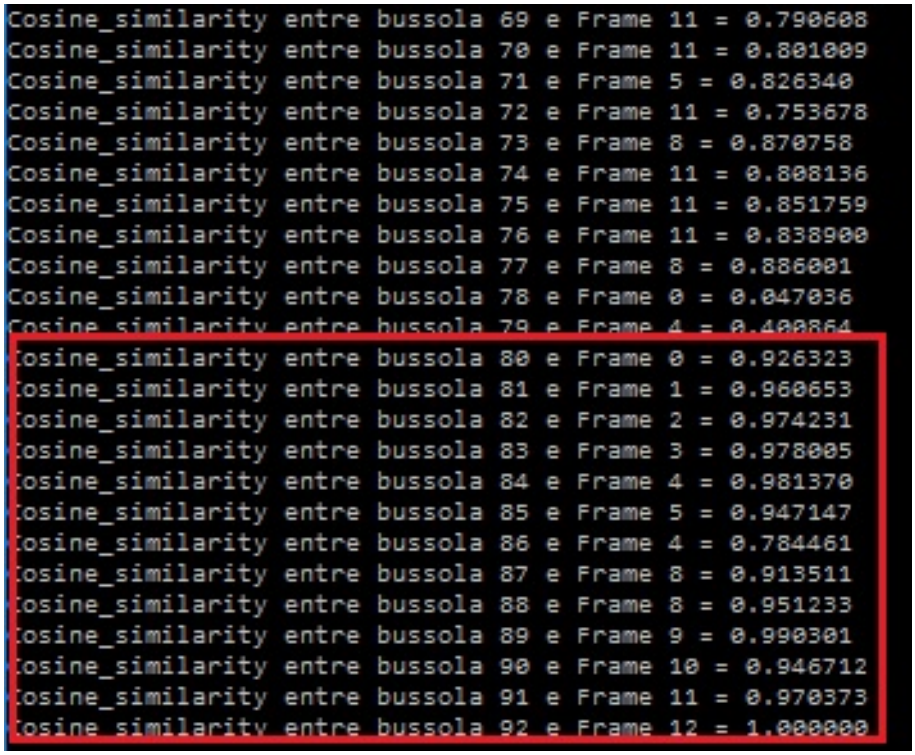


Figura 5.7 – Resultado do segundo teste de similaridade

5.3 Caso de Estudo 3: Ponto Fraco

O terceiro teste, ainda fazendo uso da Figura 5.5 como base, apresenta um dos pontos fracos da abordagem. As áreas destacadas em vermelho na Figura 5.8 são exatamente as mesmas visualmente, e para o cálculo de similaridade de cosseno isso também ocorre, resultando em inúmeras regiões de similaridade e por vezes, apresentando o ângulo errado como área de maior similaridade. Isso ocorre devido à ausência de troca de cores nestas regiões assim como a grande ocorrência destas em diferentes regiões da imagem.

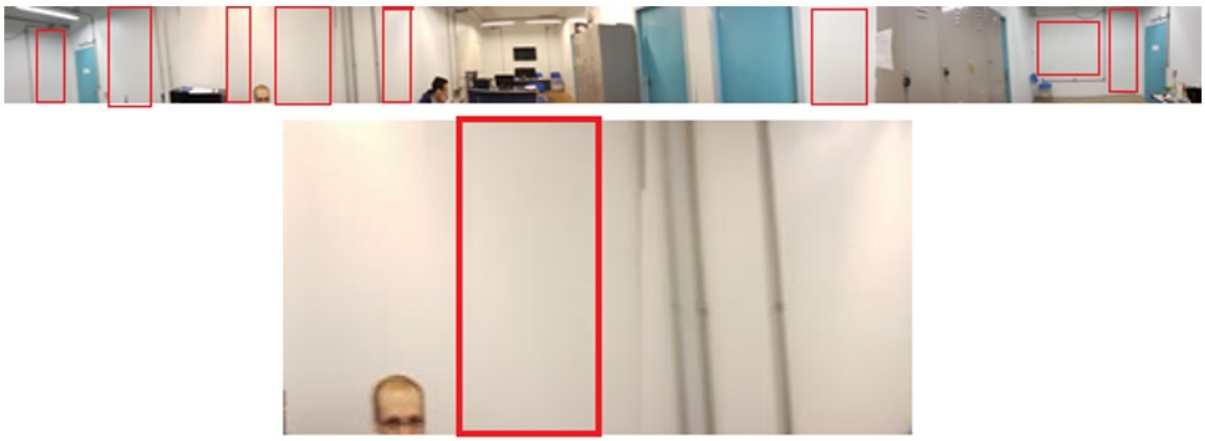


Figura 5.8 – Ponto fraco da abordagem

5.4 Caso de Estudo 4: Movimentação

O quarto teste é o que mais merece destaque. Foi retirada uma foto (*frame*) do mesmo ambiente da Figura 5.5, de uma região que não se repete na imagem em 360º. Esta imagem (Figura 5.9) foi retirada ao se mover da região central de onde foi extraída a imagem em 360º, sendo assim, alterando o ângulo da câmera, distância (zoom) e iluminação do local, inclusive com pessoas compondo a imagem, no qual se moveram e não estavam tal qual como na imagem original.

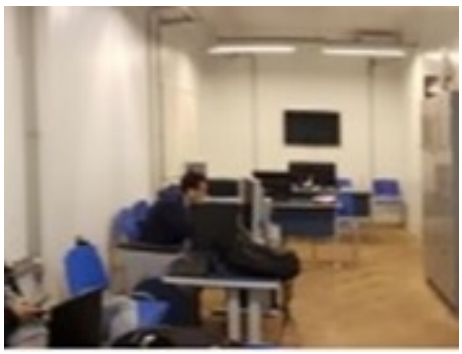


Figura 5.9 – Imagem utilizada no quarto teste

Apesar de não ter ocorrido uma sequência como nos testes 1 e 2, o algoritmo calculou corretamente a região de maior similaridade entre as duas imagens, tal como pode ser visto na Figura 5.10. A soma entre as similaridades entre uma matriz e suas matrizes adjacentes resultou em uma maior similaridade na área em destaque, e retornou como área mais similar a região de ângulo 152º da imagem em 360º.

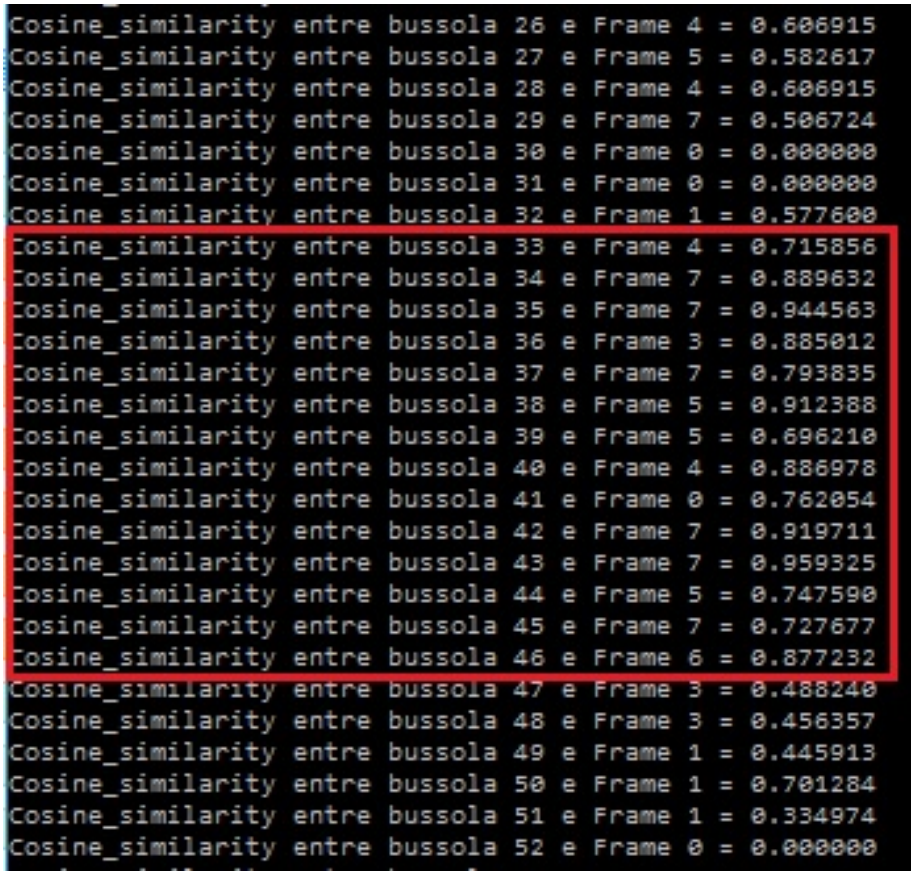


Figura 5.10 – Resultado do quarto teste

5.5 Caso de Estudo 5: Inúmeras áreas similares

O quinto teste, apresentado na Figura 5.11, faz uso de uma imagem de uma das portas presentes no ambiente, simulando a movimentação de um veículo ou robô autônomo, este último munido de movimentos de pescoço verticais e horizontais (*pan* e *tilt*), movimentos estes presentes em robôs humanoides (FUGALI; LIBRELOTTO, 2016). Estes movimentos alteram bastante os *frames* a serem analisados e podem diminuir drasticamente similaridade entre as imagens.

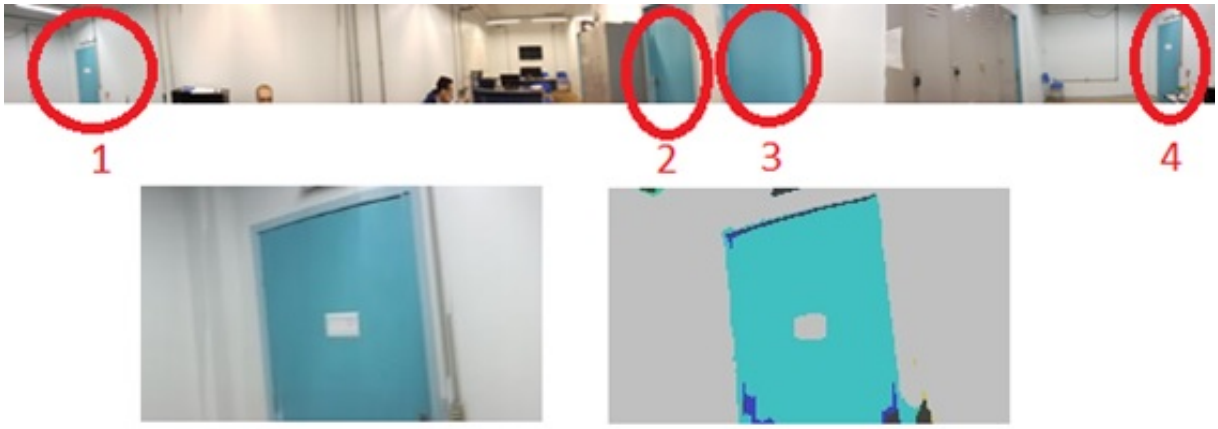


Figura 5.11 – Imagens do quinto teste

Assim como o ocorrido no segundo teste, o algoritmo encontrou mais de uma região de similaridade. Na Figura 5.11, podemos constatar com o olho humano, quatro possíveis regiões similares, porém, o algoritmo constatou similaridade na região 1 e 4, destacando a região 1. Embora as cores (azul) sejam similares, nas regiões 2 e 3 basicamente não ocorre troca de cores, característica utilizada na abordagem proposta por esse trabalho. Diferentes das regiões 1 e 4 onde ocorrem mudanças de cores quando a porta sede espaço à parede e posteriormente ao teto do ambiente.

5.6 Discussão dos Resultados

Os testes apresentados foram importantes para definir o contexto no qual a abordagem de bússola visual pretendida por este trabalho pode se tornar viável ou não. Como os cálculos são baseados nas trocas de cores que ocorrem em uma imagem, muitos ambientes heterogêneos não se tornam um contexto aconselhado, principalmente quando se trata de um ambiente aberto como o apresentado na Figura 5.12. Nota-se a enorme quantidade e falta de padrão nas trocas de cores devido a ocorrência de nuvens e árvores que "poluem"o ambiente analisado, mesmo após se aplicar as etapas de processamento de imagens apresentadas neste trabalho. Outros fatores envolta do mesmo exemplo, é o fato das nuvens e as árvores se moverem conforme o vento, e de o céu variar de cor conforme o horário do dia ou devido à fatores climáticos.

Figura 5.12 – Ambiente aberto no qual a bússola não apresentaria resultados satisfatórios

Em contrapartida, ambientes como o laboratório apresentado na Figura 5.5, apresentado em alguns dos testes, apresentaram resultados bastantes satisfatórios, mesmo com alguns problemas como a ausência de troca de cores e a repetição de algumas regiões na mesma imagem.

Em alguns casos, a movimentação do indivíduo autônomo pode acarretar mudanças na similaridade, o que pode ser contornado com a combinação de bússolas (mapeamento) do ambiente em diferentes posições. Um exemplo disso é o futebol de robôs humanoides, onde, antes de cada partida de futebol, pode ser feita uma perspectiva 360º do campo, de diferentes locais, tais como na Figura 4.7. O robô pode ser posicionado no centro de cada região (área amarela) e capturar as imagens para montar uma perspectiva 360º do ambiente, passo este denominado como calibração da bússola.

Além dos laços de repetição utilizados para percorrer as áreas de interesses (matrizes)

de ambas as imagens a serem analisadas, deve-se incluir um novo para percorrer as diferentes perspectivas em 360º. Com a combinação de várias perspectivas 360º, se tem maior confiabilidade no retorno da orientação correta, e pode ser utilizada para, além de prever a orientação, prever a localização do indivíduo autônomo.

5.7 Escolha do cálculo de similaridade

A escolha de algoritmos para calcular a similaridade entre imagens, normalmente segue por padrão, a escolha de algoritmos como a distância euclideana, bhattacharyya, mahalanobis, manhattan e entre outras. Estes algoritmos se dividem em dois grupos, os que calculam o coeficiente (similaridade) e a distancia (dissimilaridade), onde cada um possui suas próprias regras e condições no qual a sua utilização se torna viável ou não (TEKNOMO, 2015). Neste trabalho, após a extração das informações necessárias, foi construída uma matriz de números inteiros, o que abriu a possibilidade de aplicar outros algoritmos para se calcular a similaridade.

Alguns algoritmos foram testados e não apresentaram valores condizentes com o esperado, como foi o caso do coeficiente Jaccard, que assim como a similaridade de cosseno, retorna valores entre 0 e 1, onde quanto mais próximo a 1, mas similar são as matrizes; porem, este algoritmo não apresentava valores lógicos para as entradas recebidas, logo, foi concluído que este algoritmo é mais adequado para calcular distâncias (TEKNOMO, 2015); provavelmente devido à ocorrência de muitos zeros nas contagens de troca de cores e por não se trabalhar com valores binários, no qual a Jaccard se destaca.

Tan (2005) destaca a similaridade de cosseno para análise de documentos, apresentando vantagens comparado a algoritmos como distância euclideana e correlação de Pearson, devido a ocorrência de valores idênticos. Um exemplo é a grande ocorrência de contagens de troca de cores resultando em zero. Ao efetuar os testes com a similaridade de cosseno, esta técnica se apresentou promissora desde o inicio, fato que culminou na sua utilização.

5.8 Sumário do capitulo

Neste capitulo foram apresentados alguns testes efetuados no protótipo desenvolvido de bússola visual, com o intuito de apresentar seus pontos fortes e fracos, destacando as melhores situações no qual a bússola possa se tornar confiável. Os pontos fracos são salientados quando não ocorre troca de cores entre as imagens e também quando ocorrem inúmeras regiões

semelhantes dentro de uma mesma imagem.

Além disso, foram discutidos os resultados dos testes efetuados, e de maneira simplificada e objetiva, descrevendo os fatores necessário para o bom funcionamento desta abordagem de bússola visual.

6 TRABALHOS RELACIONADOS

Nesta seção são abordados os trabalhos relacionados a proposta apresentada neste trabalho, ou seja, trabalhos que fazem uso de câmera(s), processamento de imagens e visão computacional para estimar uma orientação.

6.1 Câmera omnidirecional

Um dos trabalhos trazidos por Labrosse (2006) afirma que informações de rotação tem a grande desvantagem de geralmente se tornando cada vez menos precisos conforme a rotação ou deslocamento de um robô, introduzindo erros em cada etapa. O autor então, faz uso de uma câmera omnidirecional acoplada a um robô para se utilizar extrair imagens panorâmicas acerca do ambiente à volta do robô. Labrosse faz uso de combinação de espaço de cor RGB e distância euclidiana, embora forneça bons resultados, pode não ser o suficiente em algumas situações, tais como quando as cores presentes armazenadas são mais fortes do que outros em termos de valor e não são uniformemente distribuídas, e também em ambientes que lembram um tabuleiro de xadrez. O autor salienta que sua abordagem pode ser melhorada, inclusive quanto ao custo computacional.

6.2 Redução de cores

Na proposta apresentada por Sturm e Visser (2009) faz uso de redução de cores para apenas 10 cores, com o intuito de facilitar o desenvolvimento para melhorar o desempenho. Então, separa-se os frames em colunas verticais com uma largura pré-definida e em cada uma dessas colunas, se faz uma procura por um *blob* de cores anteriormente definida e distribuída em áreas externas ao campo. Esta abordagem fica limitada à procura dessas marcações localizadas fora do campo como pode ser visto na Figura 6.1, tendo a certeza que estas cores não se repetiram. Ao encontrar o *blob*, efetua-se a contagem da troca de cores, e assim, estipula-se qual a orientação atual do robô. Sendo assim, a abordagem proposta não se preocupa em avaliar a similaridade entre imagens previamente armazenadas e as imagens provenientes da locomoção dos robôs, este trabalho também teve ênfase no futebol de robôs e possuiu resultados satisfatórios.

Além do trabalho de Sturm e Visser (2009), Paolieri (2011) também utiliza marcações

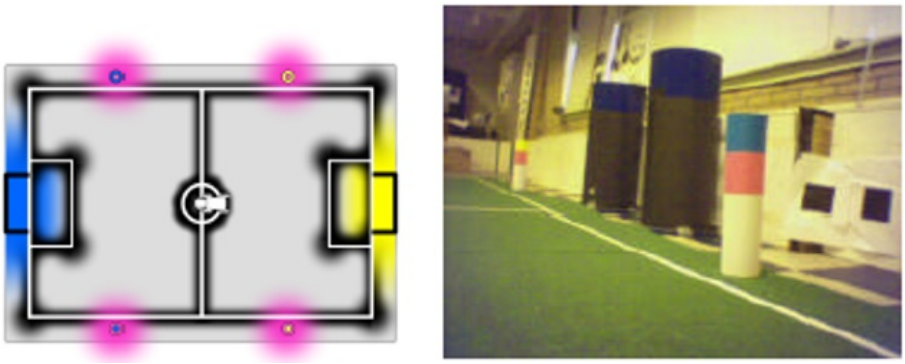


Figura 6.1 – Marcações (rosa) extra campo de Sturm e Visser (2009)

extracampo com o intuito de auxiliar e corrigir resultados recorrentes da odométrica em robôs.

A abordagem apresentada por de Kok (2013) é bastante similar com a proposta deste trabalho assim como a proposta anterior, porém com algumas diferenças na forma de implementação; onde o foco do trabalho foi o futebol de robôs, com o intuito de auxiliar os robôs na orientação durante uma partida. Por exemplo, os autores dividem o campo de futebol em uma matriz e posicionam os robôs em cada uma destas regiões resultantes, recolhendo frames (imagens) de todas as direções para cada região. A abordagem elaborada por de Kok (2013) faz uso de comunicação entre os robôs para trocar informações referentes aos frames retirados de cada região, sendo desnecessário que todos os robôs se posicionem em cada uma das regiões, tendo em vista que recebem as demais informações dos outros robôs da sua equipe. Os autores utilizam o espaço de cores YUV422 e reduzem as cores para apenas 8; e então extraem colunas com largura fixa, e geram uma matriz que contem a contagem de troca de cores para cada uma dessas colunas. Com as matrizes resultantes, então, é efetuada uma comparação com as imagens provenientes do robô em locomoção.

O principal foco de De Kok (2013) foi a atualização da bússola conforme a movimentação dos robôs na fase de mapeamento e a estipulação da localização, e não da orientação dos robôs, combinando a abordagem com um módulo que envolve o algoritmo de Monte Carlo. Os testes foram efetuas em um cenário artificial sem alterações e obteve ótimos resultados, mas de pouca importância devido ao cenário não sofrer nenhum tipo de alteração ou interferência, segundo os próprios autores.

6.3 Histograma de cores

A proposta de Bader (2013), também aplicada ao futebol de robôs faz uso de fatores extracampo para a orientação, removendo o campo dos frames para a análise. São gravados frames de todas as direções a partir do centro do campo, gerando, assim, uma perspectiva 360º, e com estas informações é construído um muro virtual dividido em quadrados. Em cada um destes quadrados, é armazenado o histograma de cores para modelar o fundo do muro. Bader compara os histogramas do seu muro com novos histogramas gerados dos novos frames provenientes da locomoção do robô no decorrer de uma partida. A Figura 6.2 mostra a representação da perspectiva 360º proposta por Bader. Os testes foram efetuados em ambiente simulado e real com o objetivo de mover o robô em locais pré-definidos pelas regras do jogo (tais como localização inicial, localização de penalidade), no qual chegou-se na conclusão que várias questões se mantiveram em aberto, por exemplo, uma contagem ideal de histogramas de cor, uma recurso diferente para ambientes de modelagem, tais como informações gradiente, a forma no qual o muro virtual foi utilizado, e a implementação de um padrão de pesquisa ideal para forçar o robô a olhar para áreas com o fundo mais distintiva.

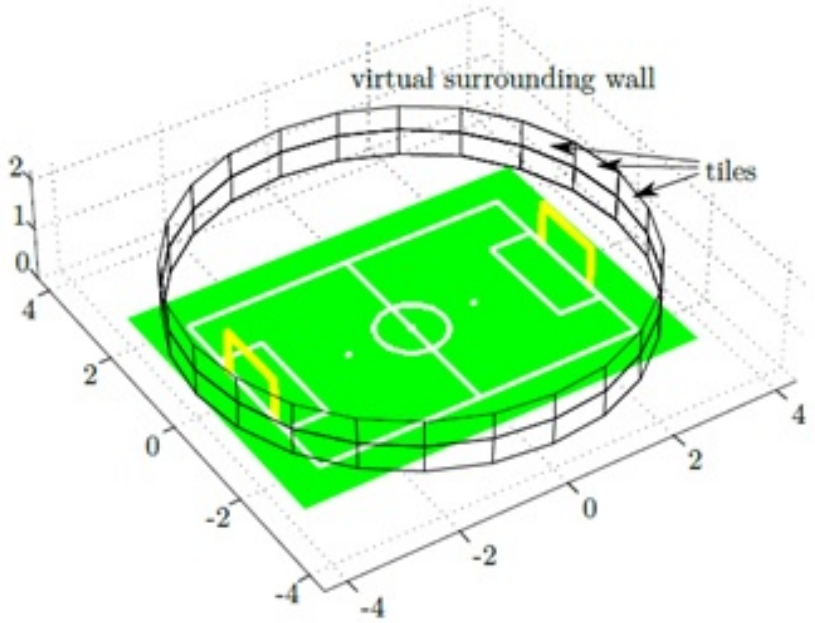


Figura 6.2 – Representação da perspectiva 60º proposta por Bader (2013)

6.4 SLAM (Simultaneous Localization and Mapping)

Outra abordagem possível, é a utilização da técnica de SLAM (Simultaneous Localization and Mapping no inglês e Localização e Mapeamento Simultâneos no português), uma técnica utilizada em veículos autônomos e na robótica em geral, no qual se constrói um mapa de um ambiente com o intuito de estipular a localização do robô. Não sendo destinada a perfeição, a gama de abordagens é bastante significativa, como em veículos terrestres e aéreos não tripulados, robôs domésticos, submarinos e na medicina, no corpo humano. Dentre os principais problemas, estão a imprecisão, o alto custo computacional e a grande complexidade de implementação. Um dos principais fatores de imprecisão, é a locomoção de pedestres, veículos e demais objetos que fazem parte do cenário (MONTIEL; DAVISON, 2006) (HAN; LEE; JI, 2010).

6.5 Comparação entre os trabalhos

Todos os trabalhos apresentados possuem limitações, tanto quanto à implementação ou as abordagens são situacionais, ou seja, resolvem o problema em algumas situações e não em outras, e nenhum dos casos foi apresentado uma solução genérica para resolver o problema de orientação em ambientes de forma autônoma, sempre vinculado à algum contexto específico. Baseado nisso, a proposta deste trabalho se diferencia por desenvolver uma bússola visual genérica que seja confiável ou o mais perto possível disto, levando em consideração aspectos positivos dos trabalhos relacionados.

Tabela 6.1 – Trabalhos relacionados

Trabalho	Livre de Contexto	Baseado em Cores	Futebol de Robôs	Livre de Marcações
Londero	X	X	X	X
Labrosse	-	X	-	-
Sturm e Visser	-	X	X	X
De Kok	-	X	X	-
Paolieri	-	-	-	-
Bader	X	X	X	X

Na Tabela 7.1 pode-se ver a relação entre as características do presente trabalho com os principais trabalhos relacionados. Embora a grande maioria seja baseado em análise de cores, cada um segue uma forma diferente de utilizar as cores em prol do seu trabalho; Bader (2013) por exemplo utiliza histogramas, enquanto Labrosse (2006) faz uso de análise de intensidade

e distribuição de cores; o presente trabalho, o de Sturm (2009) e De Kok (2013) utilizam a contagem de troca de cores, mas os dois últimos citados possuem maior foco em localização e não são livres de contexto , por se limitarem ao futebol de robôs e serem dependentes de marcações fora do campo. Os trabalhos que envolvem a técnica de SLAM não foram incluída na tabela devido a serem aplicados em trabalhos que envolve mapeamento e localização de ambientes.

7 CONSIDERAÇÕES FINAIS

Este capítulo apresenta a conclusão sobre a pesquisa realizada e considerações sobre a utilização da abordagem proposta, finalizando a pesquisa sobre a utilização de bússola visual para estipular a orientação em indivíduos autônomos. Por fim, relata-se sobre trabalhos futuros relacionados ao tema.

A motivação deste trabalho se deu pela baixa quantidade de trabalhos que envolvam este assunto, assim como o objetivo de estudar melhores abordagens para se utilizar no futebol de robôs humanoides junto a equipe de competição e pesquisa TAURA. Sendo assim, obteve-se conhecimento acerca de técnicas de processamento de imagens e visão computacional assim como a biblioteca OpenCV; com o intuito de utilizar uma webcam como ferramenta para estipular a orientação dentro de um ambiente/contexto para veículos ou robôs autônomos, fazendo uso das cores presentes nas imagens.

Os testes efetuados demonstraram a utilização da técnica de contagem das trocas de cores de colunas extraídas das imagens apresenta resultados satisfatórios com as etapas de processamento de imagens descritas neste trabalho. Assim como a aplicação do cálculo de similaridade de cosseno para comparação entre as matrizes de troca de cores mostra-se totalmente adequado à essa funcionalidade até então não aplicada em nenhum trabalho relacionado.

O porém desta abordagem fica relacionado a imagens onde não ocorram troca de cores em muitas colunas assim como regiões que venham a se repetir na imagem de perspectiva 360º armazenada para análise de orientação. Também quando ocorre movimentação ou afastamento da área central da perspectiva 360º, o que diminui a similaridade entre as imagens, mas que é facilmente contornado com a combinação de mais de uma perspectiva 360º do mesmo ambiente; fato este que pode estender a abordagem para, além de prever a orientação, seja capaz de estimar a localização do veículo ou robô autônomo.

A metodologia apresentada é descrita com todos os passos necessários para que esta abordagem seja replicada e possivelmente melhorada em trabalhos futuros. Uma das possibilidades é acrescentar a Inferência Bayesiana no qual com o passar do tempo e as alterações ocorridas no cenário, pode-se prever a quantidade de incertezas acarretadas. Outro ponto possível a ser trabalhado futuramente é a questão de manter atualizado as perspectivas 360º do cenário ao longo do tempo. Fica incluído nos trabalhos futuros, aplicar a abordagem de bússola visual deste trabalho em um veículo ou robô autônomo real, fato este que não interferiria

diretamente nos resultados obtidos. Em alguns casos, as técnicas de processamento de imagens utilizadas com o intuito de se remover ruídos e aumentar a homogeneidade da imagem, possam ser alterados conforme o contexto que a bússola for aplicada, aumentando ou reduzindo o tamanho do elemento estruturante das transformações morfológicas aplicadas por exemplo.

REFERÊNCIAS

AKHTARUZZAMAN, M.; SHAFIE, A. A. **Evolution of Humanoid Robot and Contribution of Various Countries in Advancing the Research and Development of the Platform**. [S.l.]: International Conference on Control, Automation and Systems, 2010.

ANDERS, B. et al. WF Wolves Taura Bots – Humanoid Teen Size Team Description for RoboCup 2016. , [S.l.], 2016.

BADER, M.; PRANKL, J.; VINCZE, M. Visual Room-Awareness for Humanoid Robot Self-Localization. **arXiv preprint arXiv:1304.5878**, [S.l.], 2013.

BALTES, J. et al. **RoboCup Humanoid League Rule Developments 2002-2014 and Future Perspectives**. João Pessoa, Brasil: [s.n.], 2014.

BPMN. Notation (BPMN) version 2.0. **OMG Specification, Object Management Group**, [S.l.], 2011.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**: computer vision with the opencv library. [S.l.]: "O'Reilly Media, Inc.", 2008.

CALXITO, E. P. Análise Comparativa da Influência da Forma do Elemento Estruturante em Imagens. **Universidade Federal Fluminense. Pós Graduação em Ciência da Computação**, [S.l.], 2003.

CONCI, A. et al. Utilização de Imagens Monocromáticas com Grãos Sobrepostos na Avaliação de Geometria dos Meios Porosos. **Proc. of DINCON**, [S.l.], 2004.

DAWSON-HOWE, K. **A practical introduction to computer vision with opencv**. [S.l.]: John Wiley & Sons, 2014.

FACON, J. A Morfologia Matemática e suas Aplicações em Processamento de Imagens. , [S.l.], 2011.

FUGALI, F.; LIBRELOTTO, G. specificação de Comportamento de Futebol para Robôs Utilizando Máquina de Estados e Ontologia. **Universidade Federal de Santa Maria. Pós Graduação em Ciência da Computação**, [S.l.], 2016.

GERNDT, R. et al. **Humanoid Robots in Soccer**: robots versus humans in robocup 2050. Endereço: Digital Object Identifier, 2015. (IEEE Robotics & Automation Magazine).

GONZALEZ, R. C.; RICHARD, E. Woods, digital image processing. **ed: Prentice Hall Press, ISBN 0-201-18075-8**, [S.l.], 2002.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. Digital image using Matlab processing. **Person Prentice Hall, Lexington**, [S.l.], 2004.

HA, I. et al. **Development of open humanoid platform DARwIn-OP**. [S.l.]: In SICE Annual Conference (SICE), 2011.

HAN, J.-S.; LEE, S.-M.; JI, S.-H. A Visual Compass based on UKF SLAM. , [S.l.], 2010.

KITANO, H. et al. Robocup: a challenge problem for ai and robotics. In: **RoboCup-97**: robot soccer world cup i. [S.l.]: Springer, 1998. p.1–19.

KOK, P. M. de; METHENITIS, G.; NUGTEREN, S. ViCTOriA: visual compass to orientate accurately. , [S.l.], 2013.

LABROSSE, F. et al. The visual compass: performance and limitations of an appearance-based method. **Journal of Field Robotics**, [S.l.], v.23, n.10, p.913, 2006.

LAGANIÈRE, R. **OpenCV 2 Computer Vision Application Programming Cookbook**: over 50 recipes to master this library of programming functions for real-time computer vision. [S.l.]: Packt Publishing Ltd, 2011.

MARENGONI, M.; STRINGHINI, S. Tutorial: introdução à visão computacional usando opencv. **Revista de Informática Teórica e Aplicada**, [S.l.], v.16, n.1, p.125–160, 2009.

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

MELCHIADES, F. G.; BOSCHI, A. O. Cores e tonalidades em revestimentos cerâmicos. **Cerâmica Industrial**, [S.l.], v.4, n.1-6, p.1–6, 1999.

MONTENEGRO, F. J. C. **Criação de um Simulador para Auxiliar no Desenvolvimento de Estratégias de Comportamento para Futebol de Robôs**. UFSM: UFSM, 2015. v.Trabalho de Conclusão de Curso.

MONTIEL, J.; DAVISON, A. J. A visual compass based on SLAM. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2006. ICRA 2006., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1917–1922.

OPENCV. **OpenCV**. Disponível em <opencv.org/>, Acesso em: 12 dez. 2015.

ORLANDINI, G.; MARTINS, L. E. Desenvolvimento de Aplicativos Baseados em Técnicas de Visão Computacional para Robô Móvel Autônomo. 2012. , [S.l.], 2012. Disponível em <http://www.xbot.com.br/wp-content/uploads/2012/10/Dissertação$_G uilherme_V 4_r evisado_p os_d efesa - 2_c orrigido_f inal.pdf / >$.

PAOLIERI, F. et al. Correcao de Odometria de Robos Moveis Usando Visao Computacional. **Anais do Computer on the Beach**, [S.l.], p.41–50, 2011.

PENHARBEL, E. A. et al. Filtro de imagem baseado em matriz RGB de cores-padrão para futebol de robôs. **Submetido ao I Encontro de Robótica Inteligente**, [S.l.], 2004.

RoboCup Humanoid League Technical Committee. [S.l.]: RoboCup Soccer Humanoid League rules and setup for the 2015 competition, 2015.

RoboCup League Technical Committee. [S.l.]: RoboCup Soccer League rules and setup for the 2015 competition, 2015.

ROMANO, V. F. **Robótica Industrial**: aplicação na indústria de manufatura e de processos. São Paulo, SP: Edgard Blucher LTDA, 2002. v.1º.

SANTOS, D. F. D. d. Recuperação de imagens: similaridade parcial baseada em espectro de grafo e cor. , [S.l.], 2012.

SANTOS, F. L.; NASCIMENTO, F. M. S.; BEZERRA, R. M. S. **REDUC**: a robótica educacional como abordagem de baixo custo para o ensino de computação em cursos técnicos e tecnológicos. Salvador: Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA), 2010.

SCURI, A. E. Fundamentos da imagem digital. **Pontifícia Universidade Católica do Rio de Janeiro**, [S.l.], 1999.

SHAPIRO, L.; STOCKMAN, G. C. Computer vision. 2001. **ed: Prentice Hall**, [S.l.], 2001.

STURM, J.; VISSER, A. An appearance-based visual compass for mobile robots. **Robotics and Autonomous Systems**, [S.l.], v.57, n.5, p.536–545, 2009.

TAN, P. **MS and Kumar, V.**: introduction to data mining. [S.l.]: Addison Wesley, 2005.

TEKNOMO, K. Similarity Measurement. , [S.l.], 2015. Disponível em <http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html/>.

ULLRICH, R. A. **Robótica - Uma Introdução - O porquê dos robôs e seu papel no trabalho**. Rio de Janeiro: Ed. Campus, 1987.

VALERO, R. et al. **Cirugía robótica**: historia e impacto en la enseñanza. [S.l.]: Elsevier, 2011. 540-545p.

APÊNDICES

APÊNDICE A – Implementação da Similaridade de Cosseno

Neste apêndice é apresentado o código fonte desenvolvido em C++ que efetua o cálculo de similaridade entre as matrizes de cores da imagem referente a bússola com as imagens de um robô ou veículo em movimentação.

```
1
2  printf("Calculo de similaridade - cosine_similarity\n\n");
3  int bussolaCol = 0;
4  int newCol = 0;
5
6  for (std::list<A>::iterator it = MatrizesBussola.begin(); it !=
       MatrizesBussola.end(); ++it)
7  {
8          double similaridadeAnterior = 0;
9          int colunaMaisSimilar = 0;
10
11         for (std::list<A>::iterator itN = MatrizesNew.begin(); itN !=
               MatrizesNew.end(); ++itN)
12         {
13                 double similaridade = 0.0;
14                 double denom_a = 0.0, denom_b = 0.0;
15
16                 for (int j = 0; j < 8; j++)
17                 {
18                   for (int i = 0; i < 8; i++)
19                   {
20                     similaridade += it->Matriz[i][j] * itN->Matriz[i
                           ][j];
21                     denom_a += it->Matriz[i][j] * it->Matriz[i][j];
22                     denom_b += itN->Matriz[i][j] * itN->Matriz[i][j];
23                   }
24                 }
25
26                 similaridade = similaridade / (sqrt(denom_a) * sqrt(
                       denom_b));
27
28                 if (similaridade > similaridadeAnterior)
29                 {
30                   similaridadeAnterior = similaridade;
31                   colunaMaisSimilar = newCol;
32                 }
33
34                 newCol++;
35         }
36
37         printf("Cosine_similarity entre bussola %d e Frame %d = %f \n
               ", bussolaCol, colunaMaisSimilar, similaridadeAnterior);
38
```

```
39          bussolaCol++;
40          newCol = 0;
41  }
```

APÊNDICE B – Leitura e processamento de imagem

O código fonte abaixo é responsável pela leitura e pelo processamento de imagens da bússola visual desenvolvida. Na linha 8 é criado um elemento estruturante que é imediatamente utilizado em uma transformação morfológica de abertura.

```
1
2  cv::Mat ProcessaImagem()
3  {
4          cv::Mat image;
5          image = lerImage();
6
7          int erosion_size = 2; // padrao    2
8          Mat element = getStructuringElement(cv::MORPH_OPEN,
9                  cv::Size(2 * erosion_size + 1, 2 * erosion_size + 1),
10                 cv::Point(erosion_size, erosion_size));
11
12         colorReduce(image);
13
14         return image;
15 }
16
17 cv::Mat lerImage()
18 {
19         cv::Mat image;
20         image = cv::imread("360_2_mini.jpg", CV_LOAD_IMAGE_COLOR);
21         //cv::flip(image, image, -1);
22
23         return image;
24 }
```

APÊNDICE C – Listas e estruturas utilizadas

```
1
2  struct A
3  {
4          int Matriz[8][8];
5  };
6
7  struct Correlacoes
8  {
9          int correlacao;
10         int colBuss;
11         int colNew;
12 };
13
14 struct CorrelacoesSoma
15 {
16         int soma;
17         int de;
18         int ate;
19         int colunaBussola;
20         int colunaNew;
21 };
22
23 struct CossineOrder
24 {
25         int bussola;
26         int ColNew;
27         double valor;
28 };
29
30 bool sort_by_soma(const CorrelacoesSoma & cm, const CorrelacoesSoma &
       cm2)
31 {
32         return cm.soma < cm2.soma;
33 }
34
35 std::list<A> MatrizesBussola;
36 std::list<A> MatrizesNew;
```

APÊNDICE D – Destaque das áreas de interesse

Este código foi utilizado para destacar as áreas de interesses, ou seja, as colunas de 10 *pixeis* de largura utilizadas para montar as matrizes de trocas de cores necessárias para o calculo de similaridade.

```
1
2  for (int i = 0; i < aux.rows; i++)
3  {
4          for (int x = 0; x < 10; x++)
5          {
6                  for (int j = x; j < aux.cols; j = j + 20)
7                  {
8                    Vec3b bgrPixel = aux.at<Vec3b>(i, j);
9
10                   if (i == 0 && j % 10 == 0)
11                   {
12                     cv::rectangle(aux, cvRect(j, i, 10, aux.rows),
                          CvScalar(0, 128, 255), 1, 1, 0);
13                   }
14
15                   try
16                   {
17                     if (bgrPixel.val[0] < 200)
18                       bgrPixel.val[0] = bgrPixel.val[0] + 50; // B
19                     if (bgrPixel.val[1] < 200)
20                       bgrPixel.val[1] = bgrPixel.val[1] + 50; // G
21                     if (bgrPixel.val[2] < 200)
22                       bgrPixel.val[2] = bgrPixel.val[2] + 50; // R
23
24                     aux.at<Vec3b>(i, j) = bgrPixel;
25                   }
26                   catch (int e)
27                   {
28                     continue;
29                   }
30                 }
31         }
32 }
```

APÊNDICE E – Contagem das trocas de cores

Este apêndice apresenta o código utlizado responsável por efetuar a contagem de troca de cores para cada uma das áreas de interesses.

```
1
2  std::list<MyPixelInfo> lista;
3  int columnName = 0;
4  for (int i = 0; i < imagemColuna; i++)
5  {
6          char arquivo[26];
7          _snprintf(arquivo, 26, "%d trocas.txt", i);//arquivo que
               cont m as trocas de cores ocorridas
8          std::ifstream file(arquivo);
9          if (!file)
10         {
11                 char arquivoNovo[26];
12                 _snprintf(arquivoNovo, 26, "histograma %d.txt",
                       columnName);//arquivo que representa o histograma
                       das trocas de cores (contagem de cada troca de cor
                       )
13                 std::ofstream newfile;
14                 newfile.open(arquivoNovo, std::ios_base::app);
15
16                 for (std::list<MyPixelInfo>::iterator it = lista.
                       begin(); it != lista.end(); ++it)
17                 {
18                         newfile << it->origin_red;
19                         newfile << " ";
20                         newfile << it->origin_green;
21                         newfile << " ";
22                         newfile << it->origin_blue;
23                         newfile << " para ";
24                         newfile << it->destiny_red;
25                         newfile << " ";
26                         newfile << it->destiny_green;
27                         newfile << " ";
28                         newfile << it->destiny_blue;
29                         newfile << " : ";
30                         newfile << it->count;
31                         newfile << "\n";
32                 }
33                 newfile.close();
34                 i += 9;
35                 columnName += 1;
36                 lista.clear();
37                 file.close();
38                 continue;
39         }
40
```

```
41          if (file.is_open())
42          {
43                  while (!file.eof())
44                  {
45                          std::string line;
46                          std::getline(file, line);
47                          std::vector<std::string> splitted_vector =
                               split(line);
48                          if (splitted_vector.size() < 8)
49                                  break;
50                          MyPixelInfo mpi;
51                          mpi.origin_red = std::stoi(splitted_vector
                               [0]);
52                          mpi.origin_green = std::stoi(splitted_vector
                               [1]);
53                          mpi.origin_blue = std::stoi(splitted_vector
                               [2]);
54                          mpi.destiny_red = std::stoi(splitted_vector
                               [4]);
55                          mpi.destiny_green = std::stoi(splitted_vector
                               [5]);
56                          mpi.destiny_blue = std::stoi(splitted_vector
                               [6]);
57                          mpi.count = 1;
58
59                          if (lista.empty())
60                          {
61                                  lista.push_back(mpi);
62                          }
63                          else
64                          {
65                                  bool found = false;
66                                  for (std::list<MyPixelInfo>::iterator
                                       it = lista.begin(); it != lista.
                                      end(); ++it)
67                                  {
68                                          if (mpi.origin_red == it->
                                              origin_red &&
69                                                  mpi.origin_green ==
                                                      it->origin_green
                                                      &&
70                                                  mpi.origin_blue == it
                                                      ->origin_blue &&
71                                                  mpi.destiny_red == it
                                                      ->destiny_red &&
72                                                  mpi.destiny_green ==
                                                      it->destiny_green
                                                      &&
73                                                  mpi.destiny_blue ==
                                                      it->destiny_blue)
74                                          {
```

```
75                                                      found = true;
76
77                                                      it->count = it->count
                                                            + 1;
78
79                                                      break;
80                                              }
81                                      }
82
83                                      if (!found)
84                                      {
85                                              lista.push_back(mpi);
86                                      }
87                              }
88                      }
89              }
90      }
```

APÊNDICE F – Montagem das matrizes de cores

Código utilizado para a montagem da matriz de cores, posteriormente utilizada no cálculo de similaridade.

```

for (int i = 0; i < countColunas; i++)
{
        char arquivo[26];
        _snprintf(arquivo, 26, "histograma %d.txt", i);

        std::ifstream file(arquivo);

        if (!file)
        {continue;}

        if (file.is_open())     {
/*0 - amarelo - 192 192 64
                1 - cinza - 64 64 64
                2 - vermelho - 192 64 64
                3 - azul - 64 64 192
                4 - verde - 64 192 64
                5 - rosa - 192 64 192
                6 - branco - 192 192 192
                7 - ciano - 64 192 192
                */
                A MatrizBussola;

                for (int x = 0; x < 8; x++)
                {
                        for (int j = 0; j < 8; j++)
                        {
                                MatrizBussola.Matriz[x][j] = 0;
                        }
                }
                while (!file.eof())
                {
                        std::string line;
                        std::getline(file, line);

                        std::vector<std::string> splitted_vector =
                           split(line);
                        //La os e condicionais para gerar as
                           matrizes
.
.
.
                //mostra as matrizes
                for (int x = 0; x < 8; x++)
                {
```

```
44                          for (int j = 0; j < 8; j++)
45                          {
46                                  printf("[%d]", MatrizBussola.Matriz[x
                                       ][j]);
47                          }
48
49                          printf("\n");
50                  }
51
52                  MatrizesBussola.push_back(MatrizBussola);
53
54                  printf("\n");
55          }
```