apter]quadroloqQuadro

FACULDADES DOCTUM DE IPATINGA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

THAYRONE MARQUES SILVA
VINÍCIUS ANDRADE LOPES

Visão computacional aplicada no reconhecimento de jogadores de futebol americano

IPATINGA, MG

2019

THAYRONE MARQUES SILVA
VINÍCIUS ANDRADE LOPES

Unidade Ipatinga - MG

SISTEMAS DE INFORMAÇÃO

Visão computacional aplicada no reconhecimento de jogadores de futebol americano

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Sistemas de Informação das Faculdades Doctum de Ipatinga - Rede de Ensino Doctum, como requisito parcial para a obtenção do título de bacharel em Sistemas de Informação.

Prof.(a) Orientador(a): Esp. Tales Wallace Souza

Área de concentração: Sistemas de Informação, Visão Computacional. Processamento de Imagem.

Ipatinga
2019

THAYRONE MARQUES SILVA
VINÍCIUS ANDRADE LOPES
Visão computacional aplicada no reconhecimento de jogadores de futebol americano/ THAYRONE MARQUES SILVA
VINÍCIUS ANDRADE LOPES. – IPATINGA, MG, 2019-
79p. : il. (algumas color.) ; 30 cm.

Prof.(a) Orientador(a): Esp. Tales Wallace Souza

Trabalho de Conclusão de Curso – FACULDADES DOCTUM DE IPATINGA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO, 2019.

1. Visão Computacional. 2. Processamento de Imagem. 3. Reconhecimento. I. Esp. Tales Wallace Souza. II. Faculdades DOCTUM de Ipatinga. III. Trabalho de Conclusão de Curso

THAYRONE MARQUES SILVA
VINÍCIUS ANDRADE LOPES

Visão computacional aplicada no reconhecimento de jogadores de futebol americano

Este Trabalho de Conclusão de Curso foi julgado e aprovado, como requisito parcial a obtenção do título de bacharel em Sistemas de Informação nas Faculdades Doctum de Ipatinga – Rede Doctum de Ensino, em 2019.

Média Final: _____

Ipatinga, XX de dezembro de 20XX

Banca Examinadora

Prof. Orientador: Esp. Tales Wallace Souza
MBA em Gerenciamento de Projetos Doctum
Instituto Superior Doctum de Ipatinga

Prof.ª Convidada: Maíza Cristina de Souza Dias
Mestre em Informática PUC - Minas
Instituto Superior Doctum de Ipatinga

Prof. Convidado: Marcelo José Vigorito Campara
Mestre em Sistemas de Informação e Gestão do Conhecimento - Universidade FUMEC
Instituto Superior Doctum de Ipatinga

DEDICATÓRIA

Dedicamos este trabalho primeiramente a Deus que nos sustentou durante toda a jornada.
A nossos familiares e entes queridos, que acreditaram desde o início em nós.

AGRADECIMENTOS

Agradecemos primeiramente a Deus que esteve conosco em todas as tribulações de nossas vidas, nos ajudando a superá-las.

Aos nossos pais, por todo o amor que nos deram, além da educação, ensinamentos e apoio. A nossos entes queridos que estiveram conosco durante toda esta caminhada.

A todo o corpo docente da Faculdade DOCTUM de Ipatinga que esteve presente em todos os momentos de nossos estudos, em especial para o nosso professor orientador Tales Wallace Souza, que nos auxiliou com todo seu profissionalismo para o desenvolvimento deste trabalho e a Maíza Cristina de Souza Dias, que contribuiu com seu conhecimento e especialidade sobre o assunto decorrente deste trabalho.

E finalmente, mas não menos importante, a todos os integrantes da turma de sistemas de informação, que percorreram toda esta jornada conosco.

Quando todos nos unirmos contra as injustiças e em defesa da privacidade e dos direitos humanos básicos, poderemos nos defender até dos mais poderosos dos sistemas.

(*Edward Snowden*)

RESUMO

A visão computacional vem sendo utilizada em larga escala, sendo difundida em várias áreas no mercado atual. Indústria e áreas desportivas são exemplos de áreas que têm apresentado um crescimento expressivo devido as soluções empregadas pela tecnologia citada. No entanto, a implantação deste tipo de tecnologia requer um conhecimento avançado sobre esta e também sobre as regras de negócios que serão aplicadas para a resolução de algum problema em específico. Como consequência disto, o investimento em *hardwares* que são capazes de realizar altos níveis de processamento também deve ser levando em consideração, visto que um computador básico não tem potência suficiente para realizar análises de imagem em tempo real. Sendo assim, esta pesquisa exploratória tem a finalidade de apresentar, de maneira didática, o funcionamento da tecnologia de visão computacional aplicada no esporte futebol americano onde, a princípio, foi proposto o reconhecimento de um dos jogadores dentro de campo.

Palavras-chave: Visão computacional. Processamento de imagem. Reconhecimento. Tecnologia.

ABSTRACT

Computer vision has been used on a large scale and is widespread in many areas of the current market. Industry and sports are examples of areas that have shown significant growth due to the solutions employed by the technology mentioned. However, deploying this type of technology requires advanced knowledge of this technology as well as the business rules that will be applied to solve any particular problem. As a consequence of this, the investment in hardware that is capable of performing high processing levels must also be taken into consideration, as a basic computer does not have enough power to perform real-time image analysis. Thus, this exploratory research has the purpose of presenting, in a didactic way, the operation of computer vision technology applied in the football sport where, at first, the recognition of one of the players on the field was proposed.

Keywords: Computer Vision. Image Processing. Recognition. Technology.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2-D	Duas dimensões
API	*Application Programming Interface*
APS	*Active Pixel Sensor*
BSD	*Berkeley Software Distribution*
CCD	*Charge Coupled Devic*
CIE	Comissão Internacional de Iluminação
CMOS	*Complementary Metal Oxide Semiconducto*
ESPN	*Entertainment and Sports Programming Network*
IDE	*Integrated Development Environment*
MOS	*Metal Oxide Semiconductor*
NASA	*National Aeronautics and Space Administration*
NFL	*National Football League*
OpenCV	*Open Source Computer Vision Library*
RCS	*Revision Control System*
RGB	*Red (Vermelho), Green (Verde) e Blue (Azul)*
SCM	*Source Code Manager*
VAR	*Video Assistant Referee*
VCS	*Version Control System*

SUMÁRIO

1 INTRODUÇÃO

A evolução tecnológica expandiu consideravelmente nos dias atuais, proporcionando um avanço descomunal de vários hardwares poderosíssimos capazes de realizar tarefas jamais vistas pelos gênios antigos da computação. Devido a isso, a imersão de varias técnicas vem sendo estudadas progressivamente nos últimos anos, visto que atualmente existem equipamentos capazes de realizá-las em um curto prazo de tempo e de maneira mais eficaz.

A área de visão computacional se tornou de grande interesse nos tempos atuais devido a sua complexidade ao realizar processamentos de imagens e extrair o maior número de informações desta. O campo de processamento de imagem complementa esta parte, utilizando técnicas matemáticas e probabilísticas para corrigir os parâmetros visuais da imagem que será processada.

Por mais que existam alguns problemas nesta área como, por exemplo, a captura de imagens, processamento, alta precisão, similaridade, dentre outros, os estudos buscam aprimorar a ferramenta para que o campo de visão computacional se aproxime ou até ultrapasse, em algumas situações, à eficiência da uma visão humana.

O futebol americano é um esporte bastante popular nos Estados Unidos da América (EUA), na frente ate mesmo de grandes esportes como, por exemplo, *baseball*, *basketball* e *football*. Em 1869, nos EUA, aconteceu o primeiro jogo de futebol americano no mundo, onde a Universidade de *Princeton* recebia o time da Universidade de *Rugters*. Na época, o futebol americano ainda estava tomando forma, sem muitas regras e muito confuso. Mesmo assim, o esporte se popularizou, principalmente dentre os universitários (RODRIGUES et al., 2014). No Brasil o esporte tem se popularizado nos últimos anos devido ao fato das redes de TV por assinatura transmitirem o evento em larga escala (subseção 2.1.2).

1.1 Objetivo geral

Apresentar uma ferramenta que seja capaz de identificar um jogador de futebol americano dentro de campo. Para isso, a identificação será feita através da ferramenta na qual será desenvolvida utilizando visão computacional e a biblioteca de processamento de imagens *OpenCV*.

1.2 Objetivos específicos

- Classificar as imagens de um jogador de futebol americano e extrair o maior numero de informações.
- Elaborar um modelo de busca com o conjunto de informações processadas sobre jogadores de futebol americano.
- Treinar o algoritmo seguindo as configurações do modelo de busca.
- Capturar, através de um dispositivo de entrada de vídeo, as imagens de uma partidade futebol americano.
- Identificar o jogador seguindo o modelo de busca.
- Analisar o percentual de acertos e erros da ferramenta.

1.3 Justificativa

A área de visão computacional tem crescido de forma significativa no mundo presente, devido ao avanço tecnológico. Várias informações são capturadas e o volume de dados encontram-se crescendo progressivamente. Sendo assim, inumeras aplicações que utilizam a tecnologia de visão computacional estão sendo desenvolvidas para auxiliar diversas áreas, como por exemplo a medicina, segurança e esporte.

Este crescimento ocorre principalmente devido ao aumento da utilização de dispositivos móveis. Não é difícil se deparar com vários equipamentos que já utilizam essa tecnologia para alguma função, seja ela para melhorar algo, seja para realizar a identificação de algum objeto ou biometria de segurança, como por exemplo o *Face ID* (Identidade de Rosto) da *Apple* e o *Google Lens*.

Sendo assim, a visão computacional pode auxiliar na questão relacionada ao reconhecimento de um jogador em uma partida de futebol americano.

As jogadas realizadas em futebol americano são de total contato físico e de alta velocidade. As transições dos jogadores são feitas em vários momentos para que as jogadas certas possam acontecer de acordo com a tática traçada pelo técnico. A leitura dessas substituições rápidas são feitas a olho humano, onde podem ocorrer equívocos e possíveis erros na identificação dos jogadores.

Outro fator crucial é que o mercado de futebol americano é muito valioso. Conforme descrito com mais detalhes na subseção 2.1.2 deste projeto, a *NFL - National Football League* (Liga Nacional de Futebol) é a liga desportiva mais rica do mundo. A revista Forbes (2019) descreve que a liga investe grandemente em ferramentas de *marketing* digital e tecnologias que auxiliam no desempenho dos jogos. Dan Lovinger, vice-presidente executivo de vendas de publicidade da rede de transmissão *NBC Sports*, informa que o preço médio de um comercial de 30 segundo no ar gira em torno de U$ 5 milhões, totalizando um gasto médio de aproximadamente U$ 500 milhões durante apenas uma partida de futebol americano. O show do intervalo patrocinado pela empresa *Pepsi* mantêm um acordo que custa cerca de U$ 7 milhões por ano, segundo o *Sports Business Journal*. O valor dos ingressos do *Super Bowl* varia entre U$ 950 e U$ 5 mil (Assentos *Premium*), onde esses valores podem alcançar níveis mais altos em mercados paralelos. Outro fator que esta em grande crescimento dentre a população que acompanha a *NFL* é o numero de apostas no evento *Super Bowl*, que é denominado o melhor evento para realizar apostas. Na 51ª edição do evento, o *Nevada Gaming Control Board* (Conselho de Controle de Jogos de Nevada) registrou um recorde de U$ 138,5 milhões em apostas (BADENHAUSEN, 2018).

Com base nisso, pode-se reconhecer a importância de realizar um tratamento minucioso nos dados das partidas, para que não ocorra nenhum equívoco dentro de campo que pode comprometer algum índice.

1.4 Organização do trabalho

Este trabalho é composto por cinco capítulos, estruturados da seguinte forma: o Capítulo 2 é composto pelos fundamentos conceituais que foram necessários para o entendimento da área de visão computacional. No Capítulo 3 foi abordado a metodologia utilizada para a construção deste trabalho. Subsequente, o Capítulo 4 relata todo o desenvolvimento do trabalho, ressaltando todas as etapas de funcionamento, descrição do sistema e requisitos. Por fim, o Capítulo 5 apresenta as considerações finais obtidas com o projeto, seguida das análises dos resultados e possíveis estudos futuros.

2 FUNDAMENTOS CONCEITUAIS

Este capitulo tem por objetivo descrever o referencial teórico usado para a elaboração deste trabalho cientifico e suas fundamentações. O capitulo está dividido em seções que abordam os principais temas utilizados no desenvolvimento do contexto bibliográfico. A seção 2.1 relata o contexto de futebol americano e tecnologias aplicadas no meio desportivo. Ja a seção 2.2 descreve os conceitos de visão computacional e as etapas do processamento de imagem, exemplificando as principais tecnologias utilizadas neste contexto. A seção 2.3 trata os conceitos de Engenharia de *Software* e suas metodologias ágeis. Para finalizar, a seção 2.4 descreve todas as ferramentas fundamentais para o desenvolvimento desse projeto.

2.1 Futebol Americano

O futebol americano é um esporte que se destaca devido ao porte físico de seus atletas e as disputas de força dentro de campo. O tempo de duração de uma partida pode demorar várias horas dependendo das pontuações em questão.

Do mesmo modo, vários jogadores participam de inúmeras jogadas dentro de campo, e é muito complexo acompanhar todas elas de forma detalhada, até mesmo para as pessoas que são treinadas e capacitadas para as analisarem.

O futebol americano consiste de uma série de jogadas, objetivando alcançar jardas[1] para somar mais pontos que seu adversário. Cada jogada consiste em 4 tentativas de avançar mais jardas do campo adversário. São 22 jogadores dentro de campo com possibilidades infinitas de substituição enquanto a bola não estiver em jogo. Um time tenta avançar e outro tenta impedir esse avanço, caso o time atacante consiga as jardas ele continua com a posse de bola, caso contrário o time adversário recebe a bola de volta e inicia a sua tentativa de ganhar jardas e chegar a *End Zone* (Zona Final) (ESPN, 2019).

Sua origem, datada em 1876, teve suas regras e seu modo de jogo evoluídas até chegar a um ponto onde a tecnologia pareava com o esporte, proporcionando uma velocidade acima da média nos resultados e na competitividade entre os jogadores da modalidade. Os treinadores começaram a utilizar alguns recursos para conseguir os melhores resultados

[1] Jarda é um termo muito utilizado em países anglo-americanos que representam distâncias curtas. No futebol americano, as distâncias percorridas pelos jogadores são representadas por jardas. Cada jarda corresponde a aproximadamente 0,91 metros (BRASIL ESCOLA, 2019).

dentro e fora de campo. Assim, a tecnologia se tornou um dos pilares de um time de ponta ao longo das competições. Sabe-se que o esporte é algo totalmente imprevisível mas existem pessoas que discordam dessa afirmação. De acordo com Nepomuceno e Carvalho (2012), todos os times que se utilizam de estatísticas e análise de dados têm vantagens sobre os seus adversários.

Atualmente, os dados obtidos manualmente por times e técnicos são coletados e em seguida, analisados por ferramentas de estatística. Dessa forma, fica mais facil reconhecer os padrões e entregar informações que não são obtidas usando apenas a capacidade cognitiva e intelectual humana.

Com um número enorme de jogadores para avaliar durante a temporada do futebol profissional, o melhor modo para que treinadores, dirigentes e olheiros das equipes possam ter ao menos uma ideia de quais jogadores tem o potencial de agregar mais desempenho ao time, é um lugar onde todos os atletas se reúnem para mostrar suas habilidades. O *Draft* é um evento anual onde os times grandes podem analisar e selecionar jogadores de futebol americano universitário para reforçar o seu elenco. Esse é um dos maiores eventos que acontece na intertemporada da *NFL*. Já o *Combine* é um evento que acontece antes do *Draft*, que proporciona aos executivos, técnicos e responsáveis pelo departamento pessoal analisarem a capacidade física e mental dos jogadores universitários (MCGEE; BURKETT, 2003).

Sendo assim, ter dados precisos sobre as habilidades destes jogadores e compará-las com outros jogadores participantes do *Combine* e do próprio elenco também serve para decidir como o time vai ter que lidar com determinados atletas e determinadas posições.

De acordo com Bass (2012) um exemplo claro disto é a avaliação dos *quarterbacks*. Com muitos times precisando de um jogador para liderar os seus ataques, observar bem os jogadores da posição durante o *Combine* pode ser decisivo nos rumos da franquia definir se vale a pena escolher um *quarterback* no *Draft* (Evento que acontece depois do *Combine*) , ficar com um que já está no elenco ou partir para o mercado em busca de um que possa cumprir com as expectativas da equipe.

Para os jogadores, o *Combine* é a melhor forma de tentar impressionar os observadores das equipes e convencê-los de que ele pode ser o atleta ideal. Quem se destaca com bons números durante os testes pode ver sua cotação aumentar nas listas das equipes e até ter a chance de ser escolhido durante a primeira rodada do *Draft*, uma oportunidade que apenas 32 jogadores conseguem a cada ano.

2.1.1 Tecnologias aplicadas no esporte

O contexto da tecnologia atual vem sendo aplicado há largos passos na área desportiva. Tal ação proporciona uma enorme mudança nos resultados, gerando discussões relacionadas ao beneficio e malefício desse progresso.

Katz e Green (1989) explicam que com o uso de técnicas do meio desportivo incorporadas a tecnologia, é possível ampliar a performance e a inteligência dos atletas, fazendo com que as habilidades dos atletas seja cada vez mais aprimorada.

Isso ocorre porque atualmente os atletas possuem um grande numero de tecnologias para os auxiliarem no esporte. Vários *softwares* e ferramentas auxiliam os profissionais a estarem sempre preparados fisicamente e psicologicamente para exercer a sua tarefa.

A esgrima é um dos esportes que mais utiliza a tecnologia a seu favor, onde um colete feito de metal foi criado para emitir um pulso elétrico quando a espada o tocar. No *taekwondo* também existe sensores de toque, que ficam acoplados abaixo do colete acolchoado que serve para reduzir o impacto físico no corpo do atleta. Com isso, a tecnologia proporcionou uma possibilidade de calcular ate mesmo a potência de cada dano sofrido pelos atletas (SCHATTENBERG, 2013).

Além dos rádios comunicadores utilizados pelos árbitros se comuniquem dentro de campo, o futebol recebeu algumas tecnologias para que as partidas se tornem mais justas, como por exemplo os sensores implantando dentro das bolas. O dispositivo consegue verificas se a bola ultrapassou a linha delimitadores e, caso aconteça, o sensor emite imediatamente um alerta para os árbitros (MOTA, 2012).

Outra tecnologia implementada recentemente no futebol é o *VAR - Video Assistant Refere*, mais conhecida como árbitro de vídeo. Essa tecnologia surgiu com a ideia de minimizar ao máximo o erro humano dentro das partidas de futebol. No entanto, o *VAR* esta restrito a análise somente em situações de gol, pênalti e cartão vermelho. Sendo assim, além dos árbitros que já existiam dentro de campo, agora existe a equipe responsável pelo *VAR*, que analisa os lances de uma forma mais criteriosa, usando os recursos tecnológicos que conseguem captar detalhadamente cada jogada dentro de campo (RIBEIRO, 2019).

Apesar de tanta tecnologia, a reportagem feita por Amaro e Fernandes (2019) apresenta um atraso de 46% nas tomadas de decisões dos árbitros nos lances com checagem do *VAR*, o equivalente a 1:50 minutos.

Já o *VAR* na *NFL* se chama *Instant Replay* (Replay Instantâneo). Na liga em questão, todos os jogos contam com a avaliação de imagem por parte da arbitragem. “Em 2017, apenas 429 marcações, dentre 39.967 jogadas, sofreram revisões. Desde 1999, 37% das marcações em campo foram alteradas após avaliação". As avaliações levam em média 1:44 minutos para serem analisadas (LIMA, 2018).

2.1.2 O mercado do futebol americano

O futebol americano é o esporte mais popular nos Estados Unidos e, segundo Badenhausen (2018), a *NFL* é a liga esportiva mais rica do mundo, girando cerca de U$ 2,5 bilhões por cada time participante, operando com lucros de U$ 101 milhões por franquia.

Isso pode ser comprovado analisando o documentário feito por Eckstein (2019) no qual está descrito que, segundo a *Bloomberg*[2], estima-se um faturamento aproximado de U$ 15 bilhões durante a temporada de 2018, acima das estimativas de U$ 14,2 bilhões em 2017 e U$ 13,3 bilhões em 2016.

A *NFL* mantêm seus fluxos de receita categorizados em *“national revenue” and “local revenue”* (Receita nacional e local). A receita nacional esta relacionada a acordos de TV, *merchandising* e licenciamentos negociados nacionalmente pela própria *NFL*. Esse valor é dividido igualmente entre as equipes, independentemente do seu desempenho individual. O relatório anual de 2018 do time profissional de futebol americano *Green Bay Packer* apresenta um ganho de aproximadamente U$ 8,1 bilhões em receita nacional da *NFL* (Aproximadamente U$ 255 milhões para cada uma das 32 equipes). Já a receita local consiste em vendas de ingressos, concessões e patrocinadores obtidos pelas próprias equipes. O *Green Bay Packer* fechou o ano de 2018 com cerca de U$ 196 milhões em receita local, o que representa cerca de 43% da sua receita total nesse mesmo ano (ECKSTEIN, 2019).

Outra estatística importante é a apresentada por Diniz (2019), que relata um aumento significativo na audiência da rede de TV *ESPN* durante a 53ª edição do *Super Bowl*, evento final da *NFL*. O jornal Folha de S.Paulo (2018) registrou um crescimento de 33% na audiência da temporada de 2018 da *NFL* no canal de esportes *ESPN* Brasil, se comparado com a temporada anterior. “Fora os Estados Unidos, o Brasil é o segundo país mais interessado do mundo na *NFL* atualmente, perdendo apenas para o México (FOLHA DE S.PAULO, 2018).”

[2] A Bloomberg (2019, online) oferece notícias, dados, análises e vídeos aos negócios e mercados ao mundo.

2.2 Visão computacional

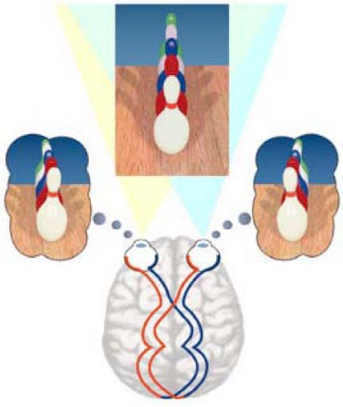
A visão computacional evoluiu consideravelmente nos últimos anos. Em consequência dessa evolução, a visão computacional se aprimorou a ponto de chegar mais próximo da visão humana, com a capacidade de maior eficiência em várias situações.

A visão computacional abrange todas as técnicas e métodos de processamento de imagem em um único meio, com o objetivo de ser mais eficiente nas análises de dados e informações compostas dentro de uma imagem. Silva, Lopes e Araújo (2017) contextualizam que algoritmos de visão computacional utilizam matrizes bidimensionais ou hiperdimensionais como entrada de dados e, a partir desta, produzem informações compactadas como saída.

De forma didática, a área de visão computacional utiliza modelos descritivos de objetos, pessoas e/ou cenas capturadas digitalmente para automatizar processos e realizar tomadas de decisão.

A visão computacional foi desenvolvida, segundo Marr (1976), através da neurofisiologia da visão humana (Figura 1). Seu modelo era estabelecido em níveis de compreensões necessárias à computação da visão estereoscópica, ou seja, o modelo é capaz de trabalhar com eficiência no ambiente tridimensional, onde a análise é feita através de duas imagens obtidas em postos diferentes.

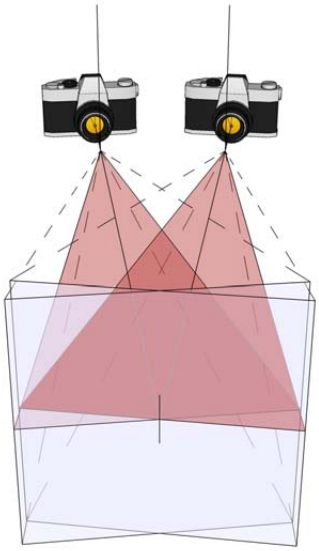
Figura 1: Esquema mostrando imagens captadas em cada olho. Devido à diferença de ponto de vista, as imagens então captadas são diferentes.



Fonte: Peronti (2008)

Peronti (2008) explica em seu artigo que estereoscopia é a visualização feita por dois mecanismos de captura de imagem em um mesmo foco (Figura 2).

Figura 2: Mecanismos para captação de imagens com focos visuais coincidentes.



Fonte: Peronti (2008)

Segundo Rehem e Trindade (2009), devido ao avanço tecnológico, desenvolveram-se computadores com maior capacidade de processamento gráfico, proporcionando ferramentas com um potencial maior na área de visão computacional. Pode-se dizer que essas ferramentas são bibliotecas onde o seu código fonte é constituído de um agrupamento de funções que potencializam o processamento gráfico de imagens e vídeos. Sendo assim, ao utilizar essas bibliotecas, tem-se a possibilidade de desenvolvimento de técnicas de aperfeiçoamento gráfico para realizar rastreamento de movimentos e de características humanas em tempo real.

2.2.1 Captura e identificação de imagem

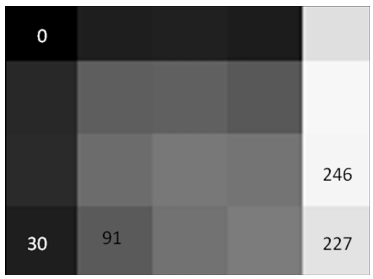
A tecnologia vem crescendo cada vez mais em todo mundo, proporcionando a ampliação do uso de dispositivos de alto desempenho. Paralelo a isso, o fluxo de dados aumenta proporcionalmente, exigindo uma velocidade de conexão maior com a Internet. Um exemplo disso são os *smartphones*, que se popularizaram por serem dispositivos multifuncionais, podendo ser utilizados para tarefas profissionais e pessoais.

Por conseguinte deste avanço científico, é notório a grande evolução na utilização de imagens digitais em várias áreas, como, por exemplo, em transmissões de TV, conteúdos via streaming, imagens capturadas por satélite e até mesmo imagens transmitidas pelas redes sociais. No entanto, como funciona a captura de uma imagem em tempo real? Qual o conceito de imagem?

Uma imagem digital pode ser considerada como sendo uma matriz de pontos elementares, em que cada ponto recebe o nome de *pixel*. Quanto maior a quantidade de *pixels* melhor a resolução da imagem e consequentemente maior o seu tamanho. Cada *pixel* é representado por um valor que indica a intensidade de brilho, denominado nível de cinza, e a quantidade de níveis de cinza depende da quantidade de bits usada na representação de cada *pixel* (SOUZA; CORREIA, 2007).

Sendo assim, a matriz de pontos elementares de uma imagem digital pode ser representada conforme na Figura 3.

Figura 3: Representação de uma imagem digital ampliada ao limite dos pixels.



Fonte: Pereira (2017)

A captura e identificação em tempo real de uma imagem consiste em analisar um ambiente tridimensional e elaborar digitalmente uma imagem 2-D (Duas dimensões), ocasionando uma imagem estática daquele ambiente tridimensional selecionado. Esse conceito está relacionado a perspectiva, segundo Peronti (2008). Para que isso seja possível, é necessário utilizar um dispositivo capaz de realizar essa ação, ou seja, uma câmera digital, *smartphones*, dentre outros.

Segundo Chmielewski (2009), isso se tornou capaz devido a uma tecnologia que surgiu nos anos 70 denominada CCD, *Charge Coupled Device* (Dispositivo acoplado de carga): “O sensor CCD ou dispositivo de carga acoplada é uma matriz de elementos sensíveis à luz, fabricados utilizando tecnologia MOS, *Metal Oxide Semiconductor* (Semicondutor de óxido metálico), onde cada *pixel* pode ser considerado como um capacitor MOS.”

Atualmente, o sensor mais utilizado nos dispositivos de captura de imagem são de sistemas digitais CMOS - *Complementary Metal Oxide Semiconductor* (Semicondutor complementar de óxido metálico), que surgiu nos anos 90 devido a um protótipo do sensor de imagem APS - *Active Pixel Sensor* (Sensor de pixel ativo) criado pela NASA - *National Aeronautics and Space Administration* (Administração Nacional Aeronáutica e Espacial), que possibilitou a fabricação direta de funções como *zoom*, diferentes resoluções de aquisições, acesso

aleatório, etc., podendo executar todas as funções do CCD. “Os sensores de imagem APS são formados por elementos sensíveis à luz, capazes de gerar um sinal elétrico ou carga proporcional à intensidade da luz que incide sobre eles.” (CHMIELEWSKI, 2009)

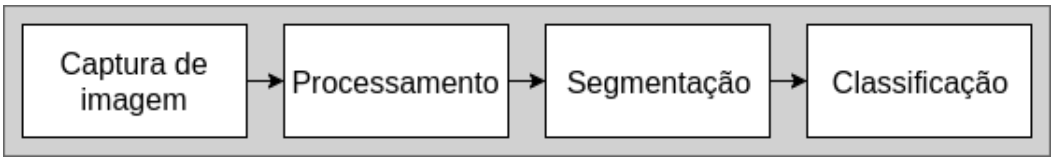
2.2.2 Processamento de imagem

O grande marco da área de processamento de imagens aconteceu no século XX com o surgimento dos primeiros computadores digitais com grande capacidade de processamento e o início do programa espacial norte-americano, ocorreu um grande impulso na área de processamento de imagem. O uso de técnicas computacionais de aprimoramento de imagens teve início no *Jet Propulsion Laboratory* (Laboratório de Propulsão a Jato), localizado no centro tecnológico da NASA, em 1964, quando "imagens da lua transmitidas por uma sonda Ranger eram processadas por computador para corrigir vários tipos de distorção inerentes à câmera de TV acoplada à sonda". Essa tecnologia foi usada em grandes expedições tripuladas, como a Apollo (FILHO; NETO, 1999).

Para realizar o processamento de uma imagem, são definidos passos a serem seguidos para a garantia do objetivo final. A captura da imagem consiste no uso de dispositivos físicos sensíveis a espectros de energia eletromagnética que convertem o sinal elétrico para um formato digital. O pré-processamento consiste no realce da imagem para enfatizar características de interesse ou recuperar imagens que sofreram alguma degradação devido à introdução de ruído, perda de contraste ou borramento. A segmentação é a extração ou identificação dos objetos contidos na imagem, separando a imagem em regiões. Por fim, a classificação, é o processo que identifica a imagem observada (GONZALEZ; WOODS et al., 2002).

As etapas básicas do processamento de imagem estão representadas através da Figura 4.

Figura 4: Etapas básicas do processamento de imagens.



Fonte: Adaptada de Gonzalez, Woods et al. (2002)

Seres humanos conseguem distinguir vários padrões de cores com certa facilidade, levando em consideração que a análise é feita em um ambiente tridimensional. Na computação, esse discernimento de cores são mais complexos, independentemente da dimensão na

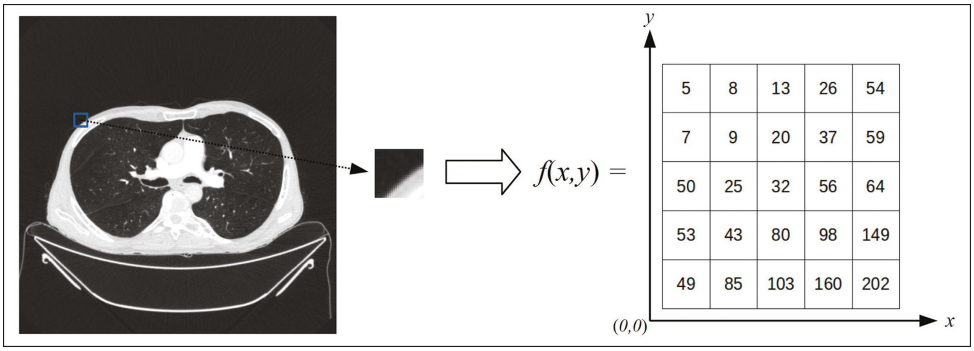
qual a imagem será analisada. Isso ocorre porque vários fatores podem contribuir para a má performance computacional no tratamento da imagem, como por exemplo a falta ou excesso de luz, qualidade do sensor, qualidade da imagem, dentre outros.

> "Objetos que emitem luz visível são percebidos em função da soma das cores espectrais emitidas. Tal processo de formação é denominado aditivo. O processo aditivo pode ser interpretado como uma combinação variável em proporção de componentes monocromáticas nas faixas espectrais associadas às sensações de cor verde, vermelho e azul, as quais são responsáveis pela formação de todas as demais sensações de cores registradas pelo olho humano. Assim, as cores verde, vermelho e azul são ditas cores primárias. Este processo de geração suscitou a concepção de um modelo cromático denominado RGB (Red, Green, e Blue), para o qual a Comissão Internacional de Iluminação (CIE) estabeleceu as faixas de comprimento de onda das cores primárias." (QUEIROZ; GOMES, 2006)

Cada *pixel* é representado por um valor numérico que corresponde a sua cor em questão. Sendo assim, para que seja possível representar uma imagem em alguma escala de cor monocromática (preto e branco, ou escalas de cinza), basta associar o *pixel* a um valor numérico relacionado a sua escala de tom.

A Figura 5 exemplifica a associação dos *pixels* de forma a obter uma imagem monocromática. Segundo Moreira (2000), para obter uma imagem em tons de cinzento, basta associar cada *pixel* um valor inteiro não negativo de um byte, onde o valor 0 corresponde a cor preta e o valor máximo, 255, corresponde a cor branca. Os valores intermediários correspondem aos variados tons de cinza.

Figura 5: Imagem médica digital em escalas de cinza (*pixels* monocromáticos).

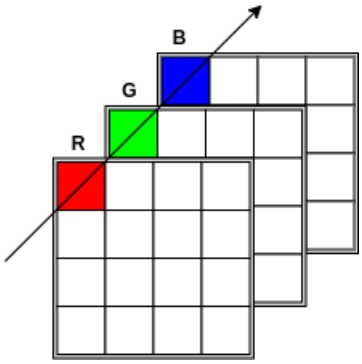


Fonte: Santos et al. (2019)

Já as imagens coloridas exigem um poder maior de processamento para serem reconhecidas. Isso ocorre porque as imagens em RGB - *Red, Green, and Blue* (Vermelho, Verde e

Azul) precisam de mais de uma banda[3] para serem processadas, ou seja, são analisadas três matrizes de cores para formar a paleta de cor específica do tom capturado (Figura 6). Depois da análise, são formadas as cores distintas que compõe a imagem.

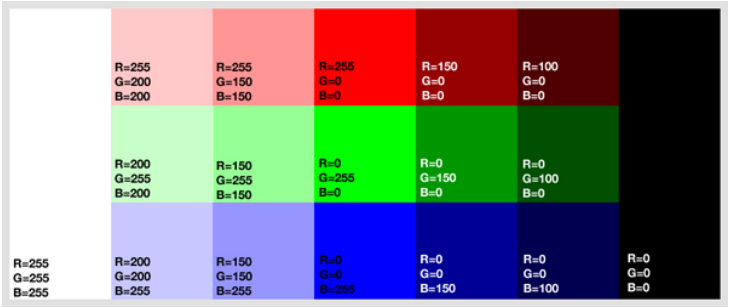
Figura 6: Matriz de *pixels* RGB.



Fonte: Elaborada pelos autores do projeto.

De forma mais detalhada, Lopes, Silva e Bonfim (2013) exemplificam que imagens coloridas também são imagens multibanda, ou multiespectral. As cores visíveis através de olhos humanos podem ser representadas pela combinação de bandas das cores primárias vermelha, verde e azul (*Red, green* e *blue*, respectivamente). A imagem colorida também pode ser armazenada por meio de imagens cromáticas e mapas de cores. Nesse caso, o valor de cinza de cada *pixel* na imagem se torna um índice para uma entrada de mapa de cores, enquanto a entrada em si do mapa de cores contem os valores dos componentes referentes as tonalidades *RGB* (Figura 7).

Figura 7: Imagem digital em tons coloridos (*pixels* RGB).



Disponível em: https://apenasimagens.com/pt/pixel-imagem-digital/

[3] Cada matriz ou conjunto de cor diferente é denominado banda de cor. Subconjuntos de três bandas espectrais azul, vermelha e verde compõem uma imagem RGB. Em imagens monocromáticas (Preto e branco), o objeto é representado em apenas uma banda espectral (CRÓSTA, 1999).

No entanto, a imagem capturada por algum dispositivo eletrônico pode chegar de forma irregular ate a parte de processamento. Essas falhas podem ser caracterizadas de várias formas, como por exemplo a presença de *pixels* ruidosos, brilho e/ou contrastes desregulados, caracteres com dígitos incompletos ou apagados como em digitalizações de documentos.

A parte de processamento fica responsável por elaborar uma melhoria da imagem em questão, ajustando todos os parâmetros para que seja possível analisar precisamente todas as informações que estão disponíveis no arquivo de imagem. Sendo assim, por analogia as imagens processadas, trata-se de uma etapa que analisa de forma profunda todos os dados contidos na imagem, ou seja, a fase de processamento abrange os níveis mais baixos de análise de imagens, pois trabalham diretamente com valores de intensidade dos *pixels*, visto que neste período não existe nenhuma informação relacionada a imagem para que seja possível facilitar o trabalho.

2.2.2.1 *Segmentação de imagens*

De forma a dar continuidade ao ciclo de processamento de imagem e verificação desta, a segmentação veio para aprimorar a parte de processamento, auxiliando na detecção de maiores detalhes e reduzindo tempo de processamento. Isso ocorre porque a segmentação é a área que tem por responsabilidade dividir a imagem em várias partes significativas, separando as áreas de interesse dentro desta (FILHO; NETO, 1999).

No entanto, a segmentação exige um poder de processamento muito grande do *hardware* para que a análise da imagem sugerida seja feita. Sendo assim, o nível no qual será feito essa subdivisão de imagem depende muito da ocasião e do problema que está sendo proposto para resolver. De acordo com Gonzalez, Woods et al. (2002), para não ocorrer perca desnecessária de processamento e, consequentemente, perda de tempo, a segmentação deve parar assim que os objetos de interesse da aplicação forem isolados.

Portanto, dentro da segmentação existem métodos que podem ser seguidos a fim de analisar detalhadamente cada figura. Segundo Morgan et al. (2008), as classificações dos métodos utilizados na segmentação são definidos como interativos ou automáticos. Basicamente, a diferença entre os dois são bem simples: um é executado através de intervenções humanas e o outro não.

Além disso, no processo de segmentação interativa, o usuário utiliza ferramentas e técnicas que se adéquam da melhor forma a sua imagem e necessidade, solucionando-a da melhor forma possível. Esse método normalmente é mais utilizado para solucionar problemas específicos, onde as condições da imagem podem interferir drasticamente na análise

final da mesma como, por exemplo, *softwares* que analisam imagens de doenças graves adquiridas através de ressonâncias, onde a aplicação pode confundir um ruído ou uma área com má iluminação em um ponto de análise clínica.

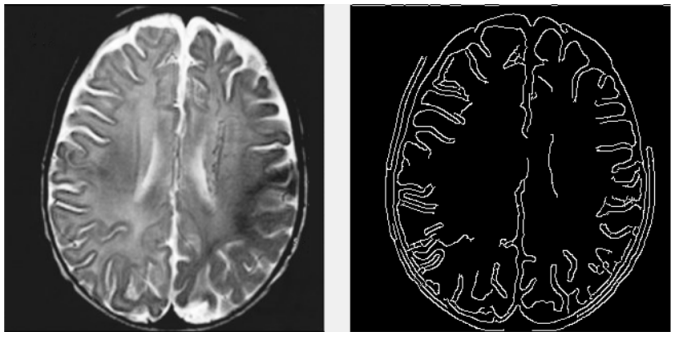
Já o processo de segmentação automática onde, na maioria das vezes, utilizam robôs para realizar as tomadas de decisões através dos resultados obtidos na análise da imagem, inexiste interferência humana no processo. Esse método está sendo aplicado, por exemplo, em carros autônomos, onde o mesmo identifica a presença de obstáculos em sua frente e com base no obstáculo e na proporção do mesmo, uma ação e tomada.

Para complementar, Morgan et al. (2008) enfatiza que existem métodos que são classificados de acordo com a representação dos objetos a serem segmentados, que são os métodos de borda ou orientados a regiões.

O processo de segmentação baseado no método de bordas utiliza basicamente pontos de uma imagem onde ocorre alguma intensidade de luz, ou seja, onde os *pixels* estão mais visíveis, realçando a borda do objeto e consequentemente diferenciando do fundo da imagem. Pode-se também localizar uma borda através de uma mudança brusca nos tons de cinza, gerados por regiões distintas. Com base na borda que foi extraída da imagem, tem-se então uma imagem topográfica do objeto que será analisado.

A Figura 8 mostra como funciona uma análise baseada no método de bordas.

Figura 8: Resultado de uma análise feita utilizando o método de bordas.



Fonte: Freitas (2016)

Conforme Morgan et al. (2008) enfatiza em seu artigo, o método orientado a regiões é dividido em três abordagens relevantes:

- Classificação por *pixel*.

- Agregação de *pixel*.
- *Split-and-merge* (Divisão e conquista).

A descrever cada uma das abordagens de forma didática, a classificação por *pixel* nada mais é que a identificação de características presentes na imagem como, por exemplo, cor, texturas, a fim de classificar os *pixels* de acordo com as várias possibilidades de classes ou objetos da imagem.

Na abordagem de agregação por *pixel* o objetivo é encontrar um *pixel* dentro da imagem e, a partir desse *pixel*, ocorre o crescimento de regiões conexas. O desenvolvimento das regiões acontece até alcançar o critério de parada do crescimento, onde estará representado um objeto dentro da imagem.

Já a segmentação utilizando a bordagem *split-and-merge* é mais complexa. Segundo Ventura (2009), ao aplicar essa abordagem de segmentação, o objetivo final consiste em conseguir subdividir a imagem em vários quadrantes que satisfaça uma prioridade. Após a concretização desta tarefa, realiza-se a verificação de cada quadrante, observando se este atende ou não a prioridade definida. Caso o quadrante não satisfaça a prioridade, a subdivisão acontece novamente em busca de outros quadrantes.

Por fim, o processo de fusão é realizado, acontecendo então o agrupamento das partes similares, ou seja, que atende as prioridades definidas. Esse processo só finaliza quando não existe nenhuma possibilidade de realizar divisões ou agrupamentos.

2.2.2.2 *Histograma*

O histograma é um método que auxilia na identificação de objetos e/ou características específicas da imagem, obtendo uma maior precisão nos resultados obtidos.

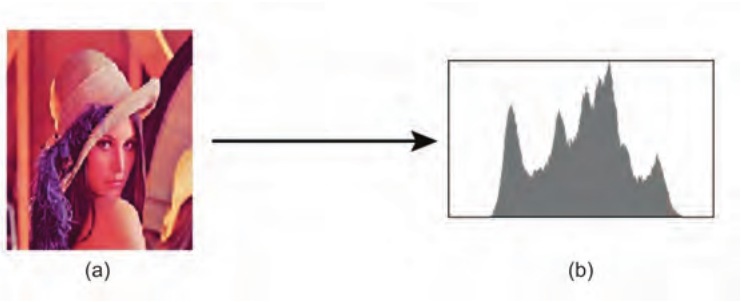
Segundo Filho e Neto (1999), histograma são conjuntos de vários números no qual são indicados os percentuais de *pixels* de uma imagem que possui determinados níveis de cinza. Estes valores, normalmente representados por gráficos, apresentam, para cada nível de cinza, o seu percentual de *pixels* correspondente na imagem. Com base nessa análise feita pelo histograma, pode-se obter os níveis de contraste, brilho, e ate mesmo informações de predominância clara ou escura.

Filho e Neto (1999) explicam que, através de equações matemáticas, é possível obter um resultado satisfatório ao analisar cada elemento deste conjunto. Este trabalho não tem por

finalidade apresentar e/ou explicar cálculos matemáticos que cada função executa.

De forma a complementar o assunto, Dias (2013) expressa em sua tese que, ao obter o histograma da imagem, pode-se alcançar medidas estatísticas dos níveis de cinza da imagem, como por exemplo o seu valor mínimo e máximo, valor médio, variância e desvio padrão. Portanto, o histograma seria como um método de probabilidade, onde o número de *pixels* de um determinado nível de cinza pode ser utilizado para calcular um outro *pixel* com o mesmo valor de cinza na imagem (Figura 9).

Figura 9: Imagem (a) e seu respectivo histograma (b).



Fonte: Dias (2013)

2.2.2.3 *Classificação de imagens*

A classificação de imagem é a ultima etapa do processamento de imagem (Figura 4). Em síntese, esta etapa é responsável por realizar a classificação das imagens levando em consideração as suas características.

Entretanto, segundo Liberman (1997), nessa etapa do processamento, o grau de abstração de cada característica da imagem podem ser classificados em três níveis distintos: baixo, médio e alto.

No processo de baixo nível são utilizados os *pixels* originais da imagem como parâmetros de comparação, para que no final do processo seja gerado propriedades da imagem, em forma de valores numéricos, associados a cada *pixel* que foi analisado. Sequencialmente, o nível médio coleta essas propriedades numéricas geradas pelo processo de baixo nível e produz uma lista de características da imagem. Por fim, o processo de alto nível reúne estas características ocasionadas pelo processo anterior buscando interpretá-las, formando assim o conteúdo da imagem.

Segundo Liberman (1997), o processo de classificação ou interpretação de uma imagem é a parte mais inteligente da visão computacional. O autor do artigo cita que essa é uma das

etapas de maior alto nível, no qual permite-se obter a “compreensão e a descrição final do fenômeno inicial”.

Para complementar, Liberman (1997) explica que o processo de classificação de imagem possui duas técnicas para realizar suas tarefas, sendo divididas em supervisionada ou não-supervisionada. A classificação não-supervisionada consiste em um agrupamento automático de sequências similares de uma imagem analisada. Conforme já prescrito neste trabalho no contexto de segmentação interativa e agora completado por Liberman (1997), nessa etapa a imagem será segmentada em um número indeterminado de classes, no qual o usuário também será responsável por gerenciar essas classes a fim de alcançar seus objetivos.

De acordo com Máximo e Fernandes (2005), no processo de classificação supervisionada, o analista ou usuário filtra as classes de informações seguindo os seus padrões de interesse e separa, na imagem, as regiões que satisfazem essas classes. Após a delimitação das classes, a técnica analisará as mesmas com o objetivo de delimitar *pixels* que serão utilizados como parâmetros para a busca de demais *pixels*.

Simplificadamente, a técnica de classificação supervisionada utiliza amostras de características coletadas durante o processo para identificar cada *pixel* definido como *pixel* desconhecido, ou seja, tons de *pixels* que não fazem parte das características já coletadas anteriormente seguindo os filtros definidos pelo usuário.

- *Haar Cascade*

A técnica de *Haar Cascade* utiliza a classificação de imagens para obter um padrão de características que foram extraídas da imagem. Essa classificação é utilizada para montar uma cascata de características, ou seja, um conjunto de imagens. A principal base para a detecção de objetos do classificador *Haar* são os recursos extraídos da imagem, ou seja, ao invés de usar os valores de intensidade de um *pixel*, usa-se as alterações nos valores de contraste entre os grupos retangulares dos *pixels*. Basicamente, *Haar Cascade* é baseada em *Haar Wavelets*, que utiliza uma sequência de funções redimensionadas em quadrantes que juntas formam uma base de *wavelets* (WILSON; FERNANDEZ, 2006).

A detecção de objetos e faces utilizando técnicas de classificadores em cascata baseados em recursos *Haar* é um método eficaz proposto por Viola, Jones et al. (2001) em seu artigo. Á abordagem é baseada em *machine learning* (aprendizado de máquina) no qual a função cascata é treinada a partir de enumeras imagens positivas e negativas. Através desse

recurso, pode-se obter a eficiência em detecção de objetos em outras imagens (OPENCV, 2019). Este trabalho não descreve os detalhes do funcionamento do detector de Viola-Jones. O leitor interessado pode encontrá-lo em (VIOLA; JONES et al., 2001).

- *Machine Learning*

Bishop (2006)

2.2.2.4 *Similaridade*

A similaridade consiste em realizar uma buscar dentro de uma imagem específica com o objetivo de reconhecer/encontrar objetos semelhantes a um modelo de busca. Nessas funções de busca por similaridade são utilizados cálculos de vetores de características para realizar as comparações de igualdade (DIAS, 2013).

Os seres humanos possuem uma grande facilidade de reconhecer informações apresentadas de maneira visual, onde consequentemente são capazes, com grande facilidade, de interpretar imagens diversificadas sem grande esforço. Exemplificando esta situação, os seres humanos conseguem distinguir de forma fácil a diferença entre um círculo grande e um círculo pequeno, um quadrado grande de um quadrado pequeno, a diferença entre um triângulo e um quadrado de tamanhos idênticos, dentre outros. Conforme Silva (2009) relata em seu artigo, com a tecnologia disponível atualmente para realizar a construção de aplicações capazes de identificar informações dentro de uma imagem ainda é muito ineficiente, quando comparada com a capacidade humana.

Segundo Maia e Souza (2013), algoritmos de similaridade trabalham com métricas que informam o quanto uma imagem é parecida com a outra. Ou seja, pode-se aplicar essas métricas utilizando padrões de buscas a fim de uma análise mais específica. De forma estatística, Maia e Souza (2013) completam que possui dois tipos básicos de medidas de similaridade: correlação e coseno. Seguindo o contexto do artigo, a similaridade por correlação entre dois vetores retorna um valor booleano, ou seja, 0 e 1, onde o valor de retorno igual a 1 significa que há uma similaridade forte naquele ponto, ou seja, os valores dos vetores são parecidos e, se o retorno for 0, não existe correlação. No entanto, o autor enfatiza a presença de um retorno igual a -1, no qual a similaridade daquele ponto é inversa ao padrão de busca. Ja a similaridade por coseno é similar a correlação, no qual o retorno também e 0 e 1, porém nesse método é analisado o tamanho do vetor e a formação de um ângulo entre os mesmos. Quanto mais próximo de 1 for o valor, mais similares são os vetores.

Dias (2013), Maia e Souza (2013) utilizam em seus artigos a função euclidiana para realizar cálculos de distâncias nas estruturas. Essa função utiliza métricas de similaridade para calcular a distância entre dois vetores de características, percorrendo o vetor apenas uma vez. A distância euclidiana entre dois pontos (Xi e Xj) é definida através de uma equação matemática, na qual não faz parte do escopo deste projeto.

2.2.3 Reconhecimento facial

Algoritmos de reconhecimento facial estão presente em diversos dispositivos, como por exemplo *smartphones* e câmeras digitais, e até mesmo carros autônomos, que escaneiam seus obstáculos para realizar uma tomada de decisão. Grandes aperfeiçoamentos dentro desta área estão sendo implementados de forma gradativa com a finalidade de realizar análises com grandes precisões e mais próximas à visão humana.

Segundo Szeliski (2010), a área de reconhecimento facial foi a que teve mais sucesso nos dias atuais. No entanto, a aplicação desse algoritmo para realizar a busca de uma pessoa dentro de milhares de pessoas em tempo real, ainda é um desafio para a tecnologia, embora pra humanos essa tarefa também seja difícil. Mas quando esse grupo de pessoas é reduzido a, por exemplo, um grupo familiar ou grupo de amigos, a ferramenta tem um desempenho excepcional. Além disso, o objeto na qual esta sendo utilizado para ler o ambiente físico, ou seja, o dispositivo de captura de imagem, influencia diretamente com o desempenho da ferramenta.

Sendo assim, o resultado do reconhecimento facial pode ser ampliado se a imagem for capturada de forma frontal às pessoas, quando é possível localizar por completo os rostos das pessoas em questão. Porém, vários fatores podem interferir na análise da imagem, por exemplo: iluminação, qualidade dos sensores, dentre outros fatores de interferência. Para tentar solucionar esses problemas, uma das primeiras abordagens a ser seguida pela ferramenta e tentar analisar os locais de características específicas da imagem como, por exemplo, nariz, boca, olhos, e aplicar medidas de distância entre os pontos de características encontrados (SZELISKI, 2010).

Existem várias ferramentas que proporcionam ao usuário utilizar a tecnologia de visão computacional para realizar o reconhecimento facial. Dentre estas ferramentas, a biblioteca OpenCV disponibiliza funções capazes de reconhecer objetos e pessoas.

2.3 Engenharia de *software*

Quando se pensa em desenvolvimento, manutenção, especificação e criação de um *software*, pensa-se também em tecnologias e práticas de gerência de projetos para que a execução da aplicação aconteça de forma organizada, produtiva e com a máxima qualidade possível. Tudo isso esta contido em engenharia de *software*.

Segundo Pressman e Maxim (2016) em seu livro, um *software* bem sucedido é aquele que atende a todos os requisitos do usuário, fica implementado durante um bom tempo, é de fácil manutenção e operabilidade. Por outro lado, um *software* mal sucedido pode acarretar diversos fatos desagradáveis, levando os usuários a insatisfação e ao erro.

Apesar de gerentes, lideres de projetos e profissionais envolvidos com a área técnica entenderem a necessidade de uma metodologia mais disciplinar no desenvolvimento de *softwares*, existe ainda discursos de como e qual é a melhor metodologia a ser aplicada no projeto. Essa indecisão corre devido a grande demanda de produção que acontece atualmente, principalmente no setor de desenvolvimento de aplicações. Outra coisa que impacta negativamente é que profissionais e empresas começam a desenvolver *softwares* de forma descontrolada mesmo com uma metodologia organizacional aplicada, justamente por não estarem preparados para uma abordagem disciplinar (PRESSMAN; MAXIM, 2016).

Com base nisso, a engenharia de *software* evoluiu rigorosamente, passando de uma simples técnica implementada por um publico relativamente pequeno para uma comunidade que objetiva o planejamento e a organização ates de iniciar qualquer tipo de desenvolvimento.

Sendo assim, em 2001, o engenheiro de *software* Kent Beck juntamente com os principais desenvolvedores de métodos ágeis, assinaram o “Manifesto para o Desenvolvimento Ágil de Software” (SOMMERVILLE, 2011), que tem por iniciativa a seguinte maneira:

> "Estamos descobrindo melhores maneiras de desenvolver *software*, o fazendo e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais:
>
> Indivíduos e interações do que processos e ferramentas;
> *Software* em funcionamento do que documentação abrangente;
> Colaboração dos clientes acima de negociação contratual;
> Respostas a mudanças acima de seguir um plano;
> Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda.(SOMMERVILLE, 2011)"

2.3.1 Engenharia de Requisitos

Quando se pensa em projetar e construir um *software*, os desafios e as imaginações tomam proporções escalonáveis para obter a melhor forma de iniciar e prosseguir com o desenvolvimento do projeto. O grande problema nesse processo é definir quais são os requisitos necessários para que o sistema atenda as necessidades do usuário.

Segundo Pressman (2005), a engenharia de requisitos é basicamente uma etapa da engenharia de software, que deve ser iniciada durante as atividades de comunicação e continua no decorrer do desenvolvimento do *software*. "Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho."

A engenharia de requisitos tem por objetivo fornecer regras apropriadas para entender as necessidades do cliente afim de avaliar a viabilidade, negociar soluções razoáveis, validar as especificações e gerenciar as necessidades dos usuários na medida em que o sistema seja desenvolvido (PRESSMAN; MAXIM, 2016).

2.3.2 Desenvolvimento ágil de *software*

Quando se pensa em desenvolvimento de *software*, deve-se reconhecer que o processo é bem instável e com baixa previsibilidade, se tornando algo complicado de estabelecer métricas a serem seguidas. Vários pontos de interferência podem intervir para que um *software* não possa ser desenvolvido da maneira correta, como por exemplo uma equipe desestruturada, nos quais os integrantes não possuem uma boa convivência entre si, ou uma metodologia de desenvolvimento inadequada. Reconhecer que este é um grande desafio é algo sensato de se fazer. No entanto existem mecanismos de correção para melhorar o processo de desenvolvimento.

A metodologia ágil surgiu para organizar esses processos de desenvolvimento de *software*, elaborando uma padronização nos projetos para que seja possível otimizar os fluxos de trabalho e melhoras a produtividade do projeto. Segundo Soares e Caldeira (2004), a principal característica da metodologia ágil e que ela pode ser adaptada, ao invés de ser preditiva. Ou seja, se ocorrer algum problema no decorrer do desenvolvimento, a própria metodologia é flexível o bastante para contornar a situação e prosseguir com o desenvolvimento do projeto. Portanto, uma empresa pode facilmente criar a sua própria metodologia de trabalho, seguindo a sua experiência empresarial, analisando os seus acertos e erros para elaborar um procedimento que se adéqua as suas necessidades.

Soares e Caldeira (2004) continua exemplificando que a metodologia ágil trabalha com constantes *feedbacks* e reuniões, nas quais permitem aos membros da equipe expor as

facilidades e dificuldades de suas tarefas, bem como o seu status de desenvolvimento. A partir destes *feedbacks*, o gestor pode traçar a melhor maneira de organizar a equipe para que os membros com maior experiência deem apoio nas dificuldades apresentadas pelos outros integrantes da equipe. Outro grande motivo expressado no contexto é o fato ocorrer entregas constantes de partes operacionais do software. Desta maneira, o usuário final do software pode ter uma prévia de como o sistema está sendo desenvolvido, bem como suas funcionalidades e *design*, podendo solicitar alguma possível alteração ou identificar algum problema antes da implantação oficial dos modulos.

2.3.2.1 *Kanban*

O *Kanban* é um método organizacional de desenvolvimento de software que permite a interação de várias áreas e membros do projeto por meio de cartões que contem o progresso de cada atividade. Cada cartão contém uma instrução a ser seguida pela área ou pelo integrante no qual foi designado para a atividade. Sendo assim, seu principal foco e fornecer um trabalho progressivo, apresentando as evoluções e dificuldades de forma clara e transparente, favorecendo uma cultura de melhoria contínua (PRIKLADNICKI; WILLI; MILANI, 2014).

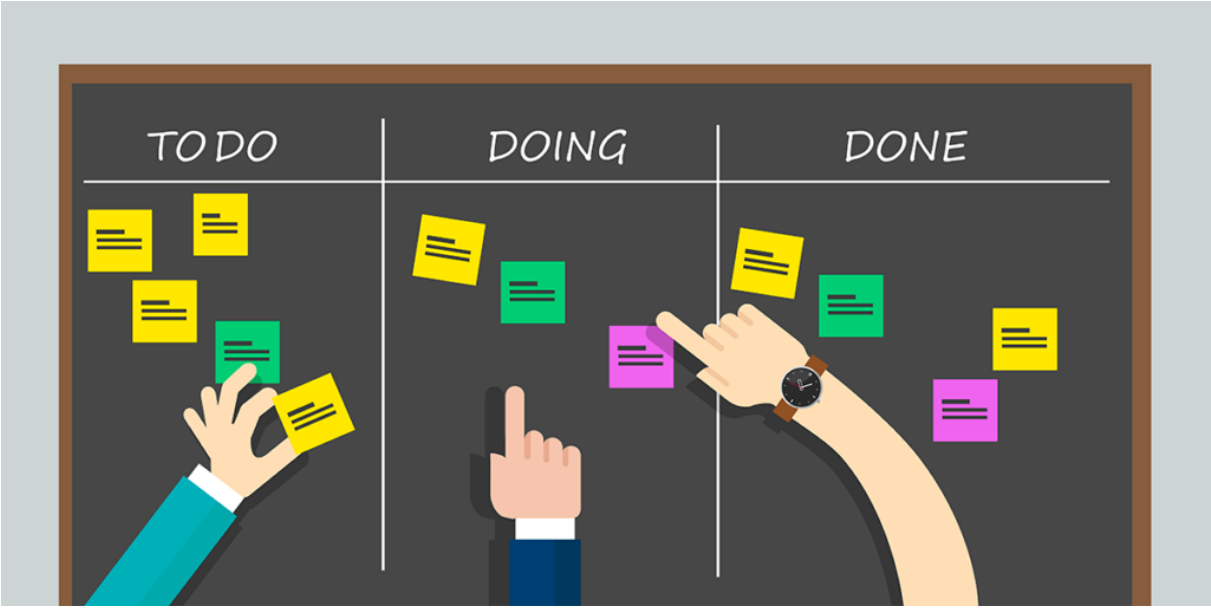
Portanto, a ferramenta *Kanban* tem um grande potencial em trabalho conjunto para a finalização de um item em especifico, justamente para que não ocorra nenhum gargalo na entrega de um item essencial para o trabalho do integrante ou da área seguinte. Outra grande característica é evitar ou diminuir o índice de trabalhos repetitivos que, por um eventual descuido, possa acontecer de desenvolvedores realizarem a mesma codificação de uma mesma função ou *API - Application Programming Interface* (Interface de Programação de Aplicações), por exemplo.

A abordagem *Kanban* foi criado pelo vice presidente da *Toyota Motor Company*, o Sr. Taiichi Ohno, no qual teve como principal objetivo o aumento do valor agregado entregue nas atividades de cada colaborador de sua equipe. Com este pensamento, Ohno concluiu que as pilhas de materiais estocados e as filas de de espera era um “dinheiro parado” que a empresa *Toyota* estava desperdiçando. Portando, Ohno uniu os princípios do método *just in time* (determina que tudo deve ser feito na hora exata) juntamente com o *Jidoka* (determina que corrigir o problema em si não é o bastante, e sim corrigir a origem do problema) para elaborar um método mais aprimorado de organização dentro da empresa *Toyota*, denominado *Kanban* (SUGIMORI et al., 1977).

Devido a sua simplicidade, facil implantação, entendimento e manipulação, atualmente a abordagem *Kanban* é utilizada em diversas empresas para realizar o controle de desem-

penho de diversas área. A Figura 10 está exemplificando um exemplo clássico de uma abordagem *Kanban*.

Figura 10: Exemplo de uma abordagem *Kanban*.



Disponível em: https://novida.com.br/blog/kanban/

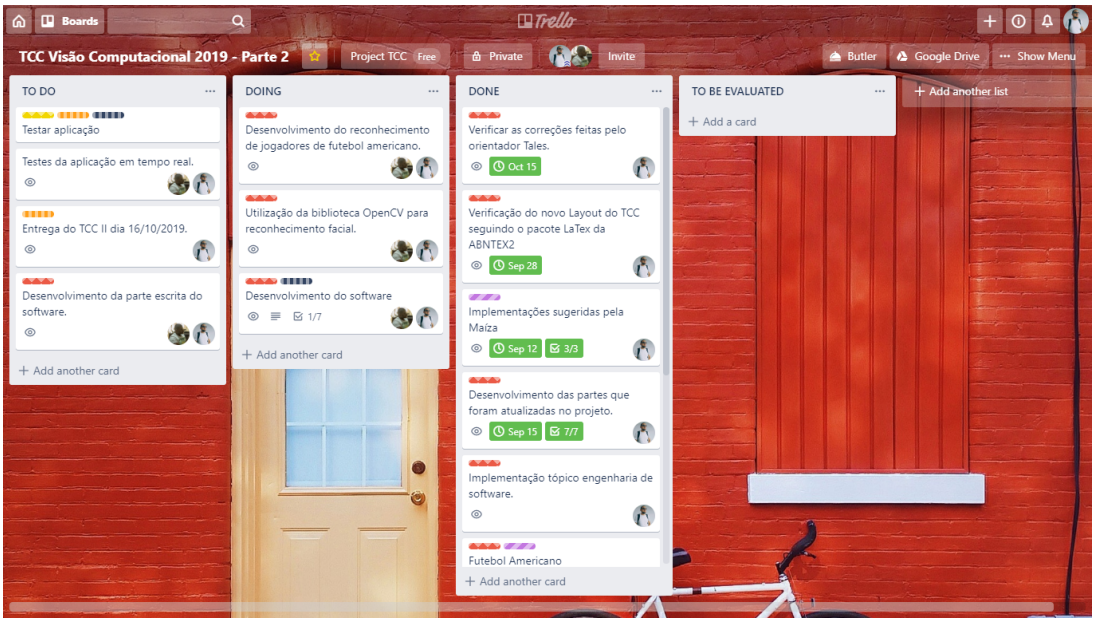
2.3.2.2 *Trello*

O *Trello* é uma ferramenta de gerenciamento de projetos que tem por finalidade auxiliar os lideres de projetos a organizar da melhor forma o fluxo de tarefas relacionadas ao seu trabalho. Utilizando o método *Kanban*, todas as atividades disponibilizadas no *Trello* são exibidas em um único ambiente de trabalho visível por todos os membros da equipe que foram adicionados a ele. As organizações das atividades são feitas através de *cards* (cartões), onde cada membro fica responsável por exercer a atividade que lhe foi designada. O *Trello* possui uma interface bastante amigável e simples de entender (JOHNSON, 2017).

Por ser uma aplicação *Web*, o *Trello* necessita de uma conexão com a internet para que os trabalhos possam ser compartilhados e disponibilizados para toda a equipe. A aplicação disponibiliza grandes possibilidades de organização, como por exemplo a criação de atividades distintas dentro de um mesmo projeto, classificação das atividades com rótulo de cores (urgente, não urgente, pouco urgente, descartar, etc), definir a situação de cada atividade (concluída, a fazer, fazendo, etc), dentre outras. Outra grande utilidade que o *Trello* disponibiliza são as notificações por e-mail, que informa a todos os membros a situação de suas atividades dentro do projeto, bem como seu *status*, prazo e níveis de urgência.

A Figura 11 apresenta a estrutura de organização deste projeto feita no *Trello*.

Figura 11: Estrutura organizacional deste projeto feita no *Trello*.



Fonte: Elaborada pelos autores.

2.4 Ferramentas de desenvolvimento do projeto

Em relação as ferramentas de desenvolvimento, é notório que existe um trajeto amplo a ser percorrido pelos programadores. Isso ocorre porque existem várias linguagens de programação e *frameworks* que auxiliam no desenvolvimento de sistemas. A escolha da melhor linguagem de programação e o melhor *framework* para desenvolvimento depende muito do problema a ser resolvido e também das habilidades do programador. Há também um grande impasse na escolha da melhor linguagem e/ou do melhor *framework*, que está relacionado a: interesses e aplicações comerciais; comunidade, para sanar possíveis duvidas e curva de aprendizado.

2.4.1 Linguagem de programação

Linguagem de programação são codificações escritas sequencialmente, seguindo uma estrutura assíncrona para a resolução de algum problema ou tarefa, que tem por finalidade ser compreendida por um computador. Essas codificações descrevem ao computador a sequência lógica de execução das funções e os comandos nos quais ele deve executar para que a tarefa e/ou problema seja executado da melhor forma possível. Basicamente são divididas em linguagem de baixo nível e de alto nível, que significam, respectivamente, linguagens próximas ao entendimento de máquina ou *hardware* (Binário ou hexadecimal)

e linguagens próximas as linguagens naturais, ou seja, de fácil entendimento humano (*While, if, write, read,* etc), nos quais são necessários compiladores para realizar a tradução, tornando possível a compreensão da máquina (KELLEHER; PAUSCH, 2005).

Na busca por uma linguagem que poderia satisfazer e tornar possível a construção de uma solução para o impasse explícito no trabalho, fora identificado uma forte tendência na utilização do *Python*.

2.4.1.1 *Python*

De acordo com Pilgrim e Willison (2009), a projeção da linguagem enfatiza o trabalho do programador sobre o computacional, possibilitando assim a construção de bibliotecas e frameworks com uma facilidade acima do normal.

Python foi criado por Guido van Rossum em 1991, com a ajuda de seus colegas Jack Jansen e Sjoerd Mullender. O objetivo deles era criar uma linguagem de fácil entendimento, orientada a objetos, menos complexa possível (SONGINI, 2005).

Segundo Oliveira (2007), a linguagem sofreu vários ajustes no decorrer dos anos, tornando-se muito popular dentre os desenvolvedores e, consequentemente, dando início a enumeras aplicações. Portanto, *Python* é uma linguagem orientada a objetos, fortemente tipada, com propósitos gerais de alto nível e de código aberto, objetivando uma construção ágil no desenvolvimento de aplicações. Sua sintaxe é bem simples e de fácil entendimento, reduzindo o custo de manutenção em *softwares* criados a partir desta. Suas bibliotecas garantem ao programador um vasto acervo de funções que tem por finalidade facilitar o seu trabalho, reduzir tempo de codificação e evitar arquivos com extensas linhas de código. Devido à comunidade de código aberto, onde desenvolvedores tem acesso ao seu código fonte, a popularização da linguagem vem crescendo de forma significativa, visto que esta ainda não é muito conhecida (SONGINI, 2005).

2.4.1.2 Biblioteca *OpenCV*

Devido aos avanços em estudos científicos tecnológicos na área de visão computacional, o surgimento de bibliotecas que utilizam desta tecnologia evoluiu consideravelmente, proporcionando aos usuários maior versatilidade na escolha da melhor ferramenta para uma possível solução de seus problemas. Dentre as principais ferramentas que implementam algoritmos de visão computacional, pode-se citar as mais utilizadas: *Matlab*, *OpenCV* e *scikit-image*. Este trabalho tem o objetivo de abordar apenas a biblioteca *OpenCV*, utilizada para o desenvolvimento de uma possível solução atendendo aos objetivos citados na se-

ção 1.1 deste projeto. A biblioteca *Open Source* (Código aberto) está disponível no seu site oficial OpenCV (2019).

Desenvolvida pela Intel no ano 2000, escrita nativamente em C++, a biblioteca OpenCV permite a manipulação de dados de imagens, manipulações vetoriais, rotinas de álgebra linear, desenvolvimento de algoritmos de processamento de imagem, calibração de câmeras, dentre outros. Sua flexibilidade com várias linguagens de programação, como por exemplo o *Python*, permite uma melhor integração com vários programas, evitando possíveis conflitos de incompatibilidade e proporcionando uma melhor flexibilidade no desenvolvimento de *softwares* (BARBOZA, 2009).

A bilioteca possui certificação BSD - *Berkeley Software Distribution*, representando que o software possui uma licença gratuita. A biblioteca contem mais de 2.500 algoritmos otimizados com diversas propriedades para resolverem problemas extensos. Há também vários setores de aplicação da biblioteca, visto que esta abrange diversas áreas como, por exemplo, reconhecimento de face e objetos, extração de modelos de objetos tridimensionais, união de imagens em uma única imagem, pesquisar por imagens semelhantes dentro de um banco de dados, acompanhar movimentos dos olhos, reconhecimento de cenários, dentre outros. No site oficial da ferramenta encontra-se dados nos quais informam que esta possui uma comunidade com mais de 47 mil usuários e um número estimado de download que ultrapassa a casa dos 18 milhões (CUNHA, 2013).

Segundo Cunha (2013), um dos objetivos da biblioteca *OpenCV* é fornecer uma infraestrutura robusta na área de visão computacional na qual seja de fácil manipulação, ajudando os desenvolvedores no processo de ampliação de aplicações sofisticadas de visão.

2.4.2 Ambientes virtuais

A virtualização vem sendo muito utilizada na área de desenvolvimento de *software* devido a sua flexibilidade e fácil manipulação para realizar a criação de ambientes virtuais. Isso ocorre porque, com a virtualização, o usuário pode criar vários ambientes virtuais e, dentro deles, realizar a instalação das dependências necessárias para a execução do projeto. Portanto, o desenvolvedor pode ativar e desativar o ambiente virtual assim que necessário, sendo que as dependências do projeto estarão instaladas somente nesse ambiente, e não globalmente na máquina. Ou seja, as dependências do projeto serão desativadas assim que a virtualização for encerrada. A utilização de virtualização permite ainda que recursos computacionais possam ser alocados para múltiplas aplicações simultaneamente, sendo que cada uma dessas aplicações possui seu ambiente isolado das demais.

A situação citada acima é bastante válida se analisarmos o crescimento de ferramentas de programação disponíveis atualmente. Instalar vários *frameworks*, pacotes de dependências de linguagens, bibliotecas e afins globalmente na máquina pode acarretar a lentidão, conflitos entre versões de linguagens de programação e *frameworks* dos projetos, dentre outros. Quando isso acontece, por exemplo, em um projeto feito por uma equipe, a atualização de todas as dependências deve ser feita em todas as máquinas que estão envolvidas no projeto, para que a versão seja padrão em toda a equipe.

2.4.3 Sistema de controle de versões

Durante o processo de desenvolvimento de *software*, a etapa de codificação gera várias linhas de códigos. Além disso, modificações e melhorias no *software* ocorrem constantemente no decorrer do tempo, seja a pedido do usuário ou alguma possível atualização. As equipes de desenvolvimento são compostas por vários tipos de desenvolvedores, cada um com sua personalidade e experiência. Sendo assim, os desenvolvimentos de *softwares* são feitos por etapas, onde cada desenvolvedor é responsável por entregar o que foi designado a ele.

Devido a isso, para organizar essas etapas de desenvolvimento, utiliza-se ferramentas que gerenciam e controlam diferentes versões de *software*. Segundo Loeliger e McCullough (2012), uma ferramenta que realiza o gerenciamento e o controle de versões de *software* ou outro conteúdo “é referida genericamente como um VCS - *Version Control System* (Sistema de controle de versão), um SCM - *Source Code Manager* (Gerenciador de código-fonte) ou um RCS - *Revision Control System* (Sistema de controle de revisão)”. Essas ferramentas controlam quais linhas de códigos foram alteradas, qual contribuidor do projeto fez a alteração, o horário da alteração, dentre outros. Além do mais, essas ferramentas possibilitam aos desenvolvedores uma opção de voltar o código para versões anteriores, caso algum *bug* (problema) na versão atual do sistema cause algum transtorno.

Loeliger e McCullough (2012) enfatizam que nenhuma pessoa criativa e cautelosa inicia um projeto sem um método de *backup*. Sendo assim, outra funcionalidade muito importante que as ferramentas de rastreamento e gerenciamento de código proporcionam para os desenvolvedores são os repositórios de *backup*, que mantêm hospedado de forma segura todas os arquivos relacionados ao sistema desenvolvido.

2.4.3.1 *Git*

Basicamente, o *Git* é um sistema distribuído e de código aberto que controla versões de arquivos desenvolvidos, no qual possui diversos comandos que auxilia os desenvolvedores a realizar projetos de grade e pequeno porte com velocidade e eficiência. Com esses coman-

dos, o *Git* disponibiliza um amplo conjunto de ferramentas para realizar a manipulação dos arquivos no seu repositório local ou *Web*, garantindo a integridade dos dados (TORVALDS; HAMANO, 2010).

Segundo Chacon e Straub (2014), o maior diferencial do sistema de controle de versões *Git* é o seu modelo de ramificação (*branch*), que possibilita o usuário a criar uma cópia do sistema principal. Com essa cópia, o desenvolvedor pode realizar implementações de melhorias/correções em trechos de códigos sem modificar a aplicação principal. Depois da implementação e dos testes, o desenvolvedor pode substituir a *branch* principal (*master*) pela *branch* com implementações. Os únicos arquivos que serão substituídos serão os arquivos editados.

2.4.3.2 *GitHub*

Já o *GitHub* é uma plataforma de hospedagem de códigos que utiliza o *Git* como controle de versão. O *GitHub* possui uma grande interação com repositórios *Git*, concentrando uma grande comunidade de desenvolvedores que colaboram para milhões de projetos (CHACON; STRAUB, 2014).

Ainda segundo Chacon e Straub (2014), o *GitHub* hospeda a maior porcentagem de repositórios *Git*. Isso ocorre porque a plataforma também disponibiliza recursos de gerenciamento de código, como por exemplo o rastreamento de problemas, revisão de código, edição *online* do código, dentre outros.

3 METODOLOGIA

A metodologia utilizada no projeto é a exploratória.

Segundo Ventura (2007), são verificadas grandes utilidades em estudos de casos realizados através de pesquisas exploratórias.

> "Por sua flexibilidade, é recomendável nas fases iniciais de uma investigação sobre temas complexos, para a construção de hipóteses ou reformulação do problema. Também se aplica com pertinência nas situações em que o objeto de estudo já é suficientemente conhecido a ponto de ser enquadrado em determinado tipo ideal. São úteis também na exploração de novos processos ou comportamentos, novas descobertas, porque têm a importante função de gerar hipóteses e construir teorias. Ou ainda, pelo fato de explorar casos atípicos ou extremos para melhor compreender os processos típicos (VENTURA, 2007)."

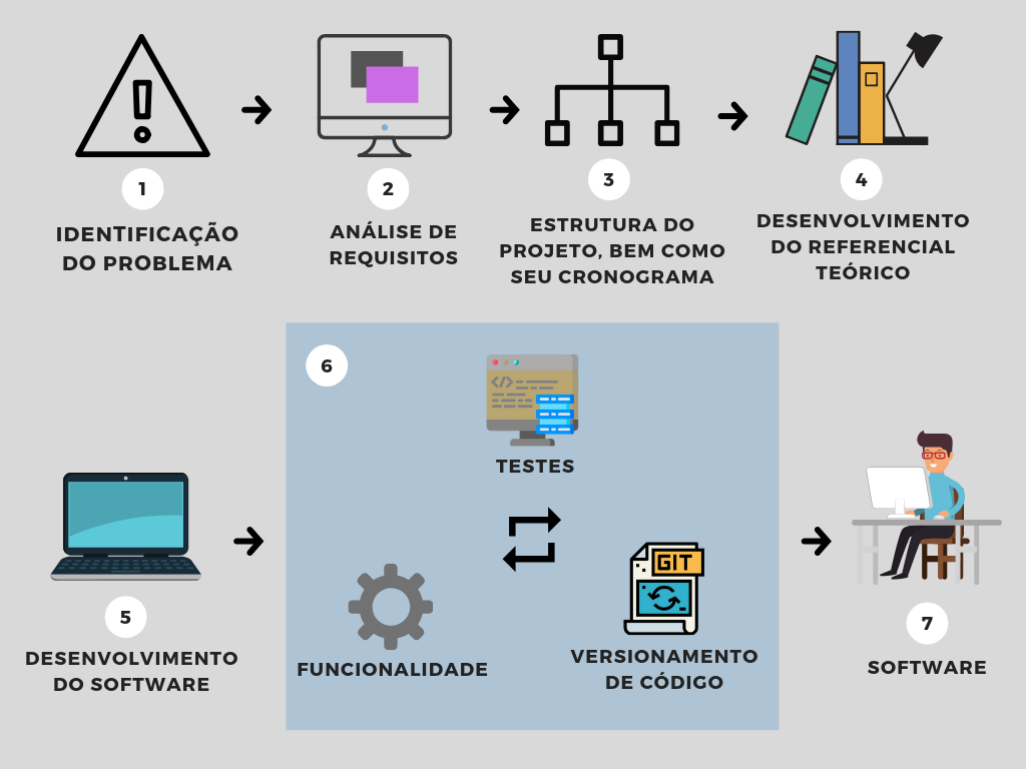
Sendo assim, será feito um estudo bibliográfico a partir do tema de visão computacional utilizando o acervo disponível sobre o assunto, com a finalidade de reunir informações para obter melhores resultados.

A pesquisa qualitativa será baseada principalmente nos três livros escolhidos: *Digital Image Processing* (GONZALEZ; WOODS et al., 2002), *Computer Vision: Algorithms and Applications* (SZELISKI, 2010) e Processamento Digital de Imagens (FILHO; NETO, 1999). Todos os livros foram escolhidos por transmitirem conceitos científicos, didáticos, teóricos e práticos sobre o assunto de visão computacional.

O desenvolvimento da aplicação proposta será feito através de estudos realizados sobre o tema. A aplicação será desenvolvida utilizando metodologia de desenvolvimento ágil, ou seja, será feita o planejamento de cada etapa de desenvolvimento do sistema, na qual o objetivo de cada entrega é proporcionar uma prévia do funcionamento do *software*. Essa ação foi tomada para que a evolução do sistema seja gradativa e eficiente.

Para resumir as etapas desta pesquisa, foi elaborado uma imagem ilustrativa da metodologia utilizada no decorrer do desenvolvimento deste projeto (Figura 12).

Figura 12: Metodologia de desenvolvimento do projeto.



Fonte: Elaborada pelos autores.

4 DESENVOLVIMENTO

Este capítulo tem por finalidade abordar todo o processo utilizado para o desenvolvimento desse projeto, relatando as principais ferramentas utilizadas e testes do algoritmo. O capítulo está dividido em seções, onde a seção 4.1 apresenta a descrição do sistema, etapas de funcionamento e seus riscos e restrições. A seção 4.2 descreve os requisitos do sistema, a seção 4.3 tem por objetivo descrever os códigos utilizados para a elaboração deste projeto e a seção 4.4 relata os ambientes de teste e os testes do software.

4.1 Descrição do sistema

O *software* apresentado no decorrer deste projeto tem por objetivo reconhecer um jogador dentro de campo em tempo real, utilizando as técnicas de processamento de imagem para realizar a extração de características importantes presentes em uma imagem.

Antes de descrever qualquer coisa relacionada a ferramenta, é preciso entender que o *software* passa por um aprendizado de máquina para entender os padrões de características que são necessária para atender as necessidades do projeto, que é reconhecer um jogador em campo. Portanto, o sistema utilizará classificação de imagem para extrair as características relacionadas aos jogadores de futebol americano.

Figura 13: Etapa de extração de características de uma imagem (A) e seu padrão de características (B).

(A)

```
[array([
[[163,162,161,0],[166,166,166,0],[166,167,169,0],
...,
[46,35,31,0],[49,37,35,0],[49,41,38,0]],
[[168,168,167,0],[170,170,171,0],[168,169,171,0],
...,
[56,44,39,0],[52,39,37,0],[52,45,42,0]],
[[172,172,171,0],[173,173,173,0],[174,176,177,0],
...,
[61,47,41,0],[56,42,38,0],[51,44,39,0]],
...,
[[156,168,96,0],[157,169,97,0],[158,170,98,0],
...,
[33,27,28,0],[31,31,30,0],[90,93,92,0]],
[[156,167,100,0],[156,167,100,0],[157,168,101,0],
...,
[59,48,44,0],[36,30,26,0],[27,24,21,0]],
[[155,165,103,0],[156,166,104,0],[159,169,107,0],
...,
[54,40,30,0],[51,37,30,0],[53,43,37,0]]],dtype=uint8)]
```

(B)

Fonte: Elaborada pelos autores.

O *haar cascade* fica responsável por realizar a classificação da imagem para obter seus padrões de características. Basicamente, o mesmo é alimentado manualmente por imagens aleatórias de jogadores de futebol americano para montar esse padrão de características. Sendo assim, pode-se definir o modelo de busca como o conjunto de padrões de características de jogadores de futebol americano que será utilizado para realizar a busca.

Em seguida, o *software* passa pela etapa de *machine learning* que ficará encarregada apenas de treinar o algoritmo utilizando o modelo de busca. Ou seja, nessa etapa do processo, o sistema recebe como parâmetro os padrões de características extraídos na etapa anterior e o *machine learning* analisa todos os seus pontos de interesse.

A etapa de aprendizado de máquina realiza um filtro que pode ser definido manualmente no algoritmo. No caso deste projeto, os filtros que foram criados e definidos para treinamento foram os de fisionomia de um jogador de futebol americano, as características do seu uniforme (capacete, número na camisa, tonalidade do uniforme) e os traços faciais. Após todas estas etapas, o *software* estará pronto para realizar a busca de um jogador dentro de campo.

Em seguida, o sistema recebe, através do dispositivo de captura de imagem, um vídeo em tempo real de uma simulação de uma partida de futebol americano.

O algoritmo realiza a análise de todos os *frames*[1] por segundo do vídeo para encontrar algo similar ao modelo de busca que foi treinado.

Caso ocorra a identificação de um jogador de futebol americano dentro do vídeo analisado, o algoritmo fica encarregado de realizar uma representação do mesmo. Essa representação será feita com um contorno verde ao redor do jogador.

4.1.1 Etapas de funcionamento

O funcionamento da ferramenta consiste em analisar uma imagem de um jogador objetivando adquirir todos os pontos de interesse da imagem, que servirá como parâmetro de busca para que o algoritmo seja mais eficiente quando o mesmo for exposto a uma situação mais complexa como por exemplo, analisar um jogador com capacete.

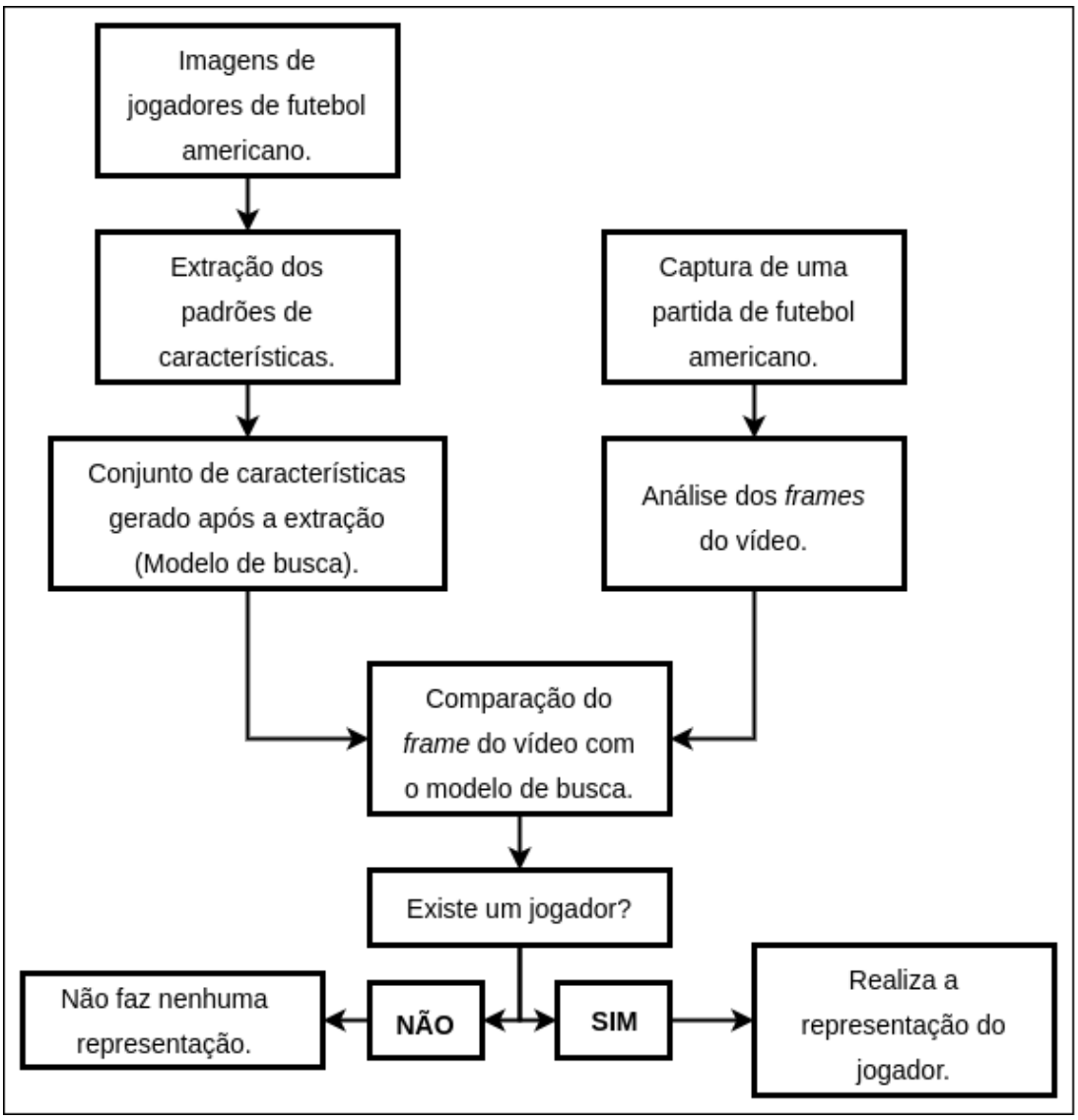
A análise feita pelo algoritmo ocorre de forma minuciosa e utilizando técnicas de processamento de imagem. Sendo assim, após a extração de todos os pontos de interesse da

[1] *Frames* por segundo é a taxa de atualização de imagens estáticas que formam uma cena animada dentro de um vídeo. A ilusão que nosso cérebro interpreta como movimento é feita através de vários quadros consecutivos em um curto período de tempo (TECMUNDO, 2011).

imagem, o algoritmo monta um conjunto de sequências lógicas das taxas de *pixels* existentes nesses pontos. Esta conversão pode ser demostrada de forma mais clara na Figura 13 da seção 4.1.

Após a montagem do conjunto, o algoritmo utiliza um dispositivo de captura de imagem para realizar uma busca por similaridade, ou seja, o algoritmo buscará algo semelhante ao padrão montado na etapa anterior. Após a busca feita pelo *software*, a detecção vai definir se existe ou não um jogador no ambiente que está sendo analisado. Se for constatado que existe um jogador e ele for igual ao modelo, a ferramenta vai apresentar o seu nome cadastrado. A Figura 14 representa o fluxograma do *software*.

Figura 14: Fluxograma de funcionamento do sistema.



Fonte: Elaborada pelos autores.

Ao final do processo, será feito uma análise dos dados apresentados pela ferramenta. Através desses dados, pode-se obter o percentual de acertos e erros, analisando também os cenários de maior e menor eficiência.

4.1.2 Restrições, riscos e exclusões do projeto

As restrições do projeto podem ser definidas através de todos os fatores que limitam as funcionalidades do mesmo. Basicamente, são condições impostas para a elaboração do projeto que devem ser obrigatoriamente cumpridas pela equipe no decorrer do desenvolvimento do sistema. Com base nisso, as restrições deste projeto são:

- Reconhecer pelo menos um jogador dentro de campo.
- Fazer a representação do jogador reconhecido.

Já a parte de riscos está relacionada a um evento ou situação incerta que pode afetar positivamente ou negativamente na execução do projeto. Descrevê-los é uma necessidade, para que saibamos identificar o momento certo que pode acontecer algum problema com relação ao software. Os riscos deste projeto são:

- Se não houver uma boa iluminação, pode acontecer falhas no reconhecimento e até mesmo inutilizá-lo.
- A qualidade do sensor utilizado para capturar a imagem pode interferir no reconhecimento.

A exclusão consiste em todos os requisitos que estão explicitamente fora do projeto. Basicamente, é tudo aquilo que a equipe de desenvolvimento deixa claro que não será realizado ao longo do desenvolvimento do projeto. Sendo assim, segue a relação de requisitos que estão excluídos deste projeto:

- Explicar matematicamente cada função executada.
- Reconhecer mais de um jogador dentro de campo.
- Solucionar algum possível problema identificado no processo de reconhecimento.

4.2 **Requisitos**

Requisitos de um sistema podem ser definidos como toda a solicitação, necessidade, funcionalidade, característica, especificação ou gerenciamento que são necessários para atender um usuário ou alguma organização.

4.2.1 Requisitos funcionais

Requisitos funcionais são responsáveis pelas funcionalidades, necessidades e características que definem um software, ou seja, eles definem o que o *software* vai fazer, o seu comportamento. Os requisitos funcionais deste projeto são:

- Realizar a classificação das imagens de jogadores de futebol americano.
- Gerar o *haar cascade* com os pontos de interesses das imagens classificadas.
- Definir o modelo de busca a partir dos padrões de características dos jogadores de futebol americano.
- Realizar o treinamento do algoritmo com o modelo de busca definido.
- Capturar o vídeo através de um dispositivo (*hardware*).
- Analisar cada *frame* por segundo do vídeo.
- Identificar um jogador de futebol americano dentro de campo.
- Representar o jogador analisado.

4.2.2 Requisitos não-funcionais

Basicamente, requisitos não funcionais descrevem como o sistema fará alguma coisa. Sendo assim, os requisitos não funcionais estão relacionados ao desempenho do sistema, usabilidade, confiabilidade, restrições de projeto, manutenção e atributos da qualidade. São os seguintes:

- O sistema deve ser implementado em *phyton*.
- O sistema necessita de dependências do *phyton* para a sua execução.
- A biblioteca *OpenCV* precisa estar instalada para a sua execução.
- O processamento da imagem será feito no próprio *hardware*.

- O sistema requer grande processamento de *hardware*.
- Para a definição de um novo modelo, o sistema precisa passar por uma manutenção dos parâmetros de busca e, consequentemente, um novo treinamento da ferramenta.
- Para que o sistema possa ser executado, é necessário realizar comandos via terminal.
- O *software* precisa de um dispositivo de captura de imagem como, por exemplo, uma *Web Cam*.
- O *software* pode ser executado em *Windows* e *Linux*.
- O *software* não pode ser executado em dispositivos *mobile*.

4.3 Explicando o código

Antes de explicar o código por trás do desenvolvimento deste projeto, é preciso informar que para executá-lo é necessário instalar a linguagem de programação *Python* e suas dependências.

Para iniciar a codificação do algoritmo em *Python*, é importante realizar a importações das bibliotecas necessárias para realizar o procedimento de reconhecimento. A biblioteca *cv2* é a *OpenCV*; a *NumPy* fica responsável por realizar cálculos de vetores multidimensionais; a *argparse* verifica e atribui os argumentos que são esperados; já a *imutils* é responsável por converter a imagem em uma matriz.

```
1 import cv2
2 import numpy as np
3 import argparse
4 import imutils
```

Também é necessário importar os módulos da biblioteca *imutils*.

```
1 from __future__ import print_function
2 from imutils.object_detection import non_max_suppression
3 from imutils import paths
```

Em seguida, o algoritmo deve ser programado para construir os argumentos necessários para iniciar o reconhecimento e analisar esses argumentos.

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--images", required=True,
    help="path to images directory")
args = vars(ap.parse_args())
```

Agora é necessário iniciar o descritor *HOG - Histogram of Oriented Gradients* (Histograma de Gradientes Orientados). Basicamente, o *HOG* utiliza cálculos matemáticos extremamente complexos que serve para classificar dados da imagem e gerar um *haar cascade* com suas configurações.

```
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

Para o melhor desempenho do algoritmo, foi criado uma pasta local contendo todas as imagens necessárias para realizar a extração dos padrões de características dos jogadores de futebol americano. Sendo assim, o trecho de código a seguir serve para ler todas estas imagens que estão dentro da pasta *images* localmente.

```
imagePaths = list(paths.list_images(args["images"]))
```

A partir desta etapa, o algoritmo vai executar uma estrutura de repetição para analisar todas as fotos alocadas na variável *imagePaths* utilizada no trecho de código acima.

Sendo assim, o primeiro laço da estrutura de repetição fica responsável por carregar e redimensionar a imagem para reduzir o tempo de detecção. Em seguida, ele melhora a precisão da detecção da imagem para melhorar a extração de características.

```
for imagePath in imagePaths:
        image = cv2.imread(imagePath)
        image = imutils.resize(image, width=min(400, image.shape[1]))
        orig = image.copy()
```

Em seguida, ele tenta detectar se existe uma pessoa na imagem analisada.

```
    (rects, weights) = hog.detectMultiScale(image, winStride=(4, 4),
        padding=(8, 8), scale=1.05)
```

Apos isso, a segunda estrutura de repetição que está alocada dentro da primeira, fica responsável por delimita os locais onde contem os pontos de interesse dentro da imagem analisada, ou seja, o algoritmo seleciona as áreas de interesse que serão utilizadas como parâmetro de busca.

```
    for (x, y, w, h) in rects:
        cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

Em seguida, o *software* realiza a técnica de supressão não máxima na imagem, ou seja, ele realiza a técnica de detecção de bordas para identificar os *pixel* de maior intensidade e, em seguida, ele realiza a técnica de supressão para eliminar os *pixels* que não possuem valores próximos aos da borda.

```
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
        pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)
```

A terceira estrutura de repetição fica responsável por montar uma delimitação do indivíduo detectado na imagem analisada.

```
    for (xA, yA, xB, yB) in pick:
                cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
```

Apos os procedimentos, o algoritmo fica responsável por desenhar as correspondências de informações contidas entre as imagens analisadas.

```
    filename = imagePath[imagePath.rfind("/") + 1:]
        print("[INFO] {}: {} original boxes, {} after suppression".format
                (filename, len(rects), len(pick)))
```

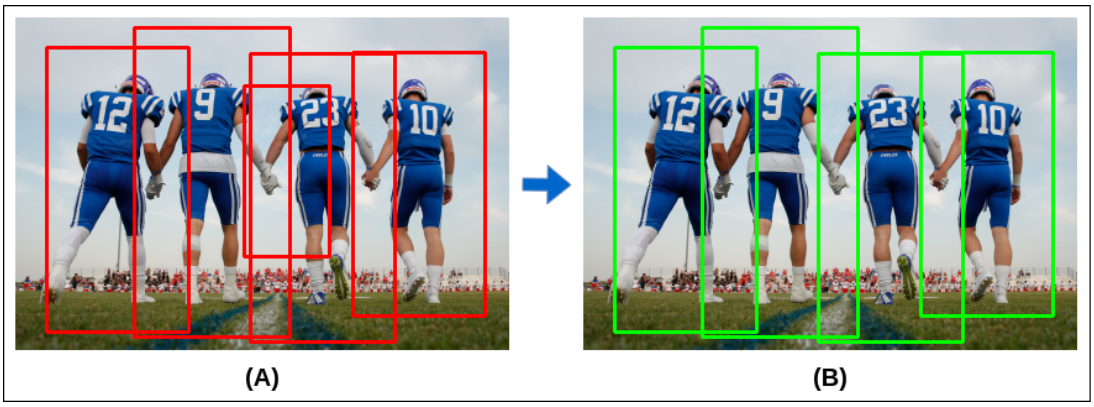
Por fim, o sistema mostra na tela a imagem analisada e os pontos de interesse que ele achou

dentro da mesma. Na outra imagem, o *software* mostra o que ele achou de semelhante após a análise.

```
cv2.imshow("Before NMS", orig)
cv2.imshow("After NMS", image)
cv2.waitKey(0)
```

A Figura 15 descreve o que o sistema faz com a imagem e o seu resultado.

Figura 15: A imagem (A) é a original, e as representações em vermelho são seus pontos de interesse. A imagem (B) é o resultado da detecção por bordas e seus pontos similares com a imagem (A).



Fonte: Elaborada pelos autores.

4.4 Ambiente de testes

O *software* criado no decorrer deste projeto é uma simples ideia de como usar técnicas de visão computacional aplicadas no reconhecimento de um jogador de futebol americano. Para exemplificar a usabilidade e os pontos fortes das técnicas utilizadas durante o desenvolvimento do mesmo, foram criado ambientes de testes para a exposição do algoritmo.

A Tabela 1 mostra quais ambientes foram escolhidos para realizas os testes do *software*:

Tabela 1: Ambiente de testes

Ambiente	Iluminação	Possível resultado
Partida de futebol americano	Boa	Positivo
Partida de futebol americano	Baixa	Negativo
Jogador parado	Boa	Positivo
Jogador em movimento (Lento)	Boa	Intermediário
Jogador de frente	Boa	Positivo
Jogador de costas	Boa	Positivo

4.4.1 Testes do *software*

5 CONSIDERAÇÕES FINAIS

Escreva aqui as conclusões deste trabalho, lembrando que os objetivos geral e específicos devem ser citados e comentados, ou seja, descreva como cada objetivo foi alcançado.

5.1 Análise dos resultados

5.2 Propostas de novos estudos

REFERÊNCIAS

AMARO, G.; FERNANDES, R. **VAR no Brasil demora 46% a mais do tempo recomendado pela Fifa**. Estadão, 2019. Disponível em: <https://esportes.estadao.com.br/noticias/futebol,var-no-brasil-demora-46-a-mais-do-tempo-recomendado-pela-fifa,70002970447>. Acesso em: 10 outubro 2019. Citado na página 29.

BADENHAUSEN, K. **Super Bowl: as cifras do maior evento esportivo do mundo**. Forbes, 2018. Disponível em: <https://forbes.uol.com.br/fotos/2018/02/super-bowl-as-cifras-do-maior-evento-esportivo-do-mundo/>. Acesso em: 29 setembro 2019. Citado 2 vezes nas páginas 25 e 30.

BARBOZA, D. P. d. S. Estudo da biblioteca opencv. Universidade Federal do Rio de Janeiro, 2009. Citado na página 50.

BASS, T. L. **Football skills & drills**. [S.l.]: Human Kinetics, 2012. Citado na página 28.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006. Citado na página 42.

BLOOMBERG. **Bloomberg**. 2019. Disponível em: <https://www.bloomberg.com/>. Acesso em: 29 setembro 2019. Citado na página 30.

BRASIL ESCOLA. **O Metro e a Jarda**. 2019. Disponível em: <https://brasilescola.uol.com.br/matematica/o-metro-jarda.htm>. Acesso em: 15 outubro 2019. Citado na página 27.

CHACON, S.; STRAUB, B. **Pro git**. [S.l.]: Apress, 2014. Citado na página 52.

CHMIELEWSKI, A. M. Análise e projeto de um sensor de imagem 0.35 μm cmos para compressão de dados no plano focal de câmeras digitais. **master dissertation**, Univ. Fed. Rio de Janeiro, 2009. Citado 2 vezes nas páginas 33 e 34.

CRÓSTA, A. P. **Processamento digital de imagens de sensoriamento remoto**. [S.l.]: UNICAMP/Instituto de Geociências, 1999. Citado na página 36.

CUNHA, A. L. B. N. d. **Sistema automático para obtenção de parâmetros do tráfego veicular a partir de imagens de vídeo usando OpenCV**. Tese (Doutorado) — Universidade de São Paulo, 2013. Citado na página 50.

DIAS, M. C. d. S. **Uso de algoritmos genéricos no ajuste de parâmetros de uma estrutura métrica para indexação e recuperação de imagens**. Dissertação (Mestrado) — Universidade Católica de Minas Gerais, 2013. Citado 3 vezes nas páginas 40, 42 e 43.

DINIZ, L. **Super Bowl: com crescimento de audiência na TV e streaming, ESPN lidera TV paga com transmissão da final da NFL**. ESPN, 2019. Disponível em: <https://www.espn.com.br/nfl/artigo/_/id/5251632/>. Acesso em: 29 setembro 2019. Citado na página 30.

ECKSTEIN, J. **How The NFL Makes Money**. Investopedia, 2019. Disponível em: <https://www.investopedia.com/articles/personal-finance/062515/how-nfl-makes-money.asp>. Acesso em: 29 setembro 2019. Citado na página 30.

ESPN. **O guia definitivo para você saber tudo de futebol americano**. 2019. Disponível em: <http://www.espn.com.br/infografico/guia-nfl-futebol-americano/>. Acesso em: 21 junho 2019. Citado na página 27.

FILHO, O. M.; NETO, H. V. **Processamento digital de imagens**. [S.l.]: Brasport, 1999. Citado 4 vezes nas páginas 34, 37, 39 e 53.

FOLHA DE S.PAULO. **Audiência do futebol americano na ESPN cresce 33% em relação à última temporada**. 2018. Disponível em: https://telepadi.folha.uol.com.br/audiencia-futebol-americano-da-nfl-na-espn-cresce-33-em-relacao-ultima-temporada/ Acessado em: 29 set. 2019. Citado na página 30.

FORBES. **Os times mais valiosos da temporada 2019 da NFL**. 2019. Disponível em: <https://forbes.uol.com.br/listas/2019/09/os-times-mais-valiosos-da-temporada-2019-da-nfl/>. Acesso em: 29 setembro 2019. Citado na página 25.

FREITAS, R. P. d. S. Sistema para detecção de bordas em exames clínicos. Universidade Federal Fluiminense, 2016. Citado na página 38.

GONZALEZ, R. C.; WOODS, R. E. et al. Digital image processing [m]. **Publishing house of electronics industry**, v. 141, n. 7, 2002. Citado 3 vezes nas páginas 34, 37 e 53.

JOHNSON, H. A. Trello. **Journal of the Medical Library Association: JMLA**, Medical Library Association, v. 105, n. 2, p. 209, 2017. Citado na página 47.

KATZ, L.; GREEN, J. Computer applications in physical education: A guide to technology in sport and recreation-lab manual. **Computes Research Ltd. Thornhill, Ontario**, 1989. Citado na página 29.

KELLEHER, C.; PAUSCH, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. **ACM Computing Surveys (CSUR)**, ACM, v. 37, n. 2, p. 83–137, 2005. Citado na página 49.

LIBERMAN, F. **Classificação de imagens digitais por textura usando redes neurais. 87 f**. Tese (Doutorado) — Dissertação (Mestrado em Ciência da Computação)–Universidade Federal do Rio . . . , 1997. Citado 2 vezes nas páginas 40 e 41.

LIMA, D. I. **'VAR da NFL' é mais flexível e abrangente que o do futebol; veja diferenças do melhor caso de sucesso na área**. ESPN, 2018. Disponível em: <https://www.espn.com.br/futebol/artigo/_/id/4754351/>. Acesso em: 10 outubro 2019. Citado na página 30.

LOELIGER, J.; MCCULLOUGH, M. **Version Control with Git: Powerful tools and techniques for collaborative software development**. [S.l.]: "O'Reilly Media, Inc.", 2012. Citado na página 51.

LOPES, D.; SILVA, F. d.; BONFIM, M. F. **Desenvolvimento do algoritmo para processamento de imagens digitais para diagnóstico de melanoma**. Tese (Doutorado) — Tese (Doutorado)—Master'sthesis, Centro Universit ario Cat olico Lasensiano . . . , 2013. Citado na página 36.

MAIA, L. C. G.; SOUZA, R. R. Medidas de similaridade em documentos eletrônicos. 2013. Citado 2 vezes nas páginas 42 e 43.

MARR, D. From computational theory to psychology and neurophysiology–a case study from vision. MIT Artificial Intelligence Laboratory, 1976. Citado na página 31.

MÁXIMO, O. A.; FERNANDES, D. Classificação supervisionada de imagens sar do sivam pré-filtradas. In: **XII Simpósio Brasileiro de Sensoriamento Remoto**. [S.l.: s.n.], 2005. p. 4139–4146. Citado na página 41.

MCGEE, K. J.; BURKETT, L. N. The national football league combine: a reliable predictor of draft status? **The Journal of Strength & Conditioning Research**, LWW, v. 17, n. 1, p. 6–11, 2003. Citado na página 28.

MOREIRA, N. **Processamento de Imagem**. 2000. Disponível em: <https://www.dcc.fc.up.pt/~nam/aulas/0001/pi/trabalho2/trab2/>. Acesso em: 19 mai. 2019. Citado na página 35.

MORGAN, J. et al. Técnicas de segmentação de imagens na geração de programas para máquinas de comando numérico. Universidade Federal de Santa Maria, 2008. Citado 2 vezes nas páginas 37 e 38.

MOTA, C. **Fifa aprova o uso de bolas com chip: Mundial de Clubes terá a tecnologia**. G1, 2012. Disponível em: <http://globoesporte.globo.com/futebol/noticia/2012/07/fifa-aprova-o-uso-de-bolas-com-chip-para-validacao-de-gols.html/>. Acesso em: 10 outubro 2019. Citado na página 29.

NEPOMUCENO, F. d. O.; CARVALHO, G. L. d. A importância das estatísticas no resgate da história dos doze mais tradicionais clubes de futebol do brasil. 2012. 43 f. **Monografia (Especialização)-Curso de Jornalismo Esportivo e Negócios do Esporte, Faculdade Metropolitanas Unidas, São Paulo**, 2012. Citado na página 28.

OLIVEIRA, I. D. d. **Uma perspectiva de extensão do modelo de aceitação de tecnologia para explicar o uso de linguagens de desenvolvimento WEB: pesquisa com desenvolvedores Pyton e Java**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2007. Citado na página 49.

OPENCV. **OpenCV - Open Source Computer Vision**. 2019. Disponível em: https://opencv.org Acessado em: 15 set. 2019. Citado 2 vezes nas páginas 42 e 50.

PEREIRA, M. S. **Imageamento com nêutrons: 30 anos de atividades no IPEN-CNEN/SP**. [S.l.: s.n.], 2017. ISBN 978-85-923404-1-4. Citado na página 33.

PERONTI, W. L. Princípios teóricos da estereoscopia. 2008. Citado 3 vezes nas páginas 31, 32 e 33.

PILGRIM, M.; WILLISON, S. **Dive Into Python 3**. [S.l.]: Springer, 2009. v. 2. Citado na página 49.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [S.l.]: McGraw Hill Brasil, 2016. Citado 2 vezes nas páginas 44 e 45.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: Palgrave Macmillan, 2005. Citado na página 45.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. [S.l.]: Bookman Editora, 2014. Citado na página 46.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. **RITA**, v. 13, n. 2, p. 11–42, 2006. Citado na página 35.

REHEM, A.; TRINDADE, F. H. Técnicas de visão computacional para rastreamento de olhar em vídeos. **Publicado em**, v. 3, n. 02, 2009. Citado na página 32.

RIBEIRO, F. **Dia do Futebol | Como a tecnologia é usada no futebol?** CanalTech, 2019. Disponível em: <https://canaltech.com.br/esportes/dia-do-futebol-como-a-tecnologia-e-usada-no-futebol-144191/>. Acesso em: 10 outubro 2019. Citado na página 29.

RODRIGUES, F. X. F. et al. futebol americano no país do futebol: o caso do cuiabá arsenal. **Barbarói**, v. 2, n. 41, p. 227–247, 2014. Citado na página 23.

SANTOS, M. K. et al. Inteligência artificial, aprendizado de máquina, diagnóstico auxiliado por computador e radiômica: avanços da imagem rumo à medicina de precisão. **Radiologia Brasileira**, SciELO Brasil, n. AHEAD, 2019. Citado na página 35.

SCHATTENBERG, L. D. Tecnologias esportivas auxiliando no esporte. **REAVI-Revista Eletrônica do Alto Vale do Itajaí**, v. 2, n. 2, p. 149–152, 2013. Citado na página 29.

SILVA, M. P. d. **Processamento de consultas por similaridade em imagens médicas visando à recuperação perceptual guiada pelo usuário**. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado na página 42.

SILVA, R.; LOPES, J.; ARAÚJO, F. Visão computacional em python utilizando as bi-bliotecas scikit-image e scikit-learn. 2017. Citado na página 31.

SOARES, M. d. S.; CALDEIRA, V. Metodologias ágeis. **Extreme Programming**, 2004. Citado na página 45.

SOMMERVILLE, I. **Software engineering**. [S.l.]: Addison-wesley, 2011. Citado na página 44.

SONGINI, M. L. Put in plain language: The high portable, object-oriented python language moves into enterprise application development. **Computerworld**, v. 12, 2005. Citado na página 49.

SOUZA, T.; CORREIA, S. Estudo de técnicas de realce de imagens digitais e suas aplicações. **João Pessoa. Paraíba**, 2007. Citado na página 33.

SUGIMORI, Y. et al. Toyota production system and kanban system materialization of just-in-time and respect-for-human system. **The International Journal of Production Research**, Taylor & Francis, v. 15, n. 6, p. 553–564, 1977. Citado na página 46.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. [S.l.]: Springer, 2010. Citado 2 vezes nas páginas 43 e 53.

TECMUNDO. **O que são Frames por Segundo?** 2011. Disponível em: <https://www.tecmundo.com.br/video/10926-o-que-sao-frames-por-segundo-.htm>. Acesso em: 15 outubro 2019. Citado na página 56.

TORVALDS, L.; HAMANO, J. Git: Fast version control system. **URL http://git-scm. com**, 2010. Citado na página 52.

VENTURA, M. M. O estudo de caso como modalidade de pesquisa. **Revista SoCERJ**, v. 20, n. 5, p. 383–386, 2007. Citado na página 53.

VENTURA, V. d. A. **Representação de Imagens Através de Grafos Utilizando o Algoritmo Split And Merge Combinado Com Descritores de Cor e Textura**. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2009. Citado na página 39.

VIOLA, P.; JONES, M. et al. Rapid object detection using a boosted cascade of simple features. **CVPR (1)**, v. 1, n. 511-518, p. 3, 2001. Citado 2 vezes nas páginas 41 e 42.

WILSON, P. I.; FERNANDEZ, J. Facial feature detection using haar classifiers. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 21, n. 4, p. 127–133, 2006. Citado na página 41.

Apêndices

APÊNDICE A – PRIMEIRO APÊNDICE

Testando

Anexos

ANEXO A – PRIMEIRO ANEXO.

Testando