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.

Orientador: 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-
67p. : il. (algumas color.) ; 30 cm.

Orientador: Esp. Tales Wallace Souza

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

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

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 na Faculdade Doctum de Ipatinga – Rede Doctum de Ensino, em 2018.

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: Giovani Scarabelli Silva
Especialista em Administração de Sistema de Informação - UFLA
Instituto Superior Doctum de Ipatinga

DEDICATÓRIA

Dedico 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 QUADROS

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*
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*
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 cada vez à eficiência de uma visão humana.

1.1 Objetivo geral

Apresentar uma ferramenta que seja capaz de identificar um jogador 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

- Capturar, através de um dispositivo de entrada de vídeo, as imagens de uma partida de futebol americano.
- Processar as imagens capturadas e definir um modelo de busca de um jogador.
- Identificar o jogador seguindo o modelo.
- 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, várias aplicações que utilizam a tecnologia de visão computacional estão sendo criadas de forma singular e concisa.

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 ou 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.

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 forma necessários para o entendimento da área de visão computacional. No capítulo 3 foi abordado a metodologia utilizada para a construção deste trabalho. Subsequente, o capitulo 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 em jogadas de uma determinada briga por jardas. Cada jogada consiste em 4 tentativas de ganhar um valor determinado a equipe que detém a posse da bola. 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 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.

Os dados obtidos pelos times são destacados e analisados, podendo então 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, a chance de ter os melhores atletas em um único lugar para observar é o melhor modo para que treinadores, dirigentes e olheiros das equipes podem ter ao menos uma ideia de quais jogadores tem potencial de agregar mais desempenho ao time. 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 - (National Football League)*. 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*, 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 Técnologias 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.

2.1.2 O mercado do futebol americano

https://forbes.uol.com.br/fotos/2018/02/super-bowl-as-cifras-do-maior-evento-esportivo-do-mundo/

https://exame.abril.com.br/blog/esporte-executivo/o-futebol-americano-quer-ser-grande-no-brasil/

https://gauchazh.clicrbs.com.br/esportes/noticia/2017/03/as-10-melhores-contratacoes-no-mercado-da-nfl-9748745.html

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 utiliza matrizes bidimensionais ou hiperdimensionais como entrada de dados e, a partir desta, produzem informações compactadas como saída.

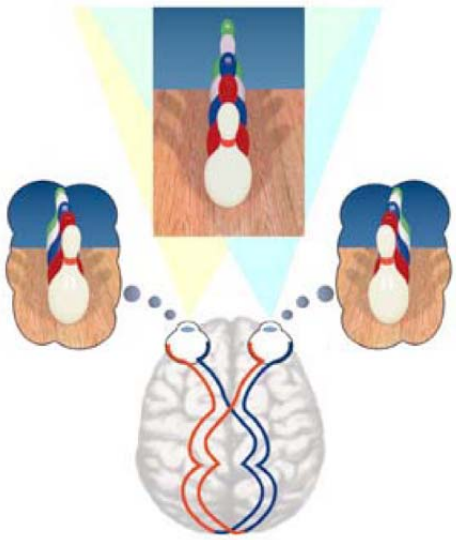
De forma didática, a área de visão computacional utiliza modelos descritivos de objetos, pessoas e/ou cenas capturadas digitalmente para realizar tomadas de decisões, a fim de automatizar processos.

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.

Contudo, 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. Peronti (2008) explica em seu artigo que estereoscopia

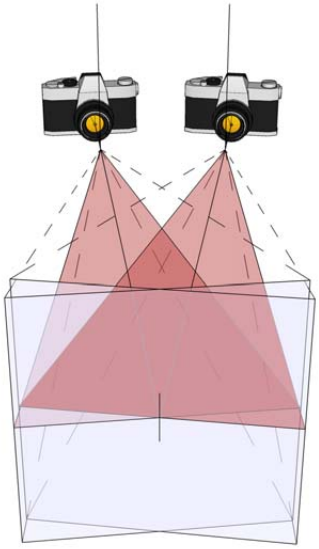
é a visualização feita por dois mecanismos de captura de imagem em um mesmo foco (Figura 2).

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



Fonte: (PERONTI, 2008)

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



Fonte: (PERONTI, 2008)

2.2.1 Captura e identificação de imagem

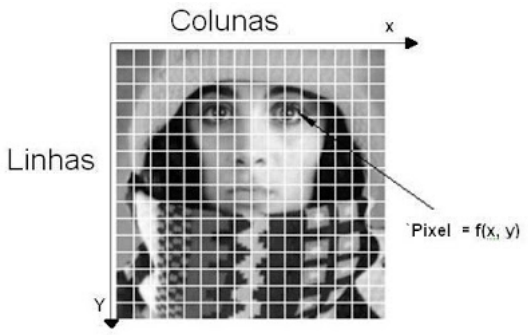
A tecnologia vem crescendo cada vez mais em todo mundo, proporcionando a ampliação do uso de dispositivos de alto desempenho. Paralelo a isso, o fluxo de dados aumenta proporcionalmente, exigindo uma velocidade de conexão maior com a Internet. Um exemplo disso são os *smartphones*, que se popularizaram 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.s



Disponível em: http://sdbmcc.blogspot.com/2009/06/imagem-digital-teoria.html

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

De forma complementar ao assunto supracitado, o processamento de imagem evoluiu drasticamente no decorrer do avanço tecnológico. Portanto, há grande interesse nessa tecnologia pelo fato de proporcionar um grande número de aplicações, como por exemplo o aprimoramento de informações relativo a imagens para interpretação humana e análise automáticas por computador de informações extraídas da imagem capturada.

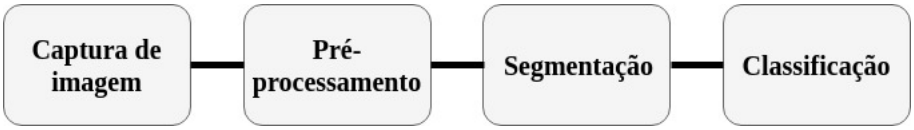
Contudo, o grande marco da área de processamento de imagens aconteceu no século XX. Com o surgimento dos primeiros computadores digitais com grande capacidade de processamento e o início do programa espacial norte-americano, ocorreu um grande impulso na área de processamento de imagem. O uso de técnicas 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)

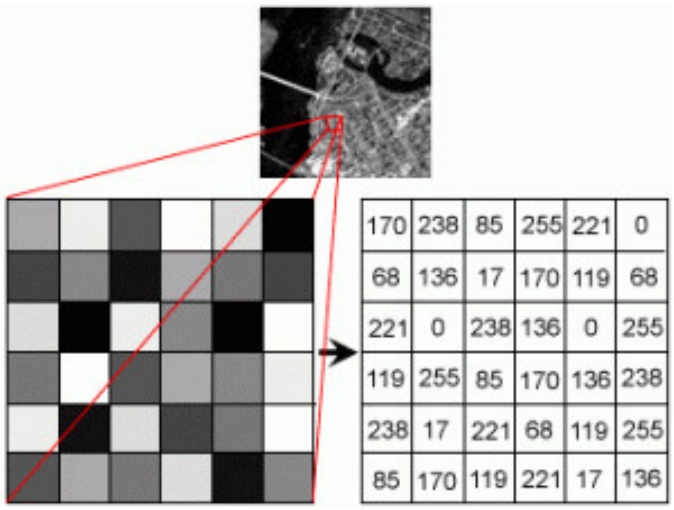
Seres humanos conseguem distinguir vários padrões de cores com certa facilidade, levando em consideração que a análise é feita em um ambiente tridimensional. Na computação, esse discernimento de cores são mais complexos, independentemente da dimensão na qual a imagem será analisada. Isso ocorre porque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

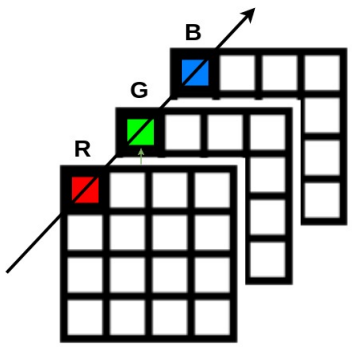
Figura 5 – Visão computacional de *pixel* monocromático.



Disponível em: https://ai.stanford.edu/ syyeung/cvweb/tutorial1.html

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 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.

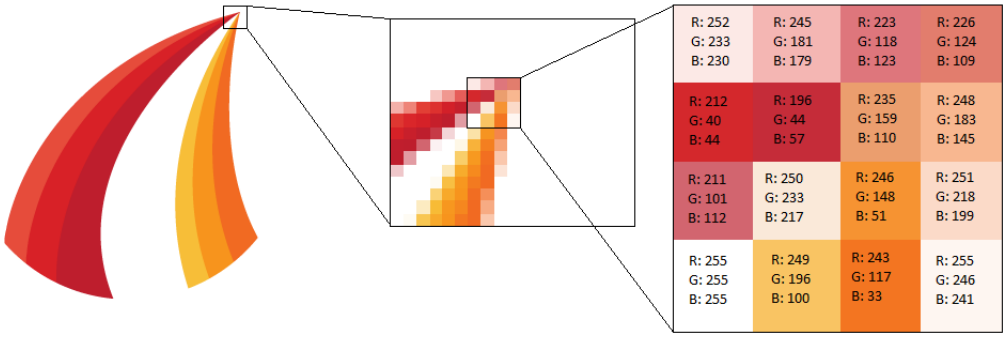


Fonte: Elaborada pelos autores do projeto.

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 – Visão computacional de *pixels* RGB.



Disponível em: https://www.analyticsindiamag.com/computer-vision-primer-how-ai-sees-an-image/

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. O resultado desse processo gera imagens digitalizadas de qualidade melhor que a original.

2.2.2.1 *Segmentação de imagens*

De forma a dar continuidade ao ciclo de processamento de imagem e verificação desta, usa-se métodos de *pixels* e tonalidades seguindo os padrões propostos. A análise deve prosseguir de forma minuciosa para então, processar seus detalhes.

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 de interesse dentro desta (FILHO; NETO, 1999).

No entanto, a segmentação exige um processamento muito grande para analisar a imagem sugerida. 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.

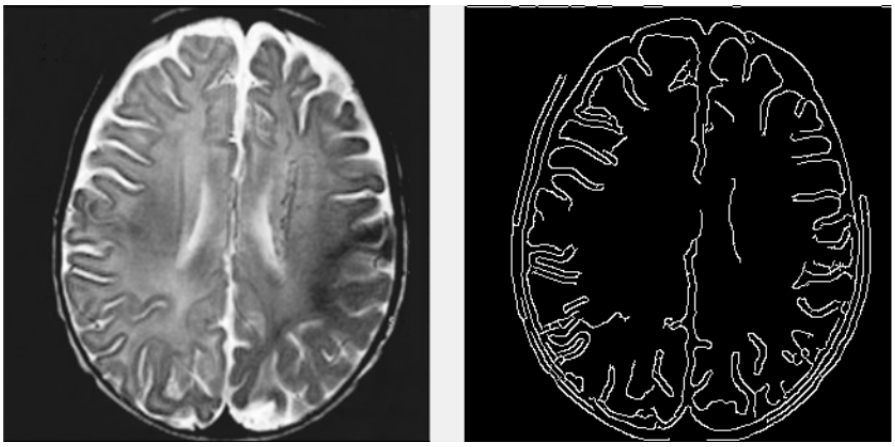
Para complementar, Morgan et al. (2008) enfatiza que existem métodos que são classificados de acordo com a representação dos objetos a serem segmentados, que são os métodos de borda ou orientados a regiões.

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* e *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 ate 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. Apos 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*

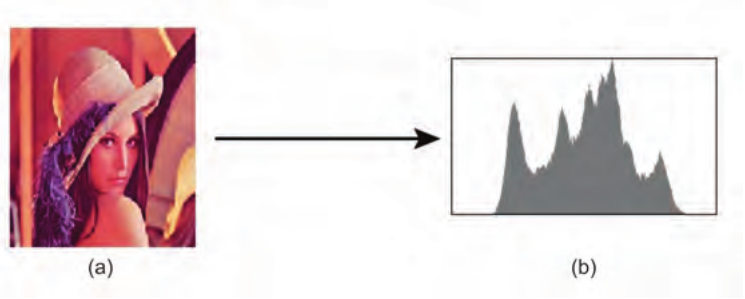
De tal forma, dando continuidade ao assunto, o histograma é um método que auxilia na identificação de objetos e/ou características específicas da imagem, obtendo uma maior precisão nos resultados obtidos.

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*

Como já mencionado, a classificação de imagem é a ultima etapa do processamento de imagem. 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*

Para complementar o assunto citado acima, 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*

De forma concisa, 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 consegue 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

Dentre as diversas tarefas que os computadores podem executar, o reconhecimento facial tem tido uma crescente, se tornando alvo de vários estudos. Algoritmos de reconhecimento facial estão presente em diversos dispositivos, como por exemplo *smartphones* e câmeras digitais, e ate 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 a 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 é bem 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. O objeto na qual esta sentá 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 intensificado quando a imagem é capturada de forma frontal as pessoas, no qual seja possível localizar por completo os rostos das pessoas em questão. Porem, 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 ferramente 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 *softwares*, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais:
>
> - Indivfduos 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 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 pode intervir para que um *software* não possa ser desenvolvido da maneira correta, como por exemplo uma má equipe de trabalho, nos quais os integrantes não possuem uma boa convivência entre si, ou uma má metodologia de desenvolvimento sem uma linha cronológica a ser seguida. 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.

2.3.1.1 *Kanban*

De forma a complementar o assunto, o *Kanban* é um método ágil 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 contem 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).

Sendo assim, o *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.

O *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 metodo mais aprimorado de organização dentro da empresa *Toyota*, denominado *Kanban* (SUGIMORI et al., 1977).

Atualmente, a metodologia ágil *Kanban* é utilizada em diversas empresas para realizar o

controle de desempenho de diversas área de atuação.

2.3.1.2 *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 notifica a todos os membros a situação de suas atividades dentro do projeto, bem como seu *status* e prazo.

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 propositos gerais de alto nível e de código aberto, objetivando uma construção ágil no desenvolvimento de aplicações. Sua sintaxe e 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 *Framework*

Atualmente, os projetos de desenvolvimento de *software* aumentou consideravelmente. Devido a isso, programadores se deparam com um excesso de funções similares dentro dos vários projetos desenvolvidos no decorrer do tempo. Sendo assim, surgiu a necessidade de reutilizar códigos para poupar tempo.

De acordo com Johnson (1997), *framework* são estruturas desenvolvidas com o objetivo de reutilizar tudo ou parte de um sistema representado por um conjunto de classes abstratas e concretas, ou seja, uma estrutura parcialmente completa projetada para ser instanci-

ada. Existe também *frameworks* que disponibilizam *templates* como base para iniciar o desenvolvimento.

Os *frameworks* disponibilizam para os desenvolvedores um vasto conjunto de bibliotecas, permitindo assim a realização de operações de grande porte. Sendo assim, os programadores focam mais em resolver problemas do que reescrever códigos, aumentando consideravelmente a produtividade da equipe.

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 gerencia e controla 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 comandos, 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 reposi-

tórios *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.

2.4.4 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*, no qual será utilizada para o desenvolvimento de uma possível resolução ao problema supracitado. 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.

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)."

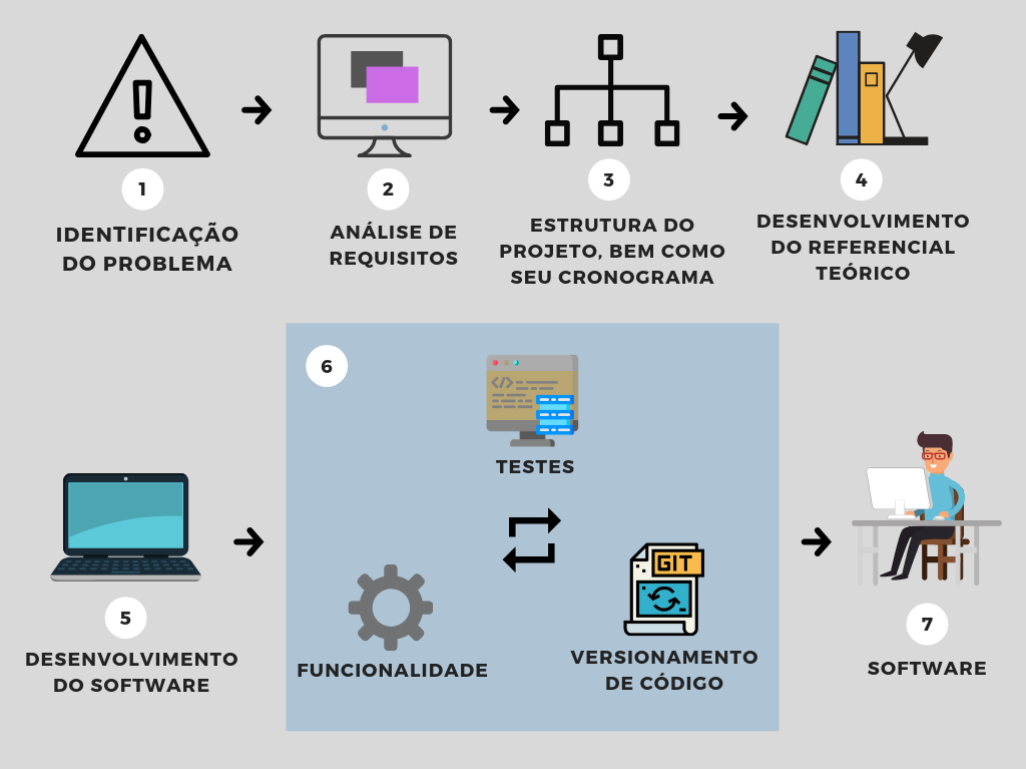
Sendo assim, será feito um estudo bibliográfico a partir do tema de visão computacional utilizando o acervo disponível sobre o assunto, com a finalidade de reunir informações para obter melhores resultados.

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 a elaboração deste projeto, foi feito uma imagem ilustrativa da metodologia utilizada no decorrer das etapas de desenvolvimento deste (Figura 10).

Figura 10 – Metodologia de desenvolvimento do projeto.



Fonte: Elaborada pelos autores do projeto.

4 DESENVOLVIMENTO

Neste capítulo aborda todo o processo utilizado para o desenvolvimento desse projeto...

4.1 Descrição do sistema

O QUE O SOFTWARE VAI FAZER

4.1.1 Etapas de funcionamento

COMO ELE VAI FAZER

4.2 Desenvolvimento do sistema

COMO FOI FEITO

4.2.1 Requisitos funcionais

4.2.2 Requisitos não-funcionais

4.3 Ambiente de testes

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

BARBOZA, D. P. d. S. Estudo da biblioteca opencv. Universidade Federal do Rio de Janeiro, 2009. Citado na página 49.

BASS, T. **Football Skills & Drills**. Human Kinetics, 2012. (Skills and Drills Series). ISBN 9780736090766. Disponível em: <https://books.google.com.br/books?id=f73_mgEACAAJ>. Citado na página 28.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006. Citado na página 40.

CHACON, S.; STRAUB, B. **Pro git**. [S.l.]: Apress, 2014. Citado na página 48.

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 na página 32.

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 49.

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 38, 40 e 41.

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

FILHO, O. M.; NETO, H. V. **Processamento digital de imagens**. [S.l.]: Brasport, 1999. Citado 4 vezes nas páginas 32, 36, 38 e 51.

FREITAS, R. P. d. S. Sistema para detecção de bordas em exames clínicos. Universidade Federal Fluiminense, 2016. Citado na página 37.

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 33, 36 e 51.

JOHNSON, H. A. Trello. **Journal of the Medical Library Association: JMLA**, Medical Library Association, v. 105, n. 2, p. 209, 2017. Citado na página 45.

JOHNSON, R. E. Frameworks=(components+ patterns). **Communications of the ACM**, Citeseer, v. 40, n. 10, p. 39–42, 1997. Citado na página 46.

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 46.

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 na página 39.

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 48.

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 34.

MAIA, L. C. G.; SOUZA, R. R. Medidas de similaridade em documentos eletrônicos. 2013. Citado na página 41.

MARR, D. From computational theory to psychology and neurophysiology–a case study from vision. MIT Artificial Intelligence Laboratory, 1976. Citado na página 29.

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 39.

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 34.

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 36 e 37.

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 46.

OPENCV. **OpenCV - Open Source Computer Vision**. 2019. Disponível em: https://opencv.org Acessado em: 15 set. 2019. Citado 2 vezes nas páginas 40 e 49.

PERONTI, W. L. Princípios teóricos da estereoscopia. 2008. Citado 3 vezes nas páginas 29, 30 e 31.

PILGRIM, M.; WILLISON, S. **Dive Into Python 3**. [S.l.]: Springer, 2009. v. 2. Citado na página 46.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [S.l.]: McGraw Hill Brasil, 2016. Citado na página 42.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. [S.l.]: Bookman Editora, 2014. Citado na página 44.

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 33.

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 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 41.

SILVA, R.; LOPES, J.; ARAÚJO, F. Visão computacional em python utilizando as bi-bliotecas scikit-image e scikit-learn. 2017. Citado na página 29.

SOARES, M. d. S.; CALDEIRA, V. Metodologias ágeis. **Extreme Programming**, 2004. Citado 2 vezes nas páginas 43 e 44.

SOMMERVILLE, I. **Software engineering**. [S.l.]: Addison-wesley, 2011. Citado na página 43.

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 46.

SOUZA, T.; CORREIA, S. Estudo de técnicas de realce de imagens digitais e suas aplicações. **João Pessoa. Paraíba**, 2007. Citado na página 31.

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 44.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. [S.l.]: Springer, 2010. Citado 3 vezes nas páginas 41, 42 e 51.

TORVALDS, L.; HAMANO, J. Git: Fast version control system. **URL http://git-scm. com**, 2010. Citado na página 48.

VENTURA, M. M. O estudo de caso como modalidade de pesquisa. **Revista SoCERJ**, v. 20, n. 5, p. 383–386, 2007. Citado na página 51.

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 37.

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 na página 40.

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 40.

Apêndices

APÊNDICE A – PRIMEIRO APÊNDICE

Testando

Anexos

ANEXO A – PRIMEIRO ANEXO.

Testando