FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO

EURIPEDES PURCINIO FERNANDES NETO
RA: 21500709

VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE CORES EM TEMPO REAL COM OPENCV E PYTHON

BRASÍLIA
2020

EURIPEDES PURCINIO FERNANDES NETO

VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE CORES EM TEMPO REAL COM OPENCV E PYTHON

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia de Computação do UniCEUB– Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier De Obaldía Díaz**

BRASÍLIA
2020

EURIPEDES PURCINIO FERNANDES NETO

VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE CORES EM TEMPO REAL COM OPENCV E PYTHON

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia de Computação do UniCEUB – Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier De Obaldía Díaz**

Brasília, 2020.

BANCA EXAMINADORA

Prof. MsC Francisco Javier De Obaldía Díaz
Orientador (a)

Me. Ivandro da Silva Ribeiro
Examinador (a)

Me. Luis Claudio Lopes de Araujo
Examinador (a)

Visão Computacional para Identificação de Cores em Tempo Real com OpenCV e Python
Computer Vision for Color Identification in Real Time with OpenCV and Python

Euripedes Purcinio Fernandes Neto[1], Francisco Javier de Obaldía Díaz [2], Ivandro da Silva Ribeiro [3], Luis Claudio Lopes de Araujo [4]

RESUMO

Em uma realidade em que a humanidade vive o desafio de entender e atender as necessidades de aproximadamente oito bilhões de pessoas e ao mesmo tempo mitigar os impactos sociais, econômicos e ambientais decorrentes da existência humana, surge a tecnologia, para apoiar estas necessidades. A visão computacional tem por objetivo que computadores possam entender e processar o conteúdo de imagens digitais, como fotografias e vídeos, para assim auxiliar no desenvolvimento humano. A identificação de cores pode ser um processo trivial quando realizado pelo olho humano em suas perfeitas condições, mas há casos onde o indivíduo possui uma anomalia nos glóbulos oculares que o impossibilita de identificar cores. Neste contexto, o daltonismo, uma limitação visual comumente encontrada na sociedade, pode ter seus danos minimizados e contornados com a ajuda de tecnologias de captação e processamento de imagens em tempo real. A proposta abordada baseia-se em identificar as cores de forma prática, através de um algoritmo que realiza o processamento digital de imagens e o demonstra em tempo real.
PALAVRAS-CHAVE: Visão Computacional, Identificação de Cores, Processamento Digital de Imagem.

ABSTRACT

In a reality in which life faces the challenge of understanding and meeting the needs of approximately eight billion people and at the same time mitigating the social, economic and environmental impacts resulting from human existence, technology appears to support these needs. Computer vision aims to enable computers to understand and process the content of digital images, such as photographs and videos, in order to assist in human development. The identification of nuclei can be a trivial process when performed by the human eye in its perfect conditions, but in cases where the individual has an anomaly in the eyeballs that makes it impossible to identify nuclei. In this context, color blindness, a visual limitation commonly found in society, can have its damage minimized and circumvented, with the help of technologies for capturing and processing images in real time. The proposal approached is based on identifying as nuclei in a practical way, through an algorithm that digital image processing and demonstration in real time.
KEYWORDS: *Computer Vision, Color Identification, Digital Image Processing.*

[1] UniCEUB, aluno.
[2] UniCEUB, orientador.
[3] UniCEUB, primeiro examinador.
[4] UniCEUB, segundo examinador.

1 INTRODUÇÃO

O termo utilizado para definir qualquer que seja o tipo de defeito na identificação de cores é “discromatopsia”. Ela contem diferentes niveis e caracteriristicas, sendo eles a Monocromia, Tritanopia, Protanopia e Deuteranopia. Porém, a expressão mais comumente empregada como sinônimo da discromatopsia é “daltonismo”, tendo como referência o químico John Dalton (1766-1844), já que este possuía protanopia e foi o primeiro cientista que estudou o assunto. Essas desordens podem ser tanto congênitas, quanto consequência de doenças oculares ou sistêmicas e traumas. Estima-se que as discromatopsias de origem genética acometem de 0,4 a 0,7% das mulheres e 6% a 10% dos homens, na população geral (MELO, GALON, FONTANELLA, 2014).

A visão computacional é uma forte aliada na identificação e processamento de imagens para facilitar e suprir as limitações humanas, ela faz com que o computador recolha informações de determinada imagem, e isso se torna possível através de algoritmos de processamento de imagens. Com a necessidade de melhorar a informação visual e de executar a percepção dos dados das imagens por meio de máquinas, surgiu o interesse por esses algoritmos (GEISLER, 2006). Elaborando-se modelos algorítmicos para interpretar imagens, a visão computacional torna possível incorporar fotografias e vídeos, dentre outros tipos de registo, como entrada de dados para a ciência forense, navegação de robôs, sistemas de gestão de informações, vigilância e aplicações bélicas (SILVA et al, 2020).

A visão computacional também pode ser estabelecida, segundo Alves (2005), como sendo a junção de técnicas computacionais utilizadas para explicitar ou estimar propriedades dinâmicas e geométricas da realidade tridimensional por meio de imagens. Câmeras, ao invés dos olhos, são empregadas para se obter as imagens digitais. Nestas, são executadas técnicas computacionais a fim de extrair as informações pretendidas do mundo tridimensional. Como essas informações sofrem variações de natureza, podem ser aplicadas em diferentes ações e áreas. A visão computacional busca obter dados de cenas através de imagens digitais capturadas previamente. Para que isso seja possível, quase sempre faz-se uso de metodologias de processamento de imagens. Para concluir, a visão computacional tem por objetivo deduzir e descrever automaticamente propriedades e estruturas do mundo 3D, que pode ser dinâmico, com base em várias imagens do mundo em 2D, as quais podem ser monocromáticas ou coloridas, sendo capturadas por apenas um ou diversos sensores, que podem ser móveis ou estacionários.

No processamento digital de imagem, normalmente, é difícil realizar a separação da imagem do plano de fundo, fazendo com que a utilização de técnicas como regularização e modelagem tenham que ser utilizadas. Com base na forma geométrica dos objetos, como resultado da segmentação, podem ser usados operadores morfológicos com o objetivo de modificar e analisar essa forma, assim como extrair dados adicionais do objeto (QUEIROZ, GOMES, 2001). Para facilitar a aplicação destas técnicas, foi desenvolvida a biblioteca OpenCV, que possui diversos algoritmos de processamento digital de imagem.

A biblioteca OpenCV possui mais de 2.500 algoritmos otimizados, que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina clássicos e de última geração. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmeras, rastrear objetos em movimento etc (OPENCV, 2020).

Para realizar qualquer processamento de imagem, são necessários mecanismos e algoritmos programáveis. Em face ao exposto, surge o questionamento: é possível realizar a identificação de cores no mundo

real através de uma câmera?

Desta forma, este trabalho pretende apresentar o desenvolvimento de um algoritmo em forma de protótipo, que identifique as cores primárias e secundárias em tempo de execução da câmera, e demonstre as cores captadas no display. Assim, tornando possível a identificação de cores por pessoas com limitações visuais através de uma simples câmera.

Inicialmente, é apresentado o referencial teórico, explicando os conceitos envolvidos no desenvolvimento do algoritmo, presente na seção 2. Em seguida, a metodologia utilizada na elaboração do trabalho, na seção 3. E, por último, os resultados e análises, na seção 4.

2 REVISÃO BIBLIOGRÁFICA

Neste tópico são apresentados os conceitos envolvidos no trabalho: visão computacional, processamento digital de imagens, biblioteca OpenCV, espaço de cores RGB e HSV.

2.1 Visão computacional

Para Dawson (2014), o termo visão computacional é atribuído à capacidade de análise automática de imagens e vídeos por computadores, com o foco em obter informações. Visão computacional consiste em extrair informações a partir de operações ou transformações feitas em dados, como imagens digitais, vídeos ou qualquer estrutura de dados multidimensional do mundo real (GARCÍA et al, 2015).

Os sistemas de visão computacional seguem um conjunto predefinido de etapas, aquisição, pré-processamento, segmentação, representação e descrição, reconhecimento e interpretação (GONZALEZ, WOODS, 2009).

O processo de aquisição consiste na captura da imagem por meio de um dispositivo ou um sensor, no caso deste trabalho, através de uma webcam acoplada ao computador, para, assim, se ter um formato adequado para sua manipulação. Nos aspectos relacionados nessa etapa, é denominada a configuração do dispositivo de captura, como o formato da imagem digital, as configurações de luminosidade, resolução, número de níveis de cinza ou cores da imagem digital. Na etapa de pré-processamento, o objetivo é a melhoria da qualidade da imagem vinda da aquisição, para isso se utiliza das técnicas de atenuação de ruídos, correção de contraste e brilho e equalização de histograma. Em seguida, temos a etapa de segmentação, a qual consiste no processo de separar a imagem em regiões de pixels similares (SONKA, 2014).

Os destaques na segmentação são selecionados com base na extração de características que possam ser usadas para distinguir classes de objetos. O processo de extração é denominado descrição e a estrutura que permite armazenar e manipular essa seleção é chamada de representação (SZELISKI, 2010).

Na última etapa, temos o reconhecimento ou classificação, que consiste no processo de atribuição de um identificador aos objetos da imagem, dada as características presentes nos seus descritores. Já a interpretação, atribui um significado aos objetos reconhecidos, isto é, dada a classificação, atribui um resultado ou decisão para a situação descoberta, podendo fazer uso de redes neurais e deep learning (DAVIES, 2017).

2.2 Imagem digital

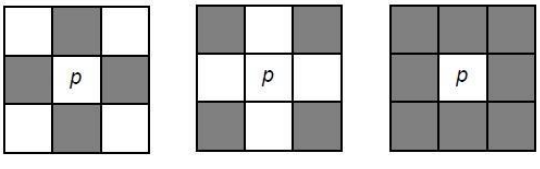
Uma imagem natural pode ser caracterizada por uma variação contínua de tons e cores. Os tons variam de claro a escuro e as cores variam de vermelho a azul, abrangendo, desta forma, todo o espectro de cores visíveis (QUEIROZ; GOMES, 2006). O que caracteriza uma imagem é que ela é formada por uma função bidimensional de intensidade da luz como f (x, y), onde x e y correspondem às coordenadas espaciais e o valor de f, em qualquer ponto x, y, é proporcional ao brilho (dado pelos níveis de

cinza) da imagem naquele ponto. Já se tratando de uma imagem digital, ela é composta por pontos discretos de tons, cores e brilho, e não por uma variação contínua. Para a criação de uma imagem digital deve-se dividir a imagem contínua em uma série de pontos que irão possuir uma determinada tonalidade ou cor (QUEIROZ; GOMES, 2006). Uma imagem digitalizada é uma matriz onde os índices de linhas e colunas identificam um determinado ponto na imagem e o correspondente valor deste elemento determina o brilho médio amostrado. Aos elementos desta matriz digital dá-se o nome de pixels (GONZALES, WOODS, 2009).

A representação da imagem pode ser feita computacionalmente, por meio de uma matriz $M(i, j)$ formada por um número de pixels finitos, que possuem um valor escalar e uma localização (i, j) a partir da origem da imagem. Cada pixel possui uma relação com seus vizinhos, sendo que cada pixel p possui quatro vizinhos horizontais e verticais, representados pelas coordenadas: $(x + 1, y)$, $(x - 1, y), (x, y + 1), (x, y - 1)$.

Esse conjunto de pontos é chamado de vizinhança - 4 de p, e pode ser expresso por $N4\ (p)$. Mas, também, deve-se considerar o conjunto de vizinhos diagonais de p, expresso por $ND\ (p)$, representado pelas coordenadas: $(x + 1, y + 1), (x + 1, y - 1), (x - 1, y - 1), (x - 1, y - 1)$. Nessa mesma linha, há a vizinhança – 8, expressa pelo produto da vizinhança – 4 e da vizinhança diagonal, $N8\ (p) = N4\ (p) \cup ND\ (p)$. Na figura 1, tem-se as representações de cada modelo, sendo (1) $N4\ (p)$, (2) $ND\ (p)$ e (3) $N8\ (p)$.

Figura 1. Vizinhança de um pixel



(1) (2) (3)

Fonte: UFU (2014)

2.2.1 Processamento de imagem digital

Processar uma imagem consiste na sua transformação constante até que se chegue no esperado. Para isso, se faz uso de técnicas para filtrar objetos e descartar dados irrelevantes, que são comumente chamados de ruídos. Gonzalez (2009) explica que o interesse desses métodos tendem de duas áreas, uma delas é a melhoria de informação visual para a interpretação humana, e a outra é quanto ao processamento de dados de imagens para a automação por máquinas.

O processamento de imagens está diretamente ligado à visão computacional, que tem como objetivo, aliada às técnicas de inteligência computacional, realizar tarefas semelhantes ao olho humano (SCHWARTZ, 2008), extraindo informações para um determinado intuito ou solução de problema, que serão analisadas e aplicadas à problemática.

As técnicas de processamento digital de imagem fundamentais para a construção deste projeto são as de limiarização e morfologia.

2.2.1.1 Limiarização

Limiarização é nome dado à técnica utilizada no contexto da segmentação de que é fundamentada na análise da similaridade de níveis de cinza de um imagem. Dada esta imagem com um plano de fundo escuro e um conjunto de objetos iluminados sobre este fundo, a partir da análise do histograma da imagem, pode-se verificar que os pixels dos objetos e do plano de fundo são agrupados em diferentes intervalos de níveis de cinza, estabelecendo, assim, entre eles, um limiar ou uma fronteira de decisão (GONZALEZ, WOODS, 2009).

Uma maneira de se extrair estes objetos das imagens seria a definição de um limiar, chamado T. Sendo f (x, y) uma imagem contendo apenas um objeto, e supondo que f (x, y) > T, logo, este pixel seria pertencente ao conjunto de pixels que definem o objeto, e o

seu inverso seria um ponto do plano de fundo. Assim, o processo de segmentação efetua a varredura em toda a imagem, comparando cada pixel com o valor T, classificando-o como 1 para o objeto e 0 para o plano de fundo, gerando uma imagem binária, onde se define com clareza a separação da região que define este objeto.

2.2.1.2 Morfologia

As transformações morfológicas são operações baseadas na forma da imagem. Comumente executadas em imagens binárias, é um processo que precisa de duas entradas, uma é a imagem original, e a segunda é chamada de elemento estruturante ou kernel, que decide a natureza da operação (OPENCV, 2020). Os dois operadores morfológicos básicos são erosão e dilatação, e destes dois processos surgem suas variantes, como abertura, fechamento, gradiente etc.

A ideia básica de erosão é como a erosão do solo, ela desgasta os limites do objeto em primeiro plano, baseados na limiarização. O kernel desliza pela imagem como na convolução 2D. Um pixel, na imagem original registrado como 1 ou 0, será considerado 1 apenas se todos os pixels sob o kernel forem 1, caso contrário, ele é corroído, tendo seu valor alterado para 0.

Neste processo, todos os pixels próximos ao limite serão descartados dependendo do tamanho do kernel utilizado. Portanto, a espessura do objeto em primeiro plano na imagem diminui. A erosão é utilizada para remover pequenos ruídos no objeto (OPENCV, 2020).

A dilatação é exatamente o oposto da erosão. Nela, um elemento de pixel é '1' se pelo menos um pixel sob o kernel for '1'. Portanto, aumenta a região de borda na imagem, isto é, o tamanho do objeto em primeiro plano. É bastante utilizada para unir partes quebradas de um objeto.

2.3 OpenCV

O OpenCV consiste em uma plataforma que contém um conjunto de bibliotecas de código livre para o desenvolvimento de softwares de visão computacional e aprendizado de máquinas. Desenvolvida originalmente pela Intel, para fornecer uma infraestrutura comum para as aplicações de visão computacional para pesquisa e uso comercial. No total, as bibliotecas do OpenCV contam com mais de 2700 algoritmos otimizados para serem utilizados na detecção e reconhecimento de faces, identificação de objetos, classificação de ações humanas em vídeos, movimentação de câmeras e de objetos, modelos 3D, trabalhos com visão estéreo, entre outras ações relacionadas à visão computacional (BARELLI, 2018). Desenvolvido inicialmente na linguagem de programação C++, hoje tem suporte para Python, Java, interfaces MATLAB e pode ser executado nos sistemas operacionais Windows, Linux, Android and Mac.

Largamente utilizado no mercado por diversas empresas do topo da economia digital, como Google, Microsoft, Intel, IBM, Sony e Toyota (BRADISK, 2017), o OpenCV possui implementações para as mais diversas técnicas de processamento digital de imagem, dentre eles, temos a transformação de espaço de cores e detectores de borda, que são os principais recursos para este artigo registro, entre outros (VETRISELVAN, 2017). Dada a diversidade de algoritmos contidos no OpenCV, essa biblioteca foi escolhida como pacote base de visão computacional deste projeto.

2.4 Espaço de cores

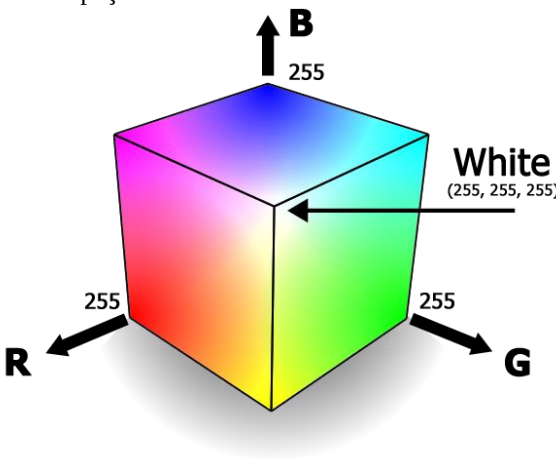
Para o processamento e análise de imagens, as cores podem ser um ótimo parâmetro, podendo simplificar a identificação de objetos na cena. Dado que a percepção de cores é um fenômeno fisiopsicológico ainda em discussão pelos

especialistas, a característica física das cores pode desenvolver resultados experimentais e teóricos (GONZALES, WOODS, 2007). Baseado nestes resultados, foram definidos espaços de cores que facilitam a identificação delas em um determinado padrão. Existem vários tipos de espaço de cores, sendo os mais comuns o RGB e o HSV no processamento digital de imagens.

2.4.1 Espectro RGB

Na figura 2, é mostrado que o espaço de cor RGB (Red, Green, Blue) é um padrão de representação de cores no qual cada cor é expressa pela combinação de três cores primárias: vermelho, verde e azul (GONZALES, WOODS, 2009).

Figura 2. Espaço RGB



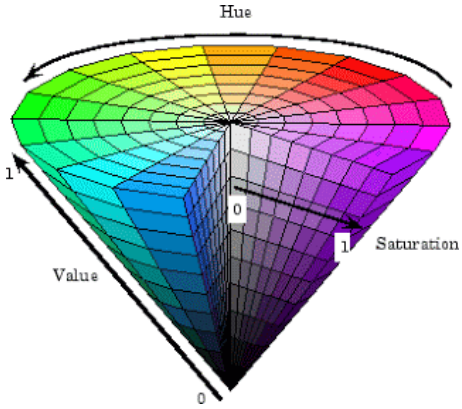
Fonte: USP (2018)

As imagens em RGB, conhecidas como imagens de cores reais, são representadas matematicamente por três matrizes bidimensionais distintas, sendo uma para cada canal. Cada elemento dessas matrizes é um número inteiro, que representa a intensidade da cor de um pixel (BARELLI, 2018). Logo, cada ponto contém três valores que representam a intensidade das cores vermelho, verde e azul, que juntas geram a cor naquele pixel. Cada canal é representado por 8 bits, e, como as três cores têm valores inteiros de 0 a 255, há um total de combinações possíveis de cores de $(2^8)^3$ = 16.777.216, demonstrado na Figura 3.

2.4.2 Espectro HSV

O Espectro HSV, também chamado de HSB (Hue, Saturation e Brightness) é formado por três componentes: Hue, Saturation e Value, traduzindo para o português tem-se, respectivamente, Matiz, Saturação e Valor, conforme a Figura 2. Sendo matiz a cor propriamente dita, saturação é o nível de pureza dessa cor, quanto menor a saturação mais clara essa cor será. O componente valor ou brilho está relacionado com a quantidade de brilho ou luminosidade contida na imagem. Esse sistema foi desenvolvido em meados de 1970, por pioneiros da computação gráfica que trabalhavam na PARC e NYIT, e foi formalmente descrito por Alvy Ray Smith e publicado oficialmente pela Microsoft (MATHWORKS, 2012).

Figura 3. Espectro de cores HSV



Fonte: MatLab (2020)

O espaço de cor HSV pode ser obtido pela seguinte forma: com a posse de uma coordenada angular, que representa a matiz ou as cores propriamente ditas, tem-se uma coordenada radial que dá a saturação, e, também, um eixo vertical associado à quantidade valor. A partir desta descrição, chega-se a uma geometria cônica. Para o processamento digital de máquina, os valores obtidos no campo do HSV podem ser mais precisos comparados ao RGB, pois o seu

valor é mais adequado para fazer o comparativo na escala de cinza (GONZALES, WOODS, 2009).

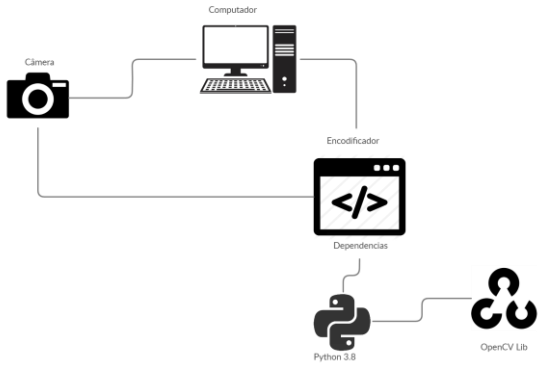
3 METODOLOGIA DO TRABALHO

Esta tese é caracterizada por um estudo exploratório e fará uso da pesquisa do tipo qualitativa, realizando a captação de imagens e o processamento, a fim de identificar as cores primárias e secundárias em tempo real, através de uma câmera acoplada a um computador, visto que o algorítimo deve ser capaz de rodar em dispositivos de baixa capacidade, a arquitetura do hardware e software disponíveis no computador foram limitadas ao uso de 2 GB de memória RAM e a capacidade de processamento restringida para 2 núcleos, para tornar um ambiente de baixa performance, similar a um dispositivo mobile. Para tornar possível a execução da temática proposta, as seguintes etapas foram seguidas:

Durante a primeira etapa, foi realizada a busca de referências bibliográficas que deram suporte científico e técnico referente a visão computacional, biblioteca OpenCV e linguagem Python, que são o alicerce do desenvolvimento deste projeto.

Com base nos levantamentos bibliográficos, partiu-se para a segunda etapa, onde foi definida a topologia e os recursos a serem utilizados no projeto para delimitar o escopo. A primeira parte foi a configuração do ambiente, acoplando a câmera ao computador e instalando as dependências necessárias para o desenvolvimento do algoritmo, conforme explicitado na figura 4.

Figura 4. Topologia do projeto



Fonte: Autor (2020)

A câmera utilizada neste protótipo é a Webcam C270, que possui uma resolução de 720p e captura 30 quadros por segundo, vendo que se faz necessário o uso de uma interface para administração dos recursos necessários, foi instalado o Pychram, uma IDE de desenvolvimento gratuita.

Após a instalação da IDE, foi necessária a instalação da linguagem de programação Python e suas bibliotecas, que, neste projeto, foi utilizada a versão 3.8.5, conforme descrito na imagem abaixo.

Figura 5. Intalação e versionamento Python

```
C:\Users\Pichau>python - V
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul
 20 2020, 15:43:08) [MSC v.1926 32 bit
 (Intel)] on win32
Type "help", "copyright", "credits" or
 "license" for more information.
>>>
```

Fonte: Autor (2020)

Após a instalação da linguagem de desenvolvimento do protótipo, partiu-se para a instalação da dependência principal do projeto, o OpenCV, que contém os algoritmos necessários para as tratativas em visão computacional, e que, para sua instalação, basta a execução via comando no prompt. A versão instalada para este projeto foi a 4.4, que foi disponibilizada em 18/07/2020.

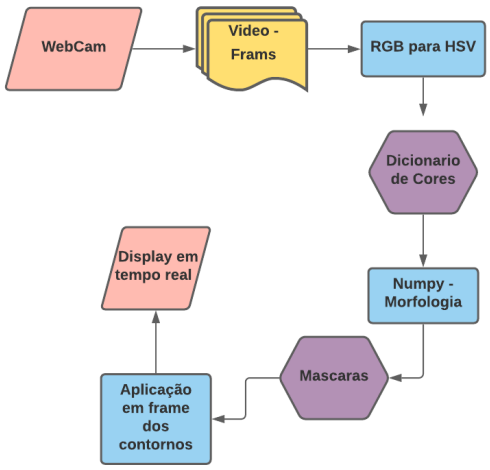
Figura 6. Instalação e versionamento OpenCv

```
C:\Users\Pichau>pip install opencv-python
Requirement already satisfied: opencv-pytho
n in c:\users\pichau\appdata\local\programs
\python\python38-32\lib\site-packages (4.4.
0.42)
Requirement already satisfied: numpy>=1.17.
3 in c:\users\pichau\appdata\local\programs
\python\python38-32\lib\site-packages (from
 opencv-python) (1.19.1)
```

Fonte: Autor (2020)

Com as dependências devidamente instaladas, pôde-se partir para a terceira etapa, onde foi desenvolvida a estrutura do código fonte do projeto, tendo como primeiro passo a definição do fluxo do programa, no qual foi realizado a captura da imagem através da câmera. Essas imagens são segmentadas em partes para poderem ser processadas, onde ocorre a conversão entre os espectros de cores da imagem e que será efetuado o processamento, para que, dessa forma, a imagem possa ser tratada e sejam aplicados os devidos processamentos em tempo real.

Figura 7. Fluxo do script



Fonte: Autor (2020)

Em posse do fluxo, partiu-se para a construção do código. Conforme descrito no fluxo, o início do programa é a aquisição das imagens através da webcam.

Figura 8. Aquisição das imagens

```
camera = cv2.VideoCapture(0)
while True:
    # leia o vídeo em tempo real
    _, frame = camera.read()
```

Fonte: Autor (2020)

As imagens captadas pela webcam são armazenadas na variável 'frame', e essa variável será a base para o armazenamento e processamento das imagens adquiridas.

De posse das imagens e com a utilização da biblioteca Imutilis nativa do Python, realiza-se o tratamento da imagem para a amostragem no display, através da função: imutils.resize, onde é redefinido o dimensionamento do display para 720 pixels, que é a capacidade máxima da webcam, evitando, assim, dilatações e ruídos na imagem.

O próximo passo do fluxo é a conversão da imagem obtida no padrão HSV. Essa conversão se faz necessária, pois o padrão de matrizes RGB define cada cor por uma matriz 3 por 1, e não é possível determinar um intervalo entre as cores, já que cada cor é representada por uma matriz distinta com a fusão das três colunas. Mas, na formação HSV, cada valor representa quanto daquela cor contém o objeto, tornando-se possível determinar um intervalo entre as cores.

Para essa conversão, utiliza-se a função cv2.COLOR_BGR2HSV, disponível nas bibliotecas do OpenCV. O cálculo dessa conversão pode ser dado pela equação contida na figura 9.

Figura 9. Fórmula de conversão RGB para HSV

$$H = \mathrm{Cos}^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

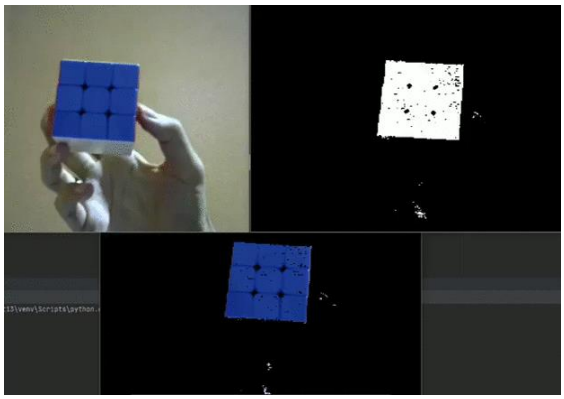
$$S = 1 - \frac{3}{R+G+B}\,[\min RGB]$$

$$V = \frac{1}{3}\,(R+G+B)$$

Fonte: IJSRCSEIT (2018)

Aplicando esta função, tem-se uma imagem descaracterizada e nos padrões de identificação de cores por ‘range’, o denominado intervalo entre as cores. Após essa descaracterização do padrão RGB, já é possível a aplicação de um dicionário de cores a serem identificadas pelo algoritmo. A figura 10 mostra a imagem em padrão HSV processada, que pode ser facilmente identificada:

Figura 10. Conversão RGB para HSV

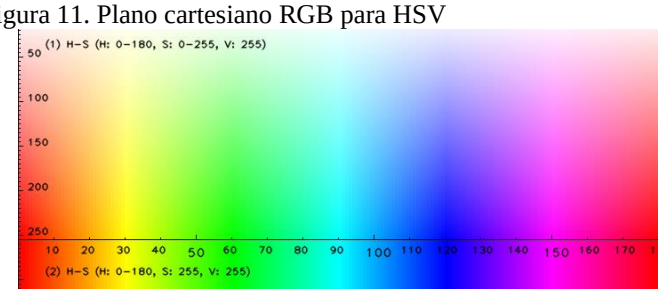


Fonte: Autor (2020)

Este caso de uso foi efetuado configurado para o reconhecimento da cor azul, a fim de exemplificar o processo de transformação entre os espaços de cores. No primeiro quadro, tem-se a imagem real captada pela câmera. Já no segundo quadro, é a imagem convertida para o padrão HSV e seccionada para a identificação entre o ‘range’ (110,50,50) e (130,255,255), que é dominante da cor azul, gerando uma máscara de detecção no objeto. No terceiro quadro, a imagem original é projetada na máscara encontrada, tornando, assim, visível apenas a cor selecionada no dicionário.

Com a imagem sendo captada pela webcam, segmentada por frames e convertida para o formato HSV, pode-se definir o dicionário de cores a serem captadas pelo algoritmo. Em face que este trabalho visa a identificação das cores primárias, vermelho amarelo e azul, e as cores secundárias, roxo, laranja e verde.

Figura 11. Plano cartesiano RGB para HSV



Fonte: OpenCV (2017)

Para a realização do levantamento do dicionário proposto, foi utilizado o plano cartesiano de correspondência de cores RGB e HSV contido na figura 11, onde o Eixo X representa os valores de “Hue”, a matização da cor, sendo que estes valores estão contidos em (0 < H < 180), e o eixo Y corresponde a saturação da imagem contida na variação entre (0 < S < 100). A variável V corresponde a luminosidade do ambiente, e foi definida na variação entre 0 à 255, sendo 0 a escuridão absoluta e 255 o brilho máximo de uma imagem.

Com a utilização do plano cartesiano e a realização de testes para a validação das cores selecionadas, foi definido o seguinte dicionário de cores:

Figura 12. Dicionário de cores

```
# definir os limites inferior e superior
# das cores no espaço de cores HSV
lower = {'vermelho': (0, 50, 20),
         'azul': (97, 100, 117),
         'amarelo': (23, 70, 120),
         'verde': (40, 70, 80),
         'laranja': (10, 100, 180),
         'roxo': (140, 140, 20)
         }

upper = {'vermelho': (5, 255, 255),
         'azul': (117, 255, 255),
         'amarelo': (30, 255, 255),
         'verde': (75, 255, 255),
         'laranja': (20, 255, 255),
         'roxo': (155, 255, 255)
         }
```

Fonte: Autor (2020)

A proposta do uso do HSV é a possibilidade de se definir um intervalo entre as cores e nomear qualquer cor contida naquela variação uma determinada cor, neste caso, o valor mínimo de cada cor está contido na variável ‘lower’ e seu valor máximo na variável ‘upper’. Assim, qualquer valor contido entre estas variáveis é catalogado como uma cor, por exemplo, o amarelo, que tem o seu valor mínimo de (23 H,70 S, 120 V), e o valor máximo de (30 H, 255 S, 255 V), qualquer valor dentre o intervalo destas matrizes é identificado como amarelo.

Com as definições claras do que deve ser identificado pelo algoritmo, pode-se efetuar o tratamento da imagem através do conceito de morfologia e aplicação de máscaras, conforme explicitado na Figura 13.

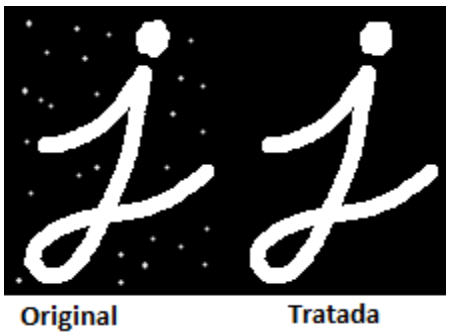
Figura 13. Tratamento de ruídos

```
kernel = np.ones((9, 9), np.uint8)
mask = cv2.inRange(hsv, lower[key], upper[key])
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
```

Fonte: Autor (2020)

As técnicas de morfologia utilizadas são as de erosão e dilatação. O OpenCv possui funções onde ele realiza a tratativa da imagem com a combinação dessas duas técnicas. A função cv2.MORPH_OPEN realiza uma erosão seguida de uma dilatação, removendo ruídos na imagem.

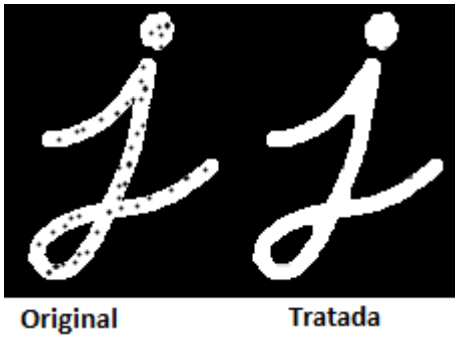
Figura 13. Técnica de abertura da imagem



Fonte: OpenCV (2020)

Com os ruídos fora da máscara removidos, foi tratado o fechamento da máscara e os ruídos dentro do objeto identificado. Dada a utilização da função cv2.MORPH_CLOSE, que é o inverso da função de abertura, ela realiza primeiro a dilatação e, em seguida, a erosão, sendo bastante útil para remover pequenos pontos dentro de objetos.

Figura 14. Técnica de fechamento da imagem



Fonte: OpenCV (2020)

De posse de uma máscara já tratada, parte-se para a delimitação do contorno do objeto detectado e para a criação da máscara de detecção a ser aplicada naquele objeto. A figura 15 mostra a função utilizada para realizar a detecção de borda do objeto com a cor identificada no dicionário.

Figura 15. Detecção de borda

```
# encontra contornos na máscara e inicializa o atual
# (x, y) centro da esfera
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None
```

Fonte: Autor (2020)

Quando uma cor correspondente ao dicionário é encontrada na imagem, ela gera uma máscara que detecta este objeto, então, essa função é acionada para que se possa determinar os contornos dele. Ela pega o último pixel de cada ponto da máscara criada e esses pontos são registrados na variável ‘cnts’. Quando esta variável tiver um valor diferente de 0, é acionada a função para a criação do círculo que delimita o objeto a partir do centro da máscara.

Figura 16. Criação da circunferência ao redor do objeto

```
# só prossegue se pelo menos um contorno for encontrado
if len(cnts) > 0:
    # encontre o maior contorno da máscara e, em seguida,
    # para calcular o círculo mínimo e centroide
    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["
```

Fonte: Autor (2020)

O contorno do objeto é gerado, mas para que este contorno seja perceptível na imagem, é atribuído a ele uma cor sólida para cada chave encontrada. As chaves são determinadas pelo dicionário de cores, com a junção do seu valor mínimo e máximo.

Figura 17. Cores de circunferência para cada chave

```
# definir cores padrão para o círculo ao redor do objeto
cores = {'vermelho': (0, 0, 255),
         'azul': (255, 0, 0),
         'amarelo': (0, 255, 217),
         'verde': (50, 255, 40),
         'laranja': (0, 165, 255),
         'roxo': (153, 51, 153)
        }
```

Fonte: Autor (2020)

Figura 18. Desenho da circunferência na tela

```
# só prossegue se o raio atingir um tamanho mínimo.
if radius > 0.5:
    # desenha o círculo e o centroide no quadro,
    # então atualiza a lista de pontos rastreados
    cv2.circle(frame, (int(x), int(y)),
               int(radius), cores[key], 2)
    cv2.putText(frame, key, (int(x-radius), int(y - radius)),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, cores[key], 2)
```

Fonte: Autor (2020)

Após todos os processos implementados, parte-se para a plotagem no display do que está sendo captado e processado.

Figura 19. Desenho da circunferência na tela

```
# mostre o quadro em nossa tela
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
```

Fonte: Autor (2020)

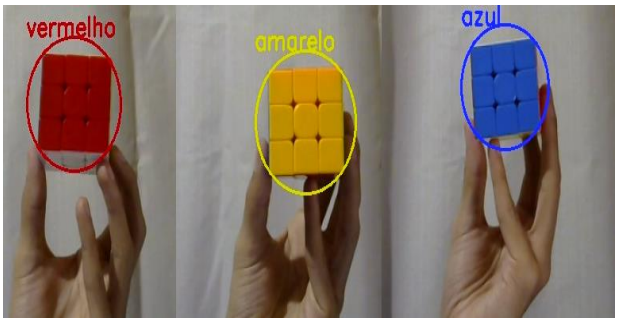
A partir deste ponto, foi possível dar início aos testes e análises finais do trabalho, para assim identificar as limitações impostas por situações adversas.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Após o desenvolvimento apresentado na seção anterior, foi possível executar o fluxo proposto. Para isso, basta executar o programa.

O primeiro teste com o programa em execução foi o reconhecimento individual das cores primárias catalogadas no dicionário ***n*** objeto, neste caso, foi utilizado um cubo mágico com várias faces coloridas, para reconhecer as cores amarelo, azul e vermelho, conforme explicitado na figura 20.

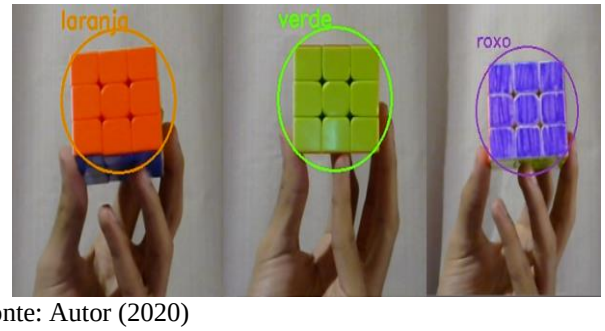
Figura 20. Teste de reconhecimento de cores primárias



Fonte: Autor (2020)

O programa se mantém estável e identifica as cores no display, conforme o esperado. O mesmo teste foi realizado com as cores secundárias, laranja, verde e roxo, conforme mostrado na figura 21. Dando, assim, fim ao teste 1.

Figura 21. Teste de reconhecimento de cores secundárias

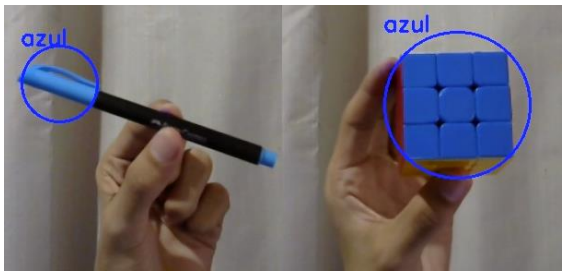


Fonte: Autor (2020)

As cores secundárias também são facilmente captadas, dando ênfase na cor roxa, onde o objeto possui pontos brancos e diferenças de tonalidade, pois foi pintado com tinta roxa, mas, após o tratamento de erosão e dilatação, o algoritmo identificou perfeitamente a cor.

Dando sequência, parte-se o teste 2, onde foi testado o "range" definido de cada cor; se estava em perfeitas condições, por exemplo. O dicionário detém a cor azul, mas a cor azul possui tonalidades que estão entre claro e escuro, ficando evidente na Figura 22.

Figura 22. Tons diferentes de azul



Fonte: Autor (2020)

No teste 2, foi utilizada uma caneta com sua tampa na coloração azul claro, e o cubo com a coloração azul escuro. Com este teste, pode-seaferir que a tonalidade de cada cor será identificada, seja ela clara ou escura.

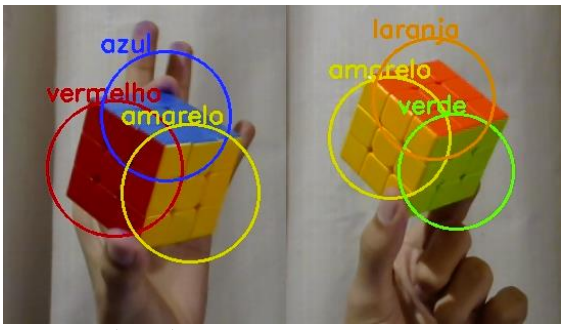
O algoritmo não possui limitação da quantidade de cores que podem ser listadas simultaneamente, mas ele lista apenas uma vez cada cor, isso é definido pelo fator raio do objeto identificado e a máscara é aplicada no maior objeto captado. Na Figura 23, tem-se a demonstração do teste 3, onde mostra que foram posicionados 6 potes de tintas em paralelo, cada pote contendo uma tinta com a cor correspondente às do dicionário. O reconhecimento das 6 cores acontece em simultâneo.

Figura 23. Reconhecimento simultâneo

Fonte: Autor (2020)

Neste teste, foi perceptível a variação da luminosidade do objeto e seu impacto na identificação precisa da cor. O pote de tinta verde perde o reconhecimento quanto maior for a sombra na imagem, se aproximando da cor preta, a qual não está catalogada para reconhecimento. Ainda sobre o teste de várias cores simultaneamente, foi posicionado um único objeto, mas com 3 faces expostas à câmera, cada uma correspondendo a uma cor, e foi possível identificar que o algoritmo detecta sem maiores dificuldades cores diferentes no mesmo objeto, conforme apresentado na Figura 24.

Figura 24. Reconhecimento simultâneo em um único objeto



Fonte: Autor (2020)

Um dos fatores que podem atrapalhar o reconhecimento de cores e detecção de objetos é a poluição visual. Por isso, no teste 4, o algoritmo foi submetido a identificação das cores propostas em uma situação adversa, onde se possui objetos distintos com cores catalogadas e não catalogadas, e objetos diferentes com a mesma cor, conforme

apresentado na Figura 25.

Figura 25. Situação adversa

Fonte: Autor (2020)

No teste 4, tem-se 3 objetos azuis, e o algoritmo reconhece apenas o maior. De acordo com o programado, as cores não catalogadas não são detectadas ou alocadas a outras tonalidades, e as 6 cores presentes no dicionário são corretamente identificadas em tempo de execução, a identificação ocorre em tempo de captura, sendo a taixa de latencia de 1ms, diante deste fator podemos aferir que a aplicação responde em tempo real.

Diante dos 4 testes executados, foi possível identificar que a iluminação do ambiente é o principal fator de interferência no reconhecimento das cores: se a intensidade da luz é muito alta, os objetos refletem a tonalidade branca, tornando-se um ruído na imagem, o que leva a não identificação da cor do objeto. Outro fator adverso é a ausência parcial ou total de luz, quanto mais escuro o ambiente é, mais difícil para que algoritmo encontre as cores do objeto, o que é um total reflexo da visão humana, aferindo-se que o algoritmo pode sim ajudar na identificação de cores em condições ideias para qualquer outro ser humano.

Uma ressalva importante a ser feita é a qualidade da câmera utilizada para a captura da imagem. A câmera utilizada neste projeto foi uma câmera básica, com o limite de 720 pixels e resolução em HD. Caso seja utilizada uma câmera com maior capacidade e qualidade, os impactos de luminosidade e reflexo podem ser menores, trazendo, dessa maneira, uma melhoria no resultado.

5 CONSIDERAÇÕES FINAIS

Através da tese exposta anteriormente, pode-se concluir que a visão computacional juntamente com a programação certa, se demonstra uma grande aliada aos portadores de qualquer nível de daltonismo, pois a implementação deste algoritmo pode ser feita através de um aplicativo em smartphones ou qualquer dispositivo que possua uma câmera acoplada e capacidade de processamento, respondendo em tempo real. Desse modo, atividades do cotidiano identificadas como simples para pessoas que não possuem esta limitação, tornam-se acessíveis para os daltônicos.

Entretanto, o projeto possui limitações. Apesar da facilidade no desenvolvimento do algoritmo, ele foi desenvolvido baseado em tecnologias de código aberto no mercado para um desktop, e não foi efetuado uma integração com um aplicativo, apesar de ser possível através de ferramentas específicas. Outra limitação é quanto a capacidade de processamento do dispositivo utilizado, porém, o código fonte do projeto é totalmente mutável e escalável, sendo possível colocar um catálogo maior de cores a serem processadas, mas isto está diretamente ligado a capacidade de processamento do dispositivo, podendo, assim, causar um mau funcionamento do algoritmo. O que pode ser concluído é que a visão computacional aliada aos algoritmos de processamento digital de imagem é altamente viável, não se limitando à tecnologia, e sim a hardwares capazes de suportar as necessidades do sistema proposto. Todavia, os benefícios proporcionados por estas tecnologias são imensuráveis, pois torna possível uma pessoa que não consegue identificar cores ser capaz de tal, através de uma interface máquina.

AGRADECIMENTOS

Agradeço aos meus mestres pelos ensinamentos e as lições passadas, as quais levarei para a vida toda, aos bravos colegas que iniciaram esta jornada ao meu lado e persistiram até o fim, e um agradecimento em especial à minha noiva, Karinne, pelo apoio e suporte neste momento difícil, que me manteve firme e forte para seguir adiante.

REFERÊNCIAS

MELO, Débora, GALON, José, FONTANELLA, Bruno. **Os "daltônicos" e suas dificuldades: condição negligenciada no Brasil?** Physis, 2014.

QUEIROZ, José, GOMES, Herman. **Introdução ao Processamento Digital de Imagens**. Revista RITA, 2001.

SILVA, Tarcízio et al. APIs de **Visão Computacional: investigando mediações algorítmicas a partir de estudo de bancos de imagens**. Logos 52, v. 27, n. 01, Rio de Janeiro, 2020.

SILVA, T. et al. **Apis De Visão Computacional: Investigando Mediações Algorítmicas a Partir De Estudo De Bancos De Imagens**. [s. l.], 2020. DOI 10.12957/logos.2020.51523. Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.5A7CCEF&lang=pt-br&site=eds-live>. Acesso em: 16 nov. 2020.

R. C. Gonzalez and R. E. Woods, **Digital Image Processing**. Prentice-Hall, Upper
Saddle River, NJ, second ed., 2002.

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

BARELLI,. **Introdução à Visão Computacional: Uma abordagem prática com**
Python e OpenCV. [S.l.]: Casa do Código, 2018.

GONZALES, R. C.; WOODS, R. E. **Processamento Digital de Imagens, 3a edição**. São Paulo: **Person Education** do Brasil LTDA, 2009.

GARCÍA, G. B. et al. **Learning Image Processing with OpenCV**. [S.l.]: Packt Publishing Ltd, 2015.

SZELISKI. R. **Computer Vision: Algorithms and Applications**. Springer, 2010.

VETRISELVAN, D.; STALIN, T. **Computer Vision with OpenCV and Python 3: Practical examples workbook.** 2017.

E. R. DAVIES. **Computer Vision: Principles, Algorithms, Applications, Learning**. Academic Press; 5ª edição 2017.

BRADSKI, G.; KAEHLER, **A. Learning OpenCV**. Sebastopol: [s.n.], 2008.

SONKA, M. et al. **Image Processing, Analysis, and Machine Vision**. engage Learning; 4ª edição 2014.

PEDRINE, H. ; SCHWARTZ, W. **Análise De Imagens Digitais: Princípios, Algoritmos E Aplicações.** engage Learning; 1ª edição, 2008.

FERNANDES, Leandro. **Visão Computacional.** 2012. Disponível em: <ic.uff.br/~julius/icc/vcomp.pdf>. Acesso em: 16 nov 2020.

MATHWORKS (2012) **Matlab R2012 User's guide**.

OpenCV. **The best computer vision library in the world got even better.** Disponível em: < https://opencv.org/> Acesso em: 10 agst 2020.

Python. **Python 3.8.6 documentation**. Disponível em: < https://docs.python.org/3.8/> Acesso em: 15 agst 2020.

Marengoni, M. ; Stringhini, D. **Tutorial: Introdução à Visão Computacional usando OpenCV.** Disponível em: < https://www.researchgate.net/publication/41799461_Tutorial_Introducao_a_Visao_Computacional_usando_OpenCV> Acesso em: 19 set 2020.

Ribeiro, V**. Espectro de Fourier - uma possibilidade para a Infoestética.** Disponível em: < https://www.ime.usp.br/~map/tcc/2014/Valdivan_BMAC_2014.pdf > Acesso em: 19 set 2020