**UNIVERSIDADE FEDERAL DO ABC**

**CENTRO DE ENGENHARIA, MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS**

**Gustavo Dantas Hidalgo**

# APLICATIVO PARA *SMARTPHONE*: IDENTIFICADOR DE CORES PARA PESSOAS COM DEFICIÊNCIA VISUAL

**SANTO ANDRÉ - SP**

**2019**

**Gustavo Dantas Hidalgo**

# APLICATIVO PARA *SMARTPHONE*: IDENTIFICADOR DE CORES PARA PESSOAS COM DEFICIÊNCIA VISUAL

Trabalho de Graduação apresentado à Universidade Federal do ABC como requisito parcial à conclusão do curso de graduação em Engenharia de Informação.

**Orientadora**: **Prof.ª Dr.ª Denise Consonni**

**SANTO ANDRÉ - SP**

**2019**

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Dantas Hidalgo, Gustavo
APLICATIVO PARA SMARTPHONE : IDENTIFICADOR DE CORES PARA PESSOAS COM DEFICIÊNCIA VISUAL / Gustavo Dantas Hidalgo. — 2019.

42 fls. : il.

Orientadora: Denise Consonni

Trabalho de Conclusão de Curso — Universidade Federal do ABC, Bacharelado em Engenharia de Informação, Santo André, 2019.

1. Tecnologia assistiva. 2. Deficiência visual. 3. Smartphone. 4. Identificação de cores. 5. Android. I. Consonni, Denise. II. Bacharelado em Engenharia de Informação, 2019. III. Título.

Para Marcela Lopes, por me mostrar que eu não tenho nem devo querer ter controle sobre tudo, e me lembrar que o destino nada mais é do que o peso das circunstâncias.

## AGRADECIMENTOS

À minha família, pelo amor demonstrado e pelo apoio.

À Prof.ª Dr.ª Denise Consonni, por ter iluminado meu caminho dentro da UFABC e permitiu que eu chegasse a esse momento.

A Otavio Silva e Lucas Paredes, pela grandiosa ajuda e exemplo.

À Guilherme Ravagnani, pelo voto de confiança e apoio que ajudaram a iniciar a minha carreira como desenvolvedor.

A todos os meus amigos pela atenção, compreensão, suporte e paciência.

A todos que de alguma forma contribuíram para a chegada desse momento.

À UFABC.

*“... porque a cor não é mais do que a luz do Sol aprisionada…”*

*(Umberto Eco)*

## RESUMO

Os avanços tecnológicos nas áreas de Engenharia e Tecnologia da Informação e Comunicação fizeram com que os telefones celulares de última geração (os *smartphones*) adquirissem *hardwares* cada vez mais poderosos e executassem funções cada vez mais inovadoras e funcionais. Com esses avanços, torna-se mais fácil o uso dessas tecnologias por pessoas com deficiência, que por sua vez podem fazer desses aparelhos, ferramentas essenciais para exercer sua cidadania, o que se denomina de Tecnologia Assistiva. A maioria da população brasileira com alguma deficiência é formada por pessoas com deficiência visual que, por curiosidade ou necessidade, podem precisar saber a cor de determinado objeto. O objetivo central desse projeto é utilizar a tecnologia disponível atualmente para tornar possível e prático que uma pessoa com deficiência visual, munido de um *smartphone*, obtenha informação sobre a cor de um determinado objeto, de forma fácil e confiável, e com uma experiência melhor que a disponível até o momento para usuários de língua portuguesa. Para o desenvolvimento do aplicativo, foi empregada uma poderosa biblioteca de processamento de imagens, aliada a um algoritmo de classificação e contagem de cores simples que, além de se mostrar funcional, permite o uso da informação de uma maneira diferente e mais eficaz que os outros aplicativos disponíveis.



**Palavras-chave:** tecnologia assistiva, deficiência visual, *smartphone*, identificação de cores, *Android*.

## ABSTRACT

The technological advances from the Information and Communication Technology and Engineering fields have made the latest generation of mobile phones (smartphones) acquire ever more powerful hardware and perform increasingly innovative and functional features. With these advances, it becomes easier the usage of these technologies by people with disabilities, who in turn can make these devices essential tools to exercise their citizenship, which is known as Assistive Technology. The majority of the Brazilian population with a disability is of visually impaired people, who, out of curiosity or need, may need to know the color of a particular object. The central objective of this project is to use the currently available technology to make it possible, practical and reliable for a visually impaired person with a smartphone, to obtain color information of a specific object, with a better experience than those available so far for Brazilian Portuguese-speaking users. For the application development, a powerful image processing library was employed, coupled with a simple algorithm of color classification and counting which, besides being functional, allows the use of this information in a different and more effective way than the other applications available.



**Keywords:** assistive technology, visual impairment, smartphone, color identification, Android.

## LISTA DE FIGURAS


Figura 1: Esquema do caminho percorrido pela luz até o sensor de imagem ..... 32
Figura 2: Fluxograma da geração da imagem digital a partir do sensor ..... 34
Figura 3: Representação do modelo RGB mapeado num cubo ..... 35
Figura 4: Representação do modelo HSV mapeado num cilindro ..... 37
Figura 5: Variação do matiz em função de *H* ..... 37
Figura 6 Translação e projeção do cubo RGB no plano xy ..... 39
Figura 7: Transformação do hexágono projetado em círculo ..... 40
Figura 8: Representação da metodologia ..... 43
Figura 9: Cilindro HSV com detalhe do plano SV e o mesmo representado em detalhes com seus limiares ..... 46
Figura 10: Cilindro HSV com detalhe do plano HS e o mesmo representado em detalhes com seus limiares ..... 48
Figura 11: Fluxograma de funcionamento do aplicativo ..... 51
Figura 12: Algoritmo identificador e mapeador de fonte de luz ..... 52
Figura 13: Algoritmo classificador e contador de cor ..... 54
Figura 14: Algoritmo classificador de tonalidade ..... 55
Figura 15: Algoritmo classificador de matiz e construtor de cor ..... 56
Figura 16: Tela de captura de imagem nos modos normal, nível de cinza e binarizada ..... 60
Figura 17: Tela de captura de imagem para usuário final ..... 61
Figura 18: Tela de gráfico de cores ..... 61
Figura 19: Tela de resultado ..... 63
Figura 20: Captura da imagem com *S* fixo ..... 65
Figura 21: Captura da imagem com *H* fixo ..... 66
Figura 22: Captura da fachada da sede da SEESP, iluminação natural ..... 68
Figura 23: Captura do mural da Praça Paulo Kobayashi, iluminação natural ..... 69
Figura 24: Captura de miniaturas, iluminação natural ..... 70
Figura 25: Captura de miniaturas, iluminação artificial ..... 71
Figura 26: Captura de miniaturas, iluminação artificial e *flash* acionado ..... 72

Figura 27: Captura de miniaturas, iluminação artificial e reflexo de luz ..... 73
Figura 28: Captura de miniaturas, iluminação artificial e fonte de luz ..... 74

## LISTA DE QUADROS

Quadro 1: Características dos aplicativos testados ..... 57
Quadro 2: Relação de cor e categoria do gráfico ..... 62
Quadro 3: Especificações *Xiaomi Mi A2* ..... 67
Quadro 4: Cronograma de atividades ..... 79

## LISTA DE TABELAS

Tabela 1: Relação de identificação, valor RGB de referência, *H* e limiares ..... 49
Tabela 2: Relação das faixas de proporção de píxeis invalidados e seus respectivos tempos de vibração e pausa no modo de captura ..... 50

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | *Application Programming Interface* |
| IDE | *Integrated Development Environment* |
| HSV | *Hue, Saturation, Value* |
| LED | *Light Emissor Diode* |
| OpenCV | *Open Source Computer Vision Library* |
| RGB | *Red, Green, Blue* |
| SDK | *Software Development Kit* |
| SO | Sistema Operacional |
| UFABC | Universidade Federal do ABC |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| *R* | Valor na escala de intensidade de cor vermelha da lista ordenada RGB |
| *G* | Valor na escala de intensidade de cor verde da lista ordenada RGB |
| *B* | Valor na escala de intensidade de cor azul da lista ordenada RGB |
| *V* | Valor na escala de brilho relativo da lista ordenada HSV |
| *S* | Valor na escala de saturação relativa da lista ordenada HSV |
| *H* | Valor na escala de matiz da lista ordenada HSV |
| *Y'* | Valor de Luma |
| *INV* | Cor inválida |
| *LS* | Limiar de saturação relativa baixa (do inglês *Low Saturation*) |
| *HS* | Limiar de saturação relativa alta (do inglês *High Saturation*) |
| *LV* | Limiar de brilho relativo baixo (do inglês *Low Value*) |
| *MV* | Limiar de brilho relativo médio (do inglês *Medium Value*) |
| *HV* | Limiar de brilho relativo alto (do inglês *High Value*) |
| *T* | Tonalidade classificada |
| *BLA* | Tonalidade de cor preta (do inglês *Black*) |
| *DGR* | Tonalidade de cor cinza escuro (do inglês *Dark Gray*) |
| *DC* | Tonalidade de cor escura (do inglês *Dark Color*) |
| *LGR* | Tonalidade de cor cinza claro (do inglês *Light Gray*) |
| *WHI* | Tonalidade de cor branca (do inglês *White*) |
| *LC* | Tonalidade de cor clara (do inglês *Light Color*) |
| *NC* | Tonalidade de cor normal (do inglês *Normal Color*) |
| *HUT* | Limiar superior de matiz (do inglês *Hue Upper Threshold*) |
| *HLT* | Limiar inferior de matiz (do inglês *Hue Lower Threshold*) |
| *RED* | Matiz vermelho (do inglês *Red*) |
| *ORA* | Matiz laranja (do inglês *Orange*) |
| *YEL* | Matiz amarelo (do inglês *Yellow*) |
| *GYE* | Matiz verde amarelado (do inglês *Green Yellow*) |
| *GRE* | Matiz verde (do inglês *Green*) |
| *GCY* | Matiz verde ciano (do inglês *Green Cyan*) |
| *CYA* | Matiz ciano (do inglês *Cyan*) |

| | |
|---|---|
| *BCY* | Matiz azul ciano (do inglês *Blue Cyan*) |
| *BLU* | Matiz azul (do inglês *Blue*) |
| *VIO* | Matiz violeta (do inglês *Violet*) |
| *MAG* | Matiz magenta (do inglês *Magenta*) |
| *PIN* | Matiz rosa (do inglês *Pink*) |
| *CQ* | Matriz da captura de quadro feito pela câmera do smartphone |
| *PL* | Proporção de luz de uma captura de quadro |
| *RDI* | Matriz da região de interesse |
| *VCL* | Vetor de contornos de luz |
| *MCI* | Mapeamento de cores da imagem |
| *IC* | Índice de cor classificado |

**SUMÁRIO**


**1 INTRODUÇÃO** **25**
1.1 O QUE JÁ EXISTE 25
1.2 FERRAMENTAS PARA DESENVOLVIMENTO 27
1.3 JUSTIFICATIVA 28
1.4 HIPÓTESE 28
1.5 OBJETIVOS 29
1.5.1 Objetivos gerais: 29
1.5.2 Objetivo específico 29
**2 FUNDAMENTAÇÃO TEÓRICA** **31**
2.1 VISÃO COMPUTACIONAL 31
2.2 IMAGEM DIGITAL 31
2.2.1 Aquisição de uma imagem digital 31
2.3 MODELO DE COR 35
2.3.1 RGB 35
2.3.2 HSV 36
2.3.3 Nível de cinza (Luma) 38
2.3.4 Imagem binária 38
2.4 PROCESSAMENTO DIGITAL DE IMAGENS 38
2.4.1 Conversão RGB para HSV 39
*2.4.1.1 Conversão do cubo RGB para o cilindro HSV* 39
*2.4.1.2 Conversão das listas ordenadas* 40
2.4.2 Conversão RGB para nível de cinza 41
2.4.3 Conversão nível de cinza para binário 41
2.4.4 Transformações morfológicas 41
*2.4.4.1 Dilatação:* 42
*2.4.4.2 Erosão:* 42
*2.4.4.3 Abertura:* 42
**3 METODOLOGIA** **43**
3.1 OPENCV 45
3.2 DEFINIÇÃO DE COR NO APLICATIVO 45
3.2.1 Divisão do plano SV 46
3.2.2 Divisão do plano HS 47
3.3 FLUXO GERAL 50
3.4 ALGORITMO IDENTIFICADOR DE FONTE DE LUZ 51
3.5 ALGORITMO CLASSIFICADOR E CONTADOR DE COR 52
3.5.1 Classificação da tonalidade 54
3.5.2 Classificação do matiz 55

**4 RESULTADOS** **57**
4.1 CARACTERÍSTICAS DOS OUTROS APLICATIVOS 57
4.2 APLICATIVO DESENVOLVIDO 59
4.2.1 Captura de imagem 59
*4.2.1.1 Captura de imagem para usuário final* 60
4.2.2 Gráfico de cores 61
4.2.3 Descrição do resultado 63
4.3 RESULTADOS OBTIDOS E DISCUSSÃO 63
4.3.1 Android Emulator 64
4.3.2 Xiaomi Mi A2 67
4.3.3 Discussão 75
**5 CONSIDERAÇÕES FINAIS** **77**
5.1 CONCLUSÕES 77
5.2 TRABALHOS E MELHORIAS FUTURAS 78
**6 CRONOGRAMA DE ATIVIDADES** **79**
**REFERÊNCIAS** **81**

# 1 INTRODUÇÃO

Segundo a Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência (BRASIL, 2016):

> Acessibilidade é um atributo essencial do ambiente que garante a melhoria da qualidade de vida das pessoas. Deve estar presente nos espaços, no meio físico, no transporte, na informação e comunicação, inclusive nos sistemas e tecnologias da informação e comunicação, bem como em outros serviços e instalações abertos ao público ou de uso público, tanto na cidade como no campo.

Dá-se o nome de Tecnologia Assistiva qualquer produto, serviço ou sistema que pode ser utilizado para aumentar, manter ou melhorar as capacidades funcionais de pessoas com deficiência. Ao fazê-lo, ela fornece acessibilidade, ou seja, facilita o desenvolvimento das atividades diárias, provê independência, empoderamento e melhorando a qualidade de vida dessas pessoas (BERSCH, 2013).

Dados do Censo Demográfico de 2010, feito pelo Instituto Brasileiro de Geografia e Estatística (IBGE, 2010), mostram que no Brasil há 35,7 milhões de pessoas (18,76% da população total) com deficiência visual, e que elas representam 78,4% da população total com qualquer tipo de deficiência (45,6 milhões de pessoas). Isso evidencia uma demanda em potencial para o uso de Tecnologia Assistiva no país, voltada para esse grupo de pessoas.

## 1.1 O QUE JÁ EXISTE

Hoje em dia existe uma série de soluções de Tecnologia Assistiva para pessoas com deficiência visual. As diversas associações de pessoas com deficiência possuem listagens das principais soluções em Tecnologia Assistiva e o Ministério da Ciência, Tecnologia e Inovação (BRASIL, 2008) possui disponibilizado num *site* à parte, o Catálogo Nacional de Produtos de Tecnologia Assistiva: uma ferramenta web que possibilita a realização de buscas sobre os produtos de Tecnologia Assistiva fabricados ou distribuídos no Brasil.

Dentre os vários produtos destinados a pessoas com deficiência visual presentes neste catálogo, destaca-se o *Auire Prisma* (AUIRE TECNOLOGIAS ACESSÍVEIS, [s.d.]): este é um dispositivo físico identificador de cores e dinheiro. O

aparelho possui 3 sensores que identificam a intensidade de cada cor primária, um diodo emissor de luz (LED, do inglês *light emissor diode*) para iluminar o objeto cuja cor se deseja identificar, um botão para acionar a função de identificação da cor do objeto e outro botão para acionar a identificação do valor de uma nota de Real. Um tempo após apertar o primeiro botão com o objeto exposto aos sensores, o dispositivo sintetiza a fala correspondente à cor do objeto em questão. Se o mesmo for feito usando o segundo botão numa cédula de Real, ele emitirá o valor da nota.

Há também aplicativos que fazem do *smartphone* um instrumento poderoso de tecnologia assistiva. Nesse grupo destaca-se o *ViaOpta Daily* (NOVARTIS PHARMA AG., [s.d.]). O *Novartis ViaOpta Daily* é uma suíte de aplicações que inclui lupa, cronômetro, identificador de cores, identificador de objetos, tudo isso construído dentro de uma interface gráfica acessível (com poucos detalhes gráficos e ícones grandes) e com sistema próprio de descrição auditiva dos menus. Esse aplicativo está disponível tanto para o sistema operacional (SO) *Google Android* quanto para o SO *Apple iOS*.

A *Microsoft* também tem um aplicativo para *smartphones* com SO *iOS* chamado *Seeing AI*. O *Microsoft Seeing AI* (MICROSOFT, [s.d.]) é uma suíte de aplicações que inclui um leitor de textos e documentos, identificador de produtos, identificador de rostos, descritor de ambientes, identificador de cédulas de dinheiro, indicador de luminosidade e detector de cores. Parte dessas tarefas, como descrever ambientes e ler textos, são feitas fazendo uso de inteligência artificial. Contudo, esse aplicativo faz a audiodescrição apenas na língua inglesa, e também, por possuir muitas funcionalidades, transitar entre elas torna-se uma tarefa complicada.

Há diversos outros aplicativos que também fazem identificação de cores de objetos através da imagem vinda da câmera do *smartphone*. Alguns desses aplicativos que foram testados para esse trabalho são: *Color Detector* da *Mobialia* (MOBIALIA, [s.d.]), *Color Detector* da *sadens Studio* (GOOGLE, [s.d.]) e *Detector de Cor* da *RamelTec* (GOOGLE, [s.d.]).

Na questão das interfaces de auxílio do SO, existe uma preferência das pessoas com deficiência visual pelos smartphones com SO *iOS*. Essa predileção é discutida por Morris e Mueller (2014, p. 69-79, tradução livre pelo autor deste

trabalho) que dizem que, em relação ao SO *Android*, “o ecossistema comparativamente melhor definido da *Apple* faz com que a obtenção de suporte de pares (e o "aprendizado" de um novo dispositivo) seja uma tarefa mais fácil”. Essa preferência pode ser observada entre pessoas com deficiência visual residentes nos Estados Unidos (WIRELESS RERC, 2016).

## 1.2 FERRAMENTAS PARA DESENVOLVIMENTO

A arquitetura da plataforma do SO *Android* é formada por diversas camadas, sendo que cada camada é formada por um conjunto de código ou de aplicações com alguma finalidade específica. Uma dessas camadas é a estrutura da interface de programação de aplicação (API, do inglês *Application Programming Interface*) *Java*, que provê o conjunto completo dos recursos do SO *Android* através de APIs programadas na linguagem *Java* (GOOGLE, [s.d.]). Dentro dessa estrutura, há uma API que provê acesso à câmera do *smartphone*, através de métodos para realizar capturas de imagens e extrair informações da imagem.

Noutra camada do SO *Android*, formada pelos aplicativos do sistema, existe um aplicativo que contém algumas funções de suporte a pessoas com deficiência visual. O leitor de tela *Google TalkBack* (GOOGLE, [s.d.]) provê descrição falada, utilizando um sistema de síntese de fala (TTS, do inglês *Text-To-Speech*) que vem embarcado no SO, para descrever e ler o que está aparecendo na tela de maneira a possibilitar o uso do *smartphone* sem a necessidade de olhar para a tela.

Fora do ambiente do SO, existe a biblioteca OpenCV (*Open Source Computer Vision Library*, em português, biblioteca de visão computacional de código aberto; BRADSKI, 2000), que provê as mesmas funcionalidades da API do SO e possui uma infinidade de ferramentas de processamento digital de imagem e vídeo que vão desde aplicação de filtros a detecção de objetos.

Fazendo o uso dessas ferramentas, foi então desenvolvido um aplicativo para identificar as cores para atender pessoas com deficiência visual. Esse aplicativo apresenta melhorias com relação ao *ViaOpta Daily*, justamente devido ao uso do *TalkBack*. O sistema de descrição auditiva do *ViaOpta Daily* possui poucas línguas, e o português brasileiro está com um sotaque de português europeu. O aplicativo

proposto é também mais prático e evoluído que o *Auire Prisma*, pois está presente no *smartphone*, sem a necessidade de um *hardware* dedicado, e oferecendo a possibilidade de se aumentar o número de funções no aplicativo.

## 1.3 JUSTIFICATIVA

Pessoas cegas convivem com pessoas que enxergam. Logo, a cor é uma informação à qual elas não têm acesso mas influencia como as pessoas que enxergam os veem. É uma questão ainda mais complicada para cegos de nascença pois, como para elas nunca houve contato com a cor, não existe uma referência sobre o seu significado. Isso tem um forte impacto, por exemplo, na questão do vestuário.

A cor também impacta quando a mesma é usada como símbolo ou representação gráfica de um estado. Por exemplo: em mapas, normalmente utilizam-se cores diferentes para referenciar regiões diferentes; em planilhas, as cores podem ser utilizadas para filtragem de dados.

Os telefones móveis celulares de última geração, conhecidos como *smartphones*, passam continuamente por evoluções que agregam a esses aparelhos recursos de *hardware* cada vez mais poderosos, com a adição de novos dispositivos sensores, ambientes de desenvolvimento e interfaces cada vez mais simples e intuitivas. Com a popularização dos *smartphones*, uma quantidade maior de pessoas tem a possibilidade de usufruir desses benefícios, inclusive pessoas com alguma deficiência. Dado isso, há uma grande área a ser explorada, que trata de desenvolver soluções usando esses dispositivos para auxiliar, empoderar e tornar mais independente a vida das pessoas com deficiência (BRILL, 2016), nesse caso específico, pessoas com deficiência visual.

## 1.4 HIPÓTESE

O uso deste aplicativo através de um *smartphone* pode promover, com alto nível de acessibilidade, maior independência a pessoas com deficiência visual.

## 1.5 OBJETIVOS

### 1.5.1 Objetivos gerais:

a) estudar a linguagem de programação *Java*;
b) estudar o desenvolvimento de aplicativos para o SO *Android*;
c) desenvolver uma solução tecnológica para auxiliar pessoas com deficiência visual (Tecnologia Assistiva).

### 1.5.2 Objetivo específico

Desenvolver um aplicativo para identificar as cores, alguns padrões de cores de objetos e posteriormente, para identificar cédulas de Real colocados frente à câmera de um telefone celular *smartphone* com SO *Android*.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 VISÃO COMPUTACIONAL

De acordo com Huang (1996, tradução do autor desse trabalho):

> A visão computacional tem um duplo objetivo. Do ponto de vista da ciência biológica, a visão computacional tem como objetivo propor modelos computacionais do sistema visual humano. Do ponto de vista da engenharia, a visão computacional visa construir sistemas autônomos que possam executar algumas das tarefas que o sistema visual humano pode executar (e até superá-lo em muitos casos).

Muitas técnicas de visão computacional são empregadas emulando o sistema de visão humano, visando obtenção de alguma informação. Neste projeto, foi elaborado um sistema que utiliza processamento digital de imagem para extrair informação de cor.

## 2.2 IMAGEM DIGITAL

Uma imagem digital de rastreio (*raster* ou mapa de bits) pode ser definida como sendo uma matriz de duas dimensões, na qual cada elemento dessa matriz representa um píxel, que por sua vez guarda consigo o valor de sua cor (SHIRLEY; MARSCHNER, 2009).

O píxel é o menor elemento de uma imagem digital. Ele representa o ponto físico de uma imagem digital e carrega o valor de sua cor. O valor da cor pode ser representado por um número, caso seja uma imagem monocromática, ou por uma lista ordenada, caso seja colorida. Essa lista ordenada de números varia de quantidade e valores conforme o modelo de cor utilizado que indica qual o espaço de cor a imagem utiliza (SHIRLEY; MARSCHNER, 2009).

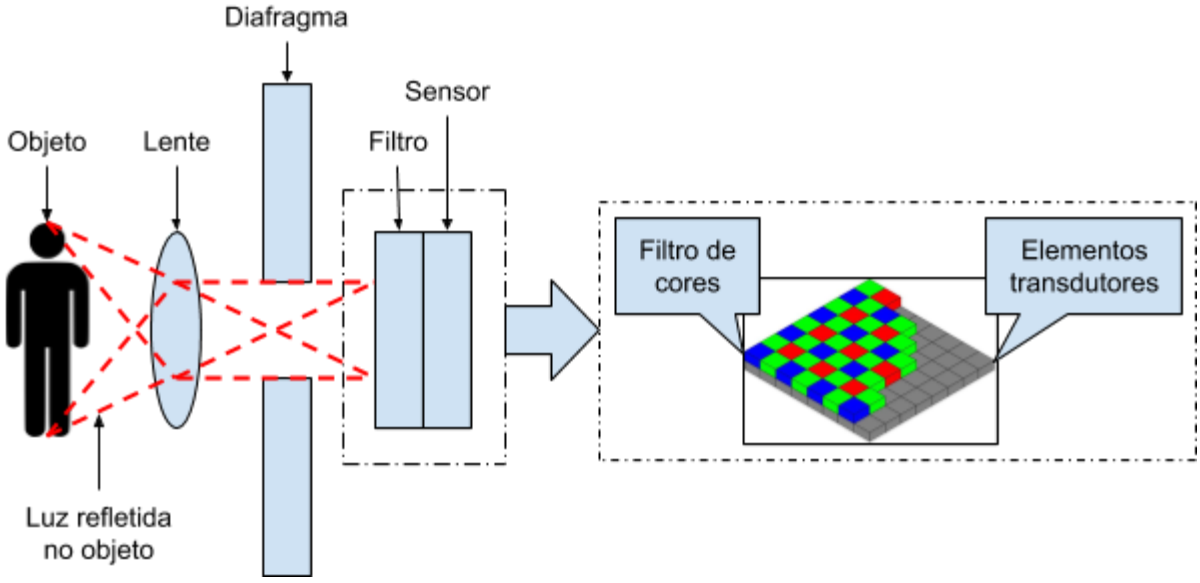
### 2.2.1 Aquisição de uma imagem digital

Na aquisição de uma imagem digital, primeiro ocorre a transdução da intensidade da luz captada a partir da reflexão no objeto de interesse, em corrente elétrica; posteriormente, essa corrente elétrica é digitalizada; depois ocorre o

tratamento desses valores digitalizados. As etapas desse processo (VILLEGAS, 2009) são descritas a seguir, e representadas nas figuras 1 e 2:

a) a luz refletida dos objetos que se deseja colher a imagem é filtrada por um conjunto de lentes, responsáveis por focalizar o objeto;
b) a luz focalizada é filtrada novamente pelo diafragma, que regula a quantidade de luz a ter sua intensidade registrada;
c) a luz remanescente passa por um filtro matricial de cores, a fim de transferir a intensidade luminosa de determinada cor em cada elemento do sensor;
d) a luz filtrada atinge o sensor de imagem, que também dispõe de elementos transdutores dispostos matricialmente;

Figura 1 – Esquema do caminho percorrido pela luz até o sensor de imagem



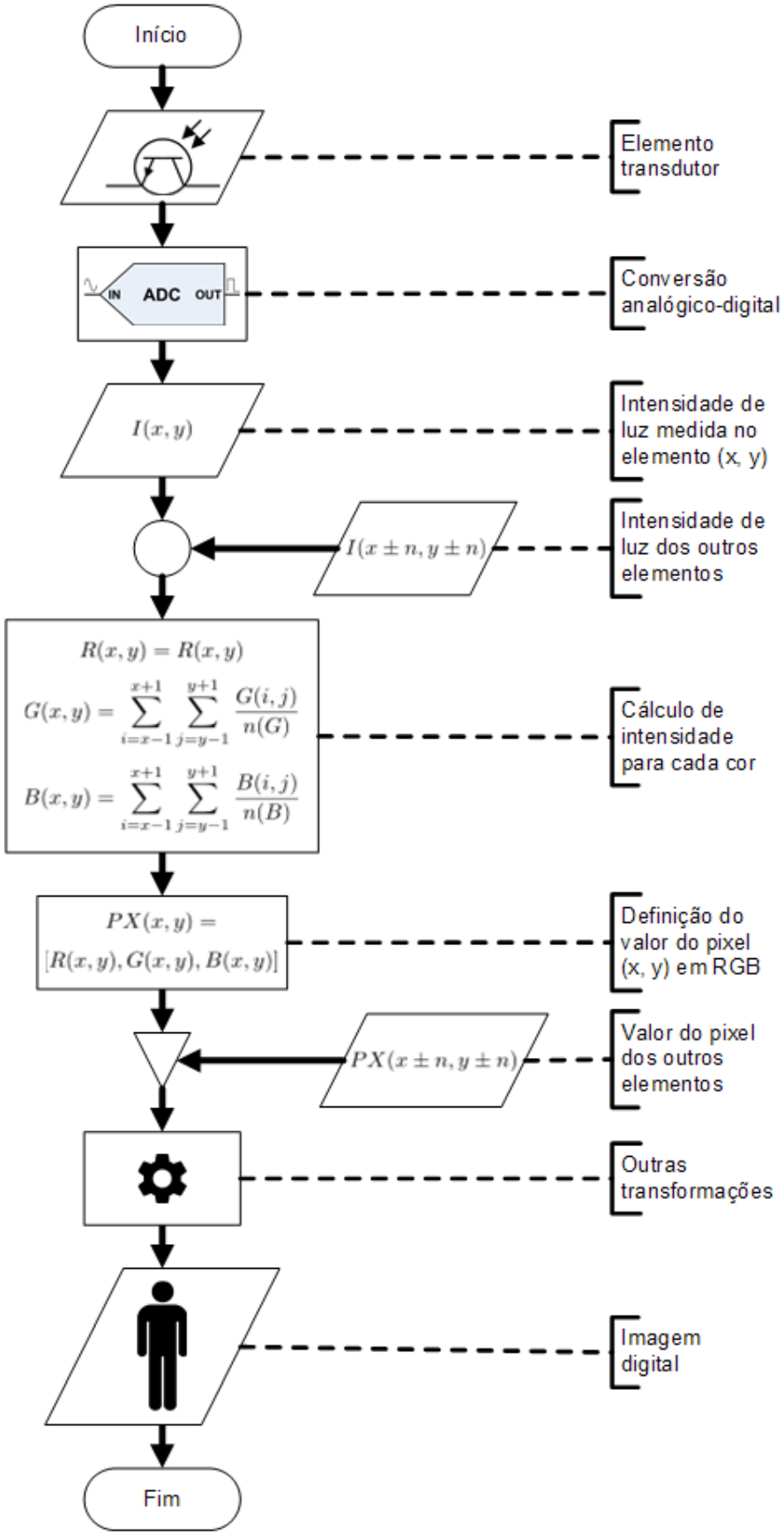

Fonte: elaborada pelo autor

e) cada elemento transdutor transforma a intensidade de luz filtrada em uma tensão elétrica de maneira proporcional;
f) a tensão elétrica de cada elemento transdutor é digitalizada, e então é obtida uma matriz da intensidade luminosa recebida em cada elemento sensor para a cor filtrada;
g) para cada elemento da matriz de intensidades, é feita uma estimativa da intensidade luminosa das outras duas cores do filtro (por exemplo,

interpolação dos valores vizinhos de mesma cor). Então é construído um vetor com o valor de intensidade captado e os valores de intensidade interpolados. Isso é feito na matriz toda, gerando uma matriz da intensidade luminosa das três cores detectadas pelo sensor de imagem (por exemplo em padrão RGB);

h) a matriz, que já pode ser considerada uma imagem, passa por diversas transformações, como balanço de branco, interpretação colorimétrica, correção de gama, redução de ruído, compressão, etc;
i) a imagem digital é salva na memória do dispositivo.

Figura 2 – Fluxograma da geração da imagem digital a partir do sensor




Fonte: elaborada pelo autor

## 2.3 MODELO DE COR

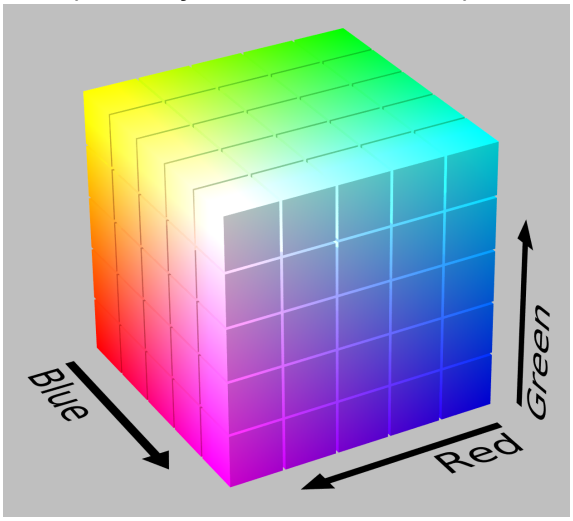
Modelo de cor é um modelo matemático abstrato que visa representar o espectro de cores num espaço n-dimensional, onde cada dimensão está relacionada a um componente formador da cor nesse modelo. Com isso, uma cor específica dentro de um modelo passa a ser representada por uma lista ordenada de números que representam os componentes formadores dessa cor (INTEL, 2018).

### 2.3.1 RGB

O modelo de cor RGB (*Red, Green, Blue*, em português, vermelho, verde, azul) é um modelo aditivo de cores, no qual os componentes formantes são as cores vermelha, verde e azul, ou seja, todas as cores são representadas por diferentes variações das intensidades das cores vermelha (*R*, *Red*), verde (*G*, *Green*) e azul (*B*, *Blue*). Com intensidade máxima para as três cores, obtém-se a cor branca, e com intensidade zero para as três cores, obtém-se a cor preta (INTEL, 2018; HEARN; BAKER, 1996).

Esse modelo é representado por um cubo num espaço de coordenadas cartesianas, no qual cada eixo representa a intensidade das cores formantes, conforme pode ser visto na figura 3.

Figura 3 – Representação do modelo RGB mapeado num cubo




Fonte: WIKIPEDIA (2018)

### 2.3.2 HSV

Apesar de ser o modelo do qual se deriva o funcionamento da maioria dos equipamentos eletrônicos reprodutores de imagens coloridas, a identificação de uma determinada cor a partir do modelo RGB (utilizando a lista ordenada) por um ser humano normal não é algo intuitivo (HEARN; BAKER, 1996; SMITH, 1978). Diversos modelos baseados em outros aspectos da cor, tendo como parâmetros: luminância, crominância (YCbCr, YUV), matiz, saturação, brilho (Munsell, NCS, HSL, HSV) foram criados a fim de se facilitar outras aplicações.

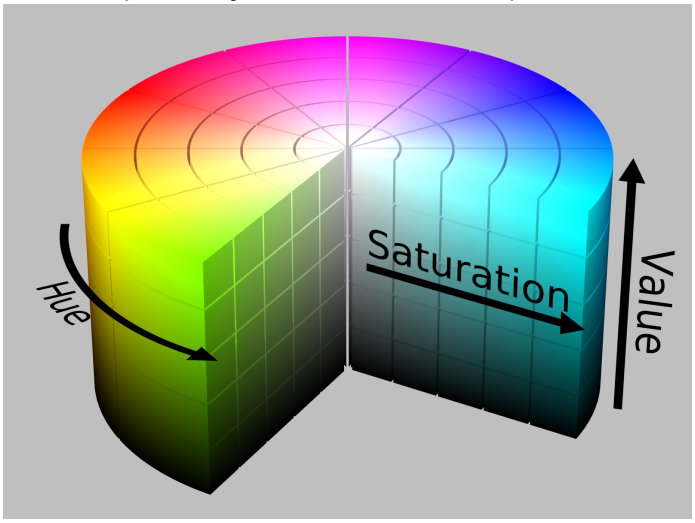
Durante a década de 70, Smith e outros pesquisadores da área de computação gráfica desenvolveram um modelo de cor derivado do modelo RGB no qual as cores são obtidas de maneira análoga ao modo como artistas plásticos obtém suas cores (SMITH, 1978). Esse modelo tem como parâmetros o matiz, a saturação relativa (BRIGGS, 2017) e o brilho relativo (BRIGGS, 2018), e trata-se de uma simples transformação do modelo RGB, de maneira que os cálculos computacionais para as transformações são simples. O modelo HSV (*Hue*, *Saturation*, *Value*, em português, matiz, saturação e valor; SMITH, 1978) representa o espectro de cores como um cilindro reto num sistema de coordenadas cilíndricas.

A coordenada cartesiana desse sistema, que é o eixo perpendicular à geratriz do cilindro, representa o brilho relativo da cor (*V*, *Value*). Em seu valor máximo (ponto na base superior do cilindro), obtém-se o brilho máximo possível da cor para aquele matiz, e conforme *V* se aproxima da origem (base inferior do cilindro), mais escura (menos brilhante) vai ficando a cor, até chegar no preto.

No plano polar, a coordenada radial, que é a reta do centro da base até a borda do cilindro, representa a saturação relativa da cor (*S*, *Saturation*). Em seu valor máximo (ponto na borda do cilindro), obtém-se a cor pura (sem esmaecimento) de um matiz com um brilho relativo específico, e conforme *S* se aproxima da geratriz do centro da base, mais esmaecida essa cor fica, até chegar a algum nível de cinza.

Já a coordenada angular representa o matiz (*H*, *Hue*), que é a cor pura, sem sombreamento ou clareamento, no espectro visível, ou seja, cada ângulo representa uma cor do espectro visível. O cilindro HSV pode ser visualizado na figura 4.

Figura 4 – Representação do modelo HSV mapeado num cilindro




Fonte: WIKIPEDIA (2018)

Por definição, na coordenada angular *H* tem-se o vermelho $(1, 0, 0)_{RGB}$ com valor $H = 0º$, o amarelo $(1, 1, 0)_{RGB}$ tem $H = 60º$, o verde $(0, 1, 0)_{RGB}$ tem $H = 120º$, o ciano $(0, 1, 1)_{RGB}$ tem $H = 180º$, o azul $(0, 0, 1)_{RGB}$ tem $H = 240º$ e o magenta $(1, 0, 1)_{RGB}$ tem $H = 300º$ (HEARN; BAKER, 1996). A variação do matiz em função de *H* pode ser vista na figura 5.

Figura 5 – Variação do matiz em função de *H*


Fonte: adaptado de WIKIPEDIA (2018)

A porção do espaço de cores dentro desse modelo cujo *S* é igual a zero (cruzando a geratriz do centro) é chamado de linha cinza, pois nele há apenas variações de tons de cinza, do preto ao branco. Nesse subespaço, *H* não tem significado (SMITH, 1978).

Com isso, uma cor agora passa a ser representada pelo seu matiz (a cor pura), saturação relativa (quão esmaecida ela é) e brilho relativo (o quão brilhante ela é), de maneira análoga ao que é feito por artistas plásticos (SMITH, 1978).

### 2.3.3 Nível de cinza (Luma)

O Luma (*Y'*) é o parâmetro que representa a porção acromática do píxel, ou seja, a intensidade absoluta de brilho de um píxel colorido. É formado pela soma de cada componente RGB do píxel com a compressão gama aplicada, multiplicado com sua proporção de contribuição para a percepção humana de brilho. É também interpretado como sendo o píxel em nível de cinza (BROWN, 2016).

O Luma é usado em espaços de cor nos quais se deseja separar a informação de cor (crominância) da informação de brilho (luminância) tais como Y'CbCr, Y'UV. Dada a natureza da visão humana de possuir uma sensibilidade maior às diferenças de luminância do que de crominância, esses espaços de cor são utilizados em sistemas de codificação de vídeo no qual a informação de crominância é subamostrada, o que traz maior eficiência na transmissão (JACK, 2005).

### 2.3.4 Imagem binária

Uma imagem binária é uma imagem cujos píxeis possuem apenas um canal no qual são admitidos apenas dois valores distintos, normalmente dois tons de cinza (preto e branco). Como consequência dessa característica, um píxel numa imagem binária pode ser armazenado em um bit, e isso faz com que uma imagem binária ocupe menos memória do que uma imagem em nível de cinza. Além disso, como as operações nos píxeis são lógicas (e não aritméticas), elas são mais rápidas (JAIN; KASTURI; SCHUNCK, 1995).

## 2.4 PROCESSAMENTO DIGITAL DE IMAGENS

Por Processamento Digital de Imagens entende-se a manipulação de uma imagem digital por um computador de modo que a entrada e a saída do processo sejam imagens. O desenvolvimento da área de processamento digital de imagens vem das necessidades de melhoria da informação visual para a interpretação humana e de processamento do conteúdo da imagem para armazenamento,

transmissão e representação para a percepção de máquinas autônomas (GONZALES; WOODS, 2002).

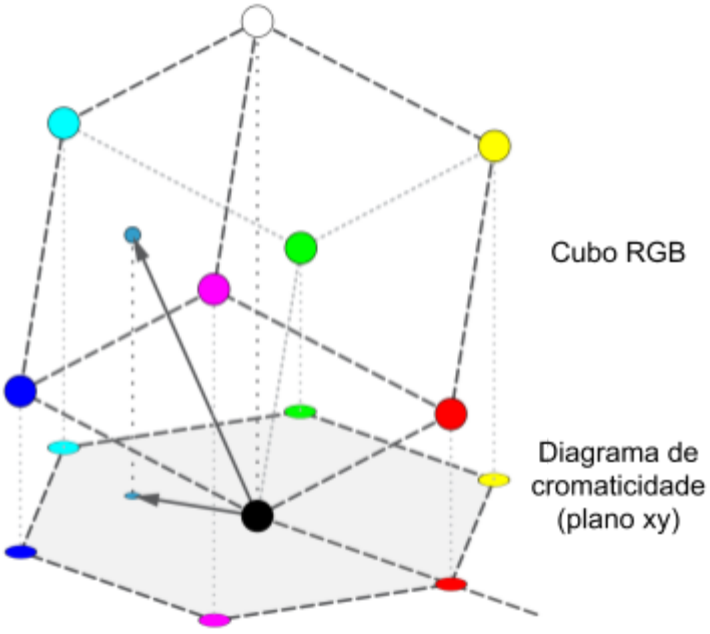
.

### 2.4.1 Conversão RGB para HSV

#### *2.4.1.1 Conversão do cubo RGB para o cilindro HSV*

Primeiro, é feita uma translação do cubo RGB de modo que a origem continue sendo o preto $(0, 0, 0)_{RGB}$ e o eixo z cruze o branco $(1, 1, 1)_{RGB}$ (figura 6). Isso faz com que a projeção do cone RGB no plano xy forme um hexágono, no qual seus vértices são o vermelho $(1, 0, 0)_{RGB}$, o amarelo $(1, 1, 0)_{RGB}$, o verde $(0, 1, 0)_{RGB}$, o ciano $(0, 1, 1)_{RGB}$, o azul $(0, 0, 1)_{RGB}$ e o magenta $(1, 0, 1)_{RGB}$ (WIKIPEDIA, 2018).

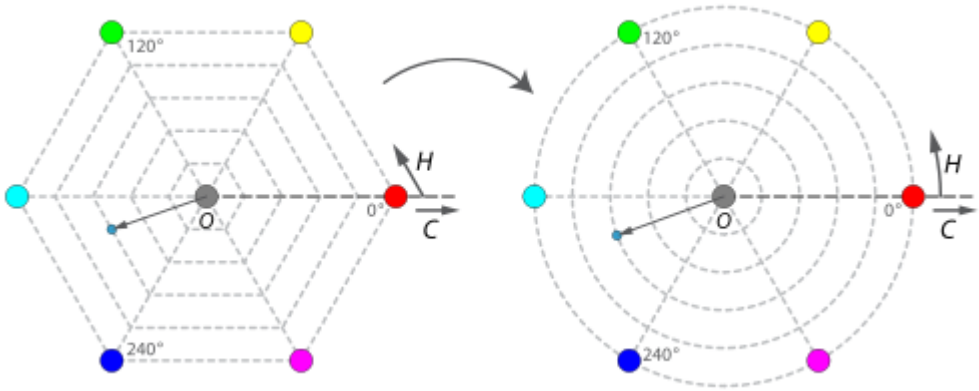
Figura 6 – Translação e projeção do cubo RGB no plano xy




Fonte: adaptado de WIKIPEDIA (2018)

Após isso é feita uma transformação nesse hexágono projetado de modo tal que ele vire um círculo, que será a base do cilindro (figura 7). Esse círculo forma o diagrama de cromaticidade do modelo HSV.

Figura 7 – Transformação do hexágono projetado em círculo




Fonte: adaptado de WIKIPEDIA (2018)

Após isso, esse círculo tem seu raio normalizado e sua base é trasladada no eixo z até z = 1, formando o cilindro HSV (figura 4).

### *2.4.1.2 Conversão das listas ordenadas*

Para a conversão de uma lista ordenada RGB para uma lista ordenada HSV, primeiro se extrai da lista ordenada RGB os seguintes parâmetros intermediários (WIKIPEDIA, 2018)

$$MAX = \max(\{R, G, B\}) \tag{1}$$

$$MIN = \min(\{R, G, B\}) \tag{2}$$

$$C = MAX - MIN \tag{3}$$

$$H' = \begin{cases} 0, & C = 0 \\ \frac{G-B}{C} \mod 6, & MAX = R \\ 2 + \frac{B-R}{C}, & MAX = G \\ 4 + \frac{R-G}{C}, & MAX = B \end{cases} \tag{4}$$

e então, a partir desses parâmetros, obtêm-se os valores da lista ordenada HSV (WIKIPEDIA, 2018), fazendo

$$H = 60° \cdot H' \tag{5}$$

$$S = \frac{C}{MAX} \tag{6}$$

$$V = MAX \tag{7}$$

### 2.4.2 Conversão RGB para nível de cinza

A conversão de uma lista ordenada RGB para nível de cinza é feita considerando que os componentes da lista ordenada já estão com a compressão gama aplicada e calculando o *Y'* dessa lista ordenada, conforme descrito na equação 8.

$$Y' = 0,299R' + 0,587G' + 0,114B' \tag{8}$$

Esta equação está definida pela recomendação ITU-R (2011) BT.601-7 e é utilizada pelo método do OpenCV *Imgproc.cvtColor* que converte uma imagem em RGB para nível de cinza (BRADSKI, 2000).

### 2.4.3 Conversão nível de cinza para binário

Um dos métodos para realizar a conversão de um píxel em nível de cinza para binário, e que foi utilizado neste aplicativo, é um processo chamado binarização por limiarização. Nele, cada píxel da imagem é comparado com um valor de limiar de binarização, que está dentro da escala de nível de cinza. Se o valor do píxel em escala de cinza (*p*) for menor que o limiar de binarização (*l*), é atribuído à ele o menor valor da escala binária. Caso contrário, é atribuído o maior valor da escala binária (SZELISKI, 2011), conforme descrito pela equação 9.

$$T(p,l) = \begin{cases} 1, & p \geq l \\ 0, & p < l \end{cases} \tag{9}$$

### 2.4.4 Transformações morfológicas

As operações de transformação em imagens binárias são chamadas de transformação morfológicas, uma vez que elas alteram a forma dos objetos contidos nela. Tais transformações resumem-se em fazer uma convolução da imagem com um elemento estruturante binário e selecionar uma saída dependendo do resultado da convolução. O elemento estruturante pode ter qualquer forma, desde uma simples janela 3×3 até estruturas mais complexas (SZELISKI, 2011). Elas também

são aplicações da equação 2, mas tendo como entrada o resultado da convolução da imagem (*i*) com o elemento estruturante (*e*). As transformações utilizadas nesse projeto estão descritas a seguir.

#### *2.4.4.1 Dilatação:*

$$dilatar(i, e) = T((i \circledast e), 1) \tag{10}$$

a) aumenta o tamanho dos objetos;
b) diminui o tamanho ou preenche completamente os buracos;
c) conecta objetos próximos.

#### *2.4.4.2 Erosão:*

$$erodir(i, e) = T((i \circledast e), \sum^{e} p) \tag{11}$$

a) diminui o tamanho dos objetos;
b) apaga detalhes e objetos menores que o elemento estruturante;
c) aumenta o tamanho dos buracos;
d) separa objetos conectados.

#### *2.4.4.3 Abertura:*

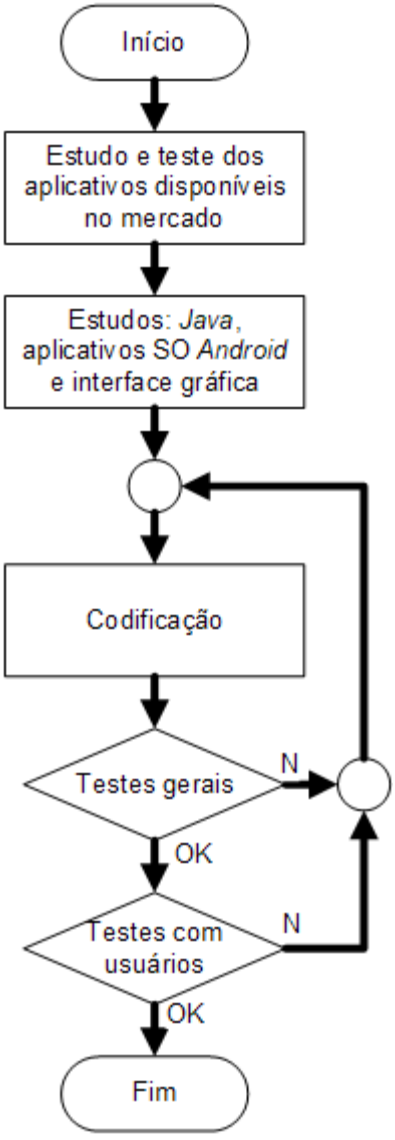
$$abrir(i, e) = dilatar(erodir(i, e), e) \tag{12}$$

a) remove detalhes e objetos menores que o elemento estruturante;
b) mantém quase totalmente o tamanho e a forma dos objetos.

# 3 METODOLOGIA

A metodologia utilizada neste projeto está ilustrada no fluxograma da figura 8.

Figura 8 – Representação da metodologia




Fonte: elaborada pelo autor

A princípio foi feito um estudo dos aplicativos disponíveis no mercado que executam funções de identificação de cor. Nesses aplicativos foi feito um levantamento das suas características, como e quão rápido é dada a informação da(s) cor(es) identificada(s), se possui e como é feita a descrição falada da(s) cor(es), entre outros.

No desenvolvimento do trabalho, houve uma etapa de aprendizagem e treinamento da linguagem de programação *Java* e de desenvolvimento de aplicativos para o SO *Android*. Tais treinamentos deram-se por meio de cursos gratuitos *online*. Em conjunto com os treinamentos, a programação inicial do aplicativo foi feita. O programa foi codificado no ambiente de desenvolvimento integrado (IDE, *Integrated Development Environment*) *Android Studio*, utilizando a linguagem de programação *Java*, tendo como alvo dispositivos com versão do SO *Android* v5.0 (*Lollipop*) ou posteriores.

Parte das classes e métodos empregados na codificação da aplicação relacionados à interface de usuário, estão presentes na estrutura da API *Java* do SO *Android*. Outra parte das classes e métodos utilizados estão presentes nas APIs de manipulação e processamento digital de imagens contidas na biblioteca OpenCV.

Também foi feita uma pesquisa sobre qual deveria ser a melhor abordagem para realizar a identificação de cédulas de Real. As mesmas classes da estrutura da API *Java* do SO *Android* e de manipulação e processamento digital de imagens contidas na biblioteca OpenCV utilizadas para identificar cor foram usadas nesse caso.

Após a codificação, uma versão executável do aplicativo foi gerada pela IDE *Android Studio* para ser testada ou num *smartphone* com o SO *Android*, ou no *Android Emulator*. Caso o aplicativo não estivesse funcionando corretamente, o mesmo era analisado, depurado e corrigido.

Após o desenvolvimento de uma versão do aplicativo mais estável e completa, foi o momento para realizar testes para a verificação de potenciais restrições físicas, como o comportamento do aplicativo ao se mudar o ambiente, a iluminação, a exposição a diferentes tipos de objetos, etc. As informações obtidas a partir desses testes foram utilizadas para realizar mais correções e melhorias ao aplicativo.

Quando a parte de identificação de cores estava completa, houve a etapa de ajuste da interface gráfica. Nesta fase, foram feitos testes para verificar como o aplicativo se comporta ao funcionar em conjunto com o *TalkBack*.

Após essa fase, o aplicativo encontra-se estável e utilizável o suficiente para ser disponibilizado ao público interessado para a função de identificação de cores. A

funcionalidade de identificação de cédulas de dinheiro (Real) não pôde ser finalizada por limitação de tempo, mas poderá ser integrada ao aplicativo com facilidade em trabalho futuro, aproveitando-se da estrutura de programação já desenvolvida.

## 3.1 OPENCV

O OpenCV (BRADSKI, 2000) é uma biblioteca de *software* de visão computacional e aprendizado de máquina em código aberto. O OpenCV foi construído para fornecer uma infra-estrutura comum para aplicações de visão computacional e para acelerar o uso da percepção da máquina nos produtos comerciais. Possui interfaces *C++*, *Python*, *Java* e *MATLAB* e suporta os SO *Windows*, *Linux*, *Android* e *MacOS*. O OpenCV é escrito nativamente em *C++* e inclina-se principalmente para aplicativos de visão em tempo real (BRADSKI, 2000).

Em suas funções básicas, o OpenCV permite capturar imagem da câmera, exibir imagens na tela e acessar os dados dessa imagem digital. Para realizar a captura de imagens direto da câmera e exibí-las na tela, o OpenCV possui uma classe específica que realiza ambos, a *JavaCameraView* (BRADSKI, 2000). Ela possui um elemento de interface de usuário que exibe o que a câmera está capturando em tempo real, conforme as características da câmera. Ela realiza a captura de imagens da câmera conforme as características da câmera e as transforma numa matriz do OpenCV (*Mat*), sendo que as informações de cor podem ser em nível de cinza ou em RGB.

Para realizar transformações nas imagens capturadas, a fim de modificar ou aferir características, utiliza-se o módulo de processamento de imagem *ImgProc*. Nele é possível separar uma região de interesse da imagem para análise, converter a informação de cor para diferentes espaços de cores, entre outras operações (BRADSKI; KAEHLER, 2008).

## 3.2 DEFINIÇÃO DE COR NO APLICATIVO

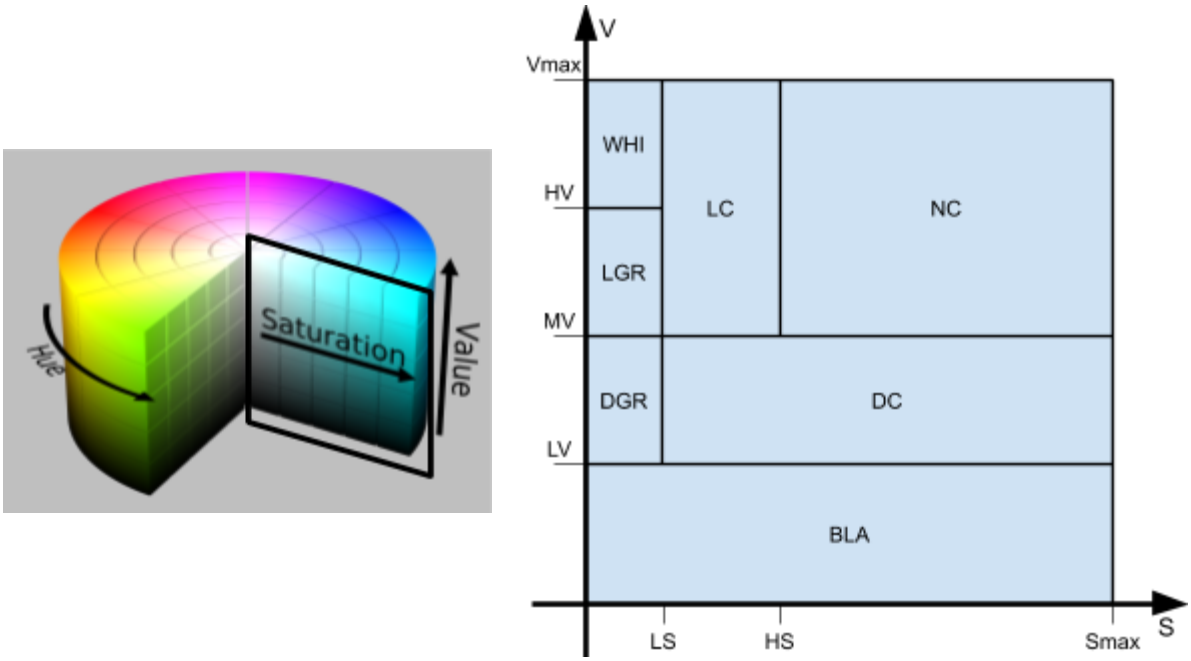
Para definir a cor no contexto do aplicativo, o modelo HSV foi dividido em 40 subespaços, onde cada subespaço é considerado como uma cor. Para tanto, foram

definidos 3 limiares para *V* e 2 limiares para *S*, que formam 7 subespaços no plano CV e 12 limiares para *H*, que formam 12 subespaços no plano HS. Além dessas 40 cores, foi definida também uma cor para ser atribuída nos casos de invalidação (*INV*).

### 3.2.1 Divisão do plano SV

Para um *H* fixo, os níveis de *S* de saturação relativa baixa (*LS*) e saturação relativa alta (*HS*) e os níveis de *V* de brilho relativo baixo (*LV*), brilho relativo médio (*MV*) e brilho relativo alto (*HV*) formam áreas que representam a classificação da cor em relação à sua tonalidade (*T*), que podem ser: preta (*BLA*), cinza escuro (*DGR*), cor escura (*DC*), cinza claro (*LGR*), branca (*WHI*), cor clara (*LC*) e cor normal (*NC*), conforme mostrado na figura 9. O subespaço do plano SV formado pela união de *BLA*, *DGR*, *LGR* e *WHI* são análogos à linha de cinza da definição do modelo HSV e, neste contexto, definido como área de cinza.

Figura 9 – Cilindro HSV com detalhe do plano SV e o mesmo representado em detalhes com seus limiares




Fonte: elaborada pelo autor

### 3.2.2 Divisão do plano HS

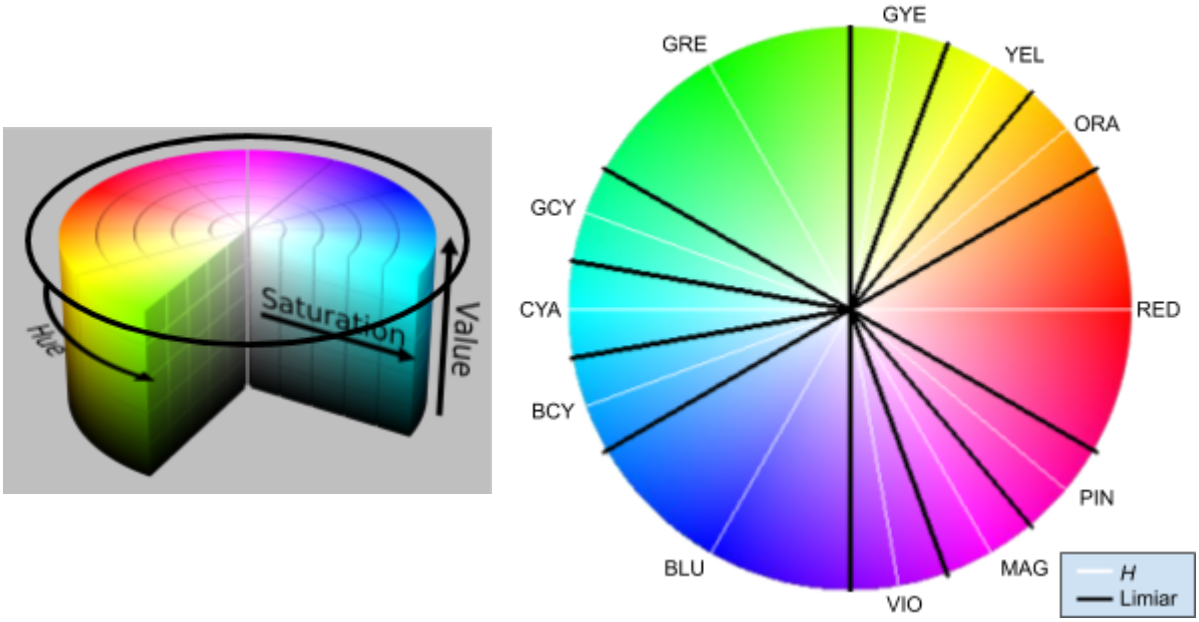
Conforme já explicado, para um *V* fixo, o plano HS forma o diagrama de cromaticidade, e nele temos os matizes vermelho $(1, 0, 0)_{RGB}$ em *H* = 0º (*RED*), amarelo $(1, 1, 0)_{RGB}$ em *H* = 60º (*YEL*), verde $(0, 1, 0)_{RGB}$ em *H* = 120º (*GRE*), ciano $(0, 1, 1)_{RGB}$ em *H* = 180º (*CYA*), azul $(0, 0, 1)_{RGB}$ em *H* = 240º (*BLU*) e magenta $(1, 0, 1)_{RGB}$ em *H* = 300º (*MAG*, HEARN; BAKER, 1996). Para permitir uma descrição mais ampla, foi adicionado um matiz entre cada par sequencial dos matizes acima, nomeadas laranja (*ORA*), verde amarelado (*GYE*), verde ciano (*GCY*), azul ciano (*BCY*), violeta (*VIO*) e rosa (*PIN*).

A partir disso, foi definido que a proporção da medida angular dos matizes de cores primárias (*RED*, *GRE* e *BLU*) para as demais definidas, seria de três para um, e a bissetriz de cada medida angular seria *H*. A partir dessas definições, derivam-se:

a) a medida angular do arco dos matizes de cor primária é de 60º, e dos demais matizes é de 20º;
b) o *H* e a lista ordenada RGB de referência para os matizes adicionados;
c) o arco limiar de um matiz, como sendo metade da medida angular de seu arco;
d) o limiar superior de matiz (*HUT*) como sendo a soma de *H* com seu arco limiar;
e) o limiar inferior de matiz (*HLT*) como sendo a subtração de *H* com seu arco limiar, sendo este igual ao *HUT* do matiz adjacente ao *HLT* (por exemplo, $HLT_{ORA} = HUT_{RED}$).

Com isso, pôde-se determinar o arco limiar de cada cor dessa escala, dividindo o diagrama de cromaticidade em 12 áreas, conforme figura 10.

Figura 10 – Cilindro HSV com detalhe do plano HS e o mesmo representado em detalhes com seus limiares




Fonte: adaptado de WIKIPEDIA (2018)

As demais informações relativas às cores definidas no aplicativo encontram-se na tabela 1.

Tabela 1 – Relação de identificação, valor RGB de referência, *H* e limiares

| **IDENTIFICAÇÃO** | | **RGB** | | | **HSV(º)** | | |
|---|---|---|---|---|---|---|---|
| **CLASSE** | **TIPO** | ***R*** | ***G*** | ***B*** | ***HLT*** | ***H*** | ***HUT*** |
| *BLA* | tom | 0 | 0 | 0 | - | - | - |
| *DGR* | tom | - | - | - | - | - | - |
| *LGR* | tom | - | - | - | - | - | - |
| *DC* | tom | - | - | - | - | - | - |
| *LC* | tom | - | - | - | - | - | - |
| *NC* | tom | - | - | - | - | - | - |
| *RED* | matiz | 1 | 0 | 0 | 330 | 0 | 30 |
| *ORA* | matiz | 1 | ⅔ | 0 | 30 | 40 | 50 |
| *YEL* | matiz | 1 | 1 | 0 | 50 | 60 | 70 |
| *GYE* | matiz | ⅔ | 1 | 0 | 70 | 80 | 90 |
| *GRE* | matiz | 0 | 1 | 0 | 90 | 120 | 150 |
| *GCY* | matiz | 0 | 1 | ⅔ | 150 | 160 | 170 |
| *CYA* | matiz | 0 | 1 | 1 | 170 | 180 | 190 |
| *BCY* | matiz | 0 | ⅔ | 1 | 190 | 200 | 210 |
| *BLU* | matiz | 0 | 0 | 1 | 210 | 240 | 270 |
| *VIO* | matiz | ⅔ | 0 | 1 | 270 | 280 | 290 |
| *MAG* | matiz | 1 | 0 | 1 | 290 | 300 | 310 |
| *PIN* | matiz | 1 | ⅔ | 1 | 310 | 320 | 330 |
| *WHI* | tom | 1 | 1 | 1 | - | - | - |
| *INV* | cor | - | - | - | - | - | - |

Fonte: elaborada pelo autor

## 3.3 FLUXO GERAL

Conforme representado na figura 11, o aplicativo inicia-se ativando a câmera e exibindo a imagem da captura de quadro (*CQ*), na tela com um retângulo vermelho no centro, representando a região de interesse, a partir da qual será feita a classificação e contagem de cores, e dois botões alinhados e posicionados na parte inferior da tela (ver figura 17).

Nessa tela, a cada captura de quadro feita pela câmera, é feita a aferição da quantidade de píxeis invalidados por intensidade da captura inteira (*mapLuz(CQ)*). Dependendo da proporção de píxeis invalidados (*PL*), o smartphone vibra em pulsos, cuja duração varia conforme a tabela 2 abaixo.

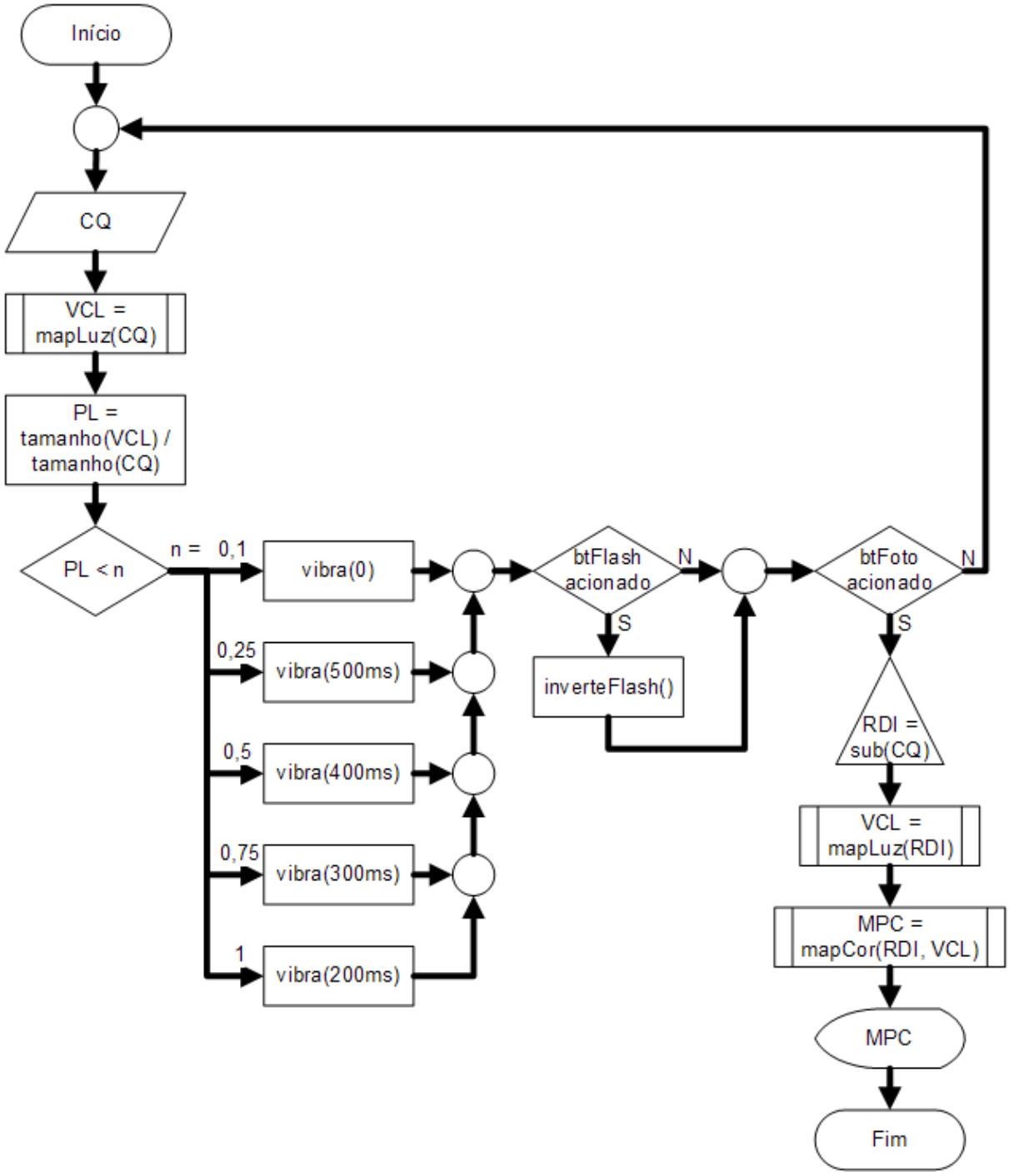
Tabela 2 – Relação das faixas de proporção de píxeis invalidados e seus respectivos tempos de vibração e pausa no modo de captura

| **Proporção** | **Tempo de vibração (ms)** | **Tempo de pausa (ms)** |
|---|---|---|
| menos que 10% | - | - |
| entre 10% e 25% | 500 | 500 |
| entre 25% e 50% | 400 | 400 |
| entre 50% e 75% | 300 | 300 |
| mais que 75% | 200 | 200 |

Fonte: elaborada pelo autor

O botão à esquerda é o de ativação do flash (*btFlash*). Ao tocá-lo, o LED do *flash* da câmera é ativado caso esteja desativado e vice-versa (*inverteFlash()*). O botão à direita é o de fazer a classificação (*btFoto*), ao tocar nele, uma cópia da imagem contida dentro da região de interesse (*RDI*) é feita (*sub(CQ)*). Dessa imagem, os contornos das regiões com píxel invalidado (*VCL*) são mapeados (*mapLuz(RDI)*) e, em seguida, a classificação e contagem das cores de cada píxel (*MCI*) é feita (*mapCor(RDI)*). Após isso, essa classificação é usada para construir um texto que dita as cores presentes na *RDI*, com suas respectivas proporções em ordem decrescente e que tenham mais que 4,5% de proporção.

Figura 11 – Fluxograma de funcionamento do aplicativo




Fonte: elaborada pelo autor

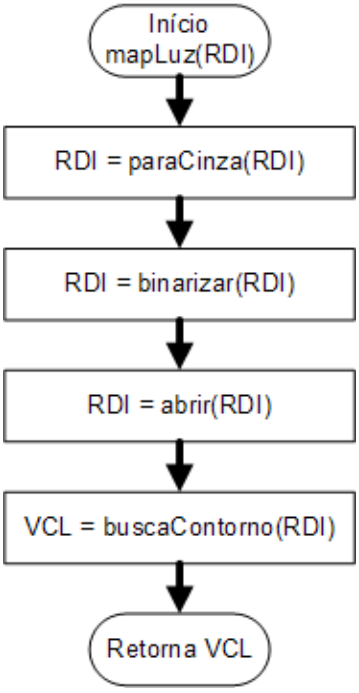
## 3.4 ALGORITMO IDENTIFICADOR DE FONTE DE LUZ

Durante os testes com o algoritmo classificador de cor implementado, verificou-se que a presença de fontes de luz na região captada pela câmera do smartphone acabava por fazer com que fosse registrado valores de *H* desses píxeis

(e de parte do entorno deles) diferente do que se esperava, gerando resultados incorretos. Para solucionar esse problema, foi desenvolvido um algoritmo para identificar, contabilizar e mapear os píxeis que, por consequência desse problema, estão com *H* comprometido.

Conforme representado na figura 12, o método inicia transformando a *RDI* (que está em RGB) para nível de cinza (*paraCinza(RDI)*); posteriormente, a *RDI* em nível de cinza é binarizada (*binarizar(RDI)*) por um processo de limiarização; então, é feita uma abertura da *RDI* binarizada (*abrir(RDI)*). Com esse processo, é obtida uma máscara com as regiões afetadas pela presença da fonte de luz em branco e o resto da imagem em preto. Dessa imagem é extraído um vetor de contornos de luz (*buscaContorno(RDI)*) que contém o mapeamento dos píxeis classificados como fonte de luz (*VCL*).

Figura 12 – Algoritmo identificador e mapeador de fonte de luz



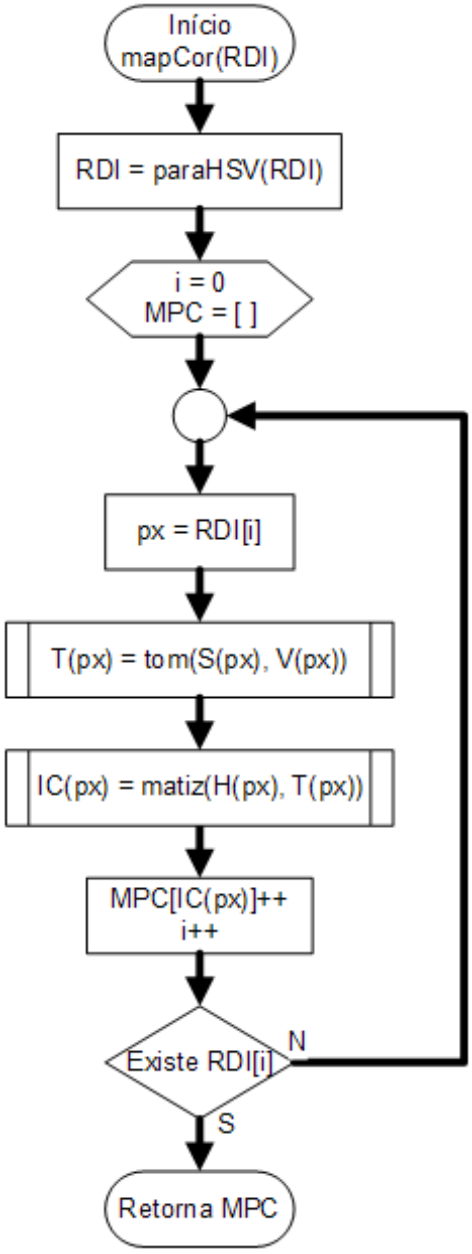

Fonte: elaborada pelo autor

## 3.5 ALGORITMO CLASSIFICADOR E CONTADOR DE COR

Conforme representado na figura 13, o algoritmo classificador e contador de cor (*mapCor(RDI)*) é iniciado convertendo a *RDI*, em formato RGB, para o modelo

HSV (*paraHSV(RDI)*). Após isso, para cada píxel da *RDI* é verificado se o mesmo não está contido no *VCL*. Se estiver, ele é classificado como *INV*. Se não, a sua *T* é aferida (*tom(S(px), V(px))*) e, na sequência, o seu matiz (*matiz(H(px), T(px))*), que tem como resultado um índice de cor (*IC(px)*), que representa uma cor na escala de cores do aplicativo. Esse índice é contabilizado no *MCI* (*MCI[IC(px)]*++), que armazena a quantidade de píxeis de cada cor na imagem. Após a classificação de toda *RDI*, o *MCI* é utilizado para construir um gráfico de barras da proporção das cores na *RDI*, que é mostrado na tela em sequência.

Figura 13 – Algoritmo classificador e contador de cor

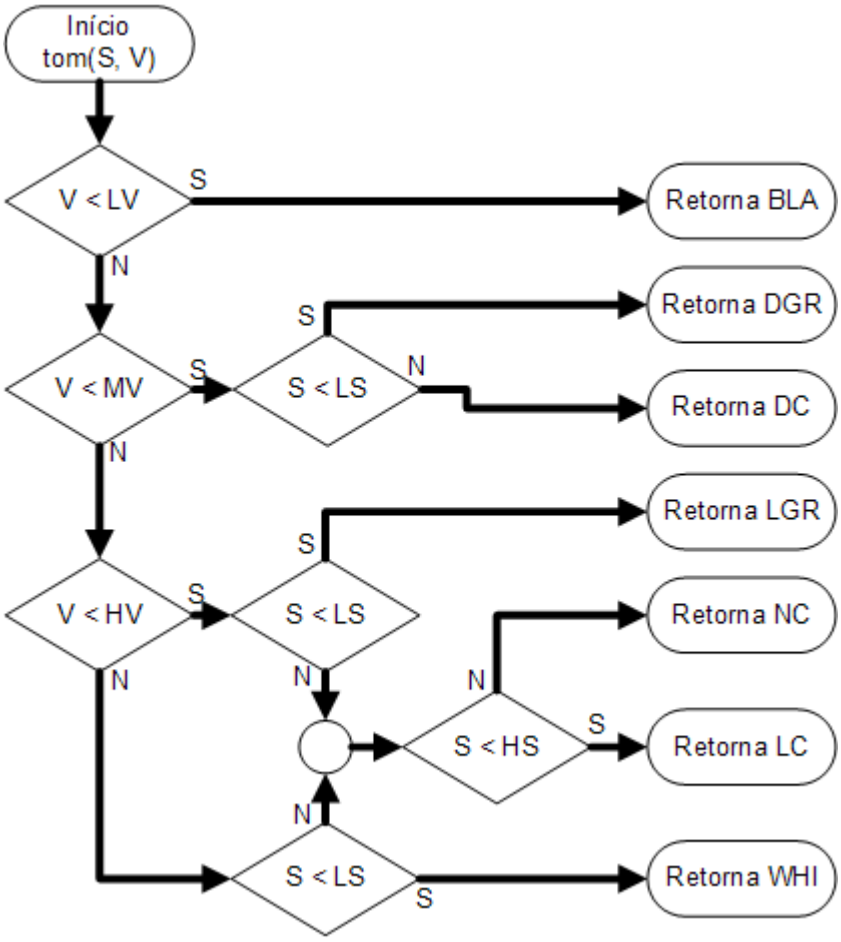



Fonte: elaborada pelo autor

### 3.5.1 Classificação da tonalidade

Na classificação de tonalidade (*tom(S, V)*), *S* e *V* do píxel analisado são comparados com os limiares de *S* e *V* para aferir em qual região de *T* a cor do píxel se encontra, que é então retornado. A classificação é feita conforme o fluxograma da figura 14.

Figura 14 – Algoritmo classificador de tonalidade

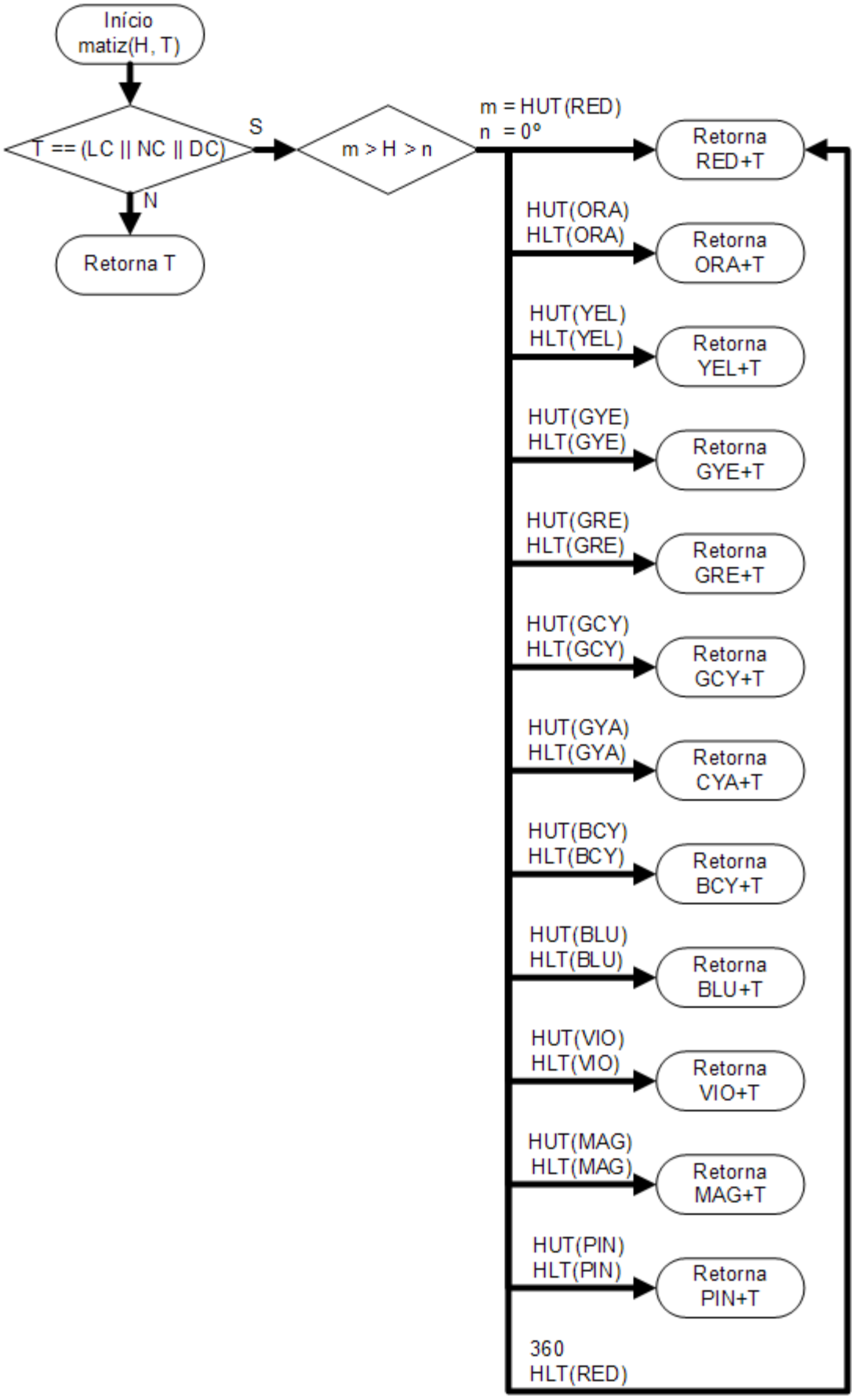



Fonte: elaborada pelo autor

### 3.5.2 Classificação do matiz

Na classificação do matiz (*matiz(H, T)*), primeiro é verificado se a *T* do píxel analisado está na área de cinza. Se positivo, *T* é retornado diretamente como sendo a cor. Caso contrário, é feita a comparação do *H* com os limiares de cada matiz (*HUT* e *HLT*) para aferir em qual das região de cromaticidade o *H* do píxel em questão se encontra. Após isso, junta-se o resultado dessa comparação com *T*, formando a cor conforme definido para o aplicativo, conforme descrito pela figura 15.

Figura 15 – Algoritmo classificador de matiz e construtor de cor




Fonte: elaborada pelo autor

# 4 RESULTADOS

## 4.1 CARACTERÍSTICAS DOS OUTROS APLICATIVOS

Nos aplicativos testados, o sistema de identificação de cores funciona de maneira bem parecida: na tela de identificação de cores, é mostrado a captura da câmera com um alvo no centro da tela, que indica o ponto na imagem do qual será feita a identificação da cor. O sistema realiza a identificação de apenas uma cor, sendo essa cor a resposta do sistema.

Apesar disso, a identificação de cores feita pelos aplicativos testados funciona bem, ocorre quase que instantaneamente, mas tende a ficar imprecisa com variações de luminosidade. Seguem abaixo outras características dos aplicativos avaliados, também apresentadas de forma resumida no quadro 1.

Quadro 1 – Características dos aplicativos testados

| **Aplicativo** | **OS** | **Descrição falada** | **Idioma** | | **Anúncio** |
|---|---|---|---|---|---|
| | | | **Menus** | **Áudio** | |
| *Microsoft Seeing AI* | *iOS* | próprio | inglês | inglês | não |
| *Novartis ViaOpta Daily* | ambos | próprio | português | português | não |
| *sadens Studio Color Detector* | *Android* | próprio | inglês | inglês | sim |
| *mobialia.com Color Detector* | *Android* | SO | inglês | SO | não |
| *RamelTec Detector de Cor* | *Android* | próprio | português | português | sim |

Fonte: elaborada pelo autor

*Microsoft Seeing AI* (*iOS*):

a) os menus escritos do aplicativo não estão disponíveis em português.
b) possui seu próprio sistema de descrição falada, mas em inglês. Textos em português são falados com sotaque inglês.
c) identificador de cor: funciona bem e rápido.
d) leitura de textos curtos: funciona muito rapidamente.
e) leitura de documentos: executa com algumas falhas na identificação de alguns caracteres.
f) identificação de produtos: funciona bem.

g) descrição de pessoas: funciona bem.
h) descrição de cenários: funciona com limitações.
i) descritor de luminosidade por resposta sonora: funciona bem.

*Novartis ViaOpta Daily* (*Android* e *iOS*):

a) os menus escritos do aplicativo estão disponíveis em português.
b) possui o próprio sistema de descrição falada, em português.
c) identificador de cor: funciona bem e rápido; a área do identificador de cor é maior; se há mais de uma cor na área de identificação, o aplicativo avisa que não consegue identificar.
d) identificador de cédula: funciona bem e rápido; utiliza o flash desnecessariamente as vezes.
e) identificador de objeto: funciona bem; precisa consultar a *internet*; a câmera é inicializada com zoom e não tem como reduzir.
f) identificador de texto: funciona bem; precisa consultar *internet*; não reconhece o acento circunflexo, e dita “acento” sempre que o mesmo aparece; funciona apenas com o celular tirando a foto do texto na posição retrato.
g) identificador de cena: não funciona; precisa consultar a *internet*; a câmera é inicializada com zoom e não tem como reduzir.

*sadens Studio Color Detector* (*Android*):

a) os menus escritos do aplicativo não estão disponíveis em português.
b) possui o próprio sistema de descrição falada, mas em inglês.
c) identificador de cor: funciona bem e rápido.
d) possui propaganda.

*mobialia.com Color Detector* (*Android*):

a) os menus escritos do aplicativo não estão disponíveis em português.
b) utiliza o Google TalkBack como sistema de descrição falada, ou seja, em português. Contudo, a parte escrita está em inglês, logo tudo é dito em inglês.
c) identificador de cor: funciona bem e rápido.

*RamelTec Detector de Cor* (*Android*):

a) os menus escritos do aplicativo estão disponíveis em português.
b) possui o próprio sistema de descrição falada em português.
c) identificador de cor: funciona bem e rápido.
d) possui propaganda.

## 4.2 APLICATIVO DESENVOLVIDO

O aplicativo foi desenvolvido com a linguagem de programação *Java* através da IDE *Android Studio*, utilizando o pacote de desenvolvimento de *software* (SDK, *Software Development Kit*) do SO *Android* e as bibliotecas OpenCV e *GraphView* (GEHRING, [s.d.]). O aplicativo possui três telas: captura de imagem, gráfico de cores e descrição.

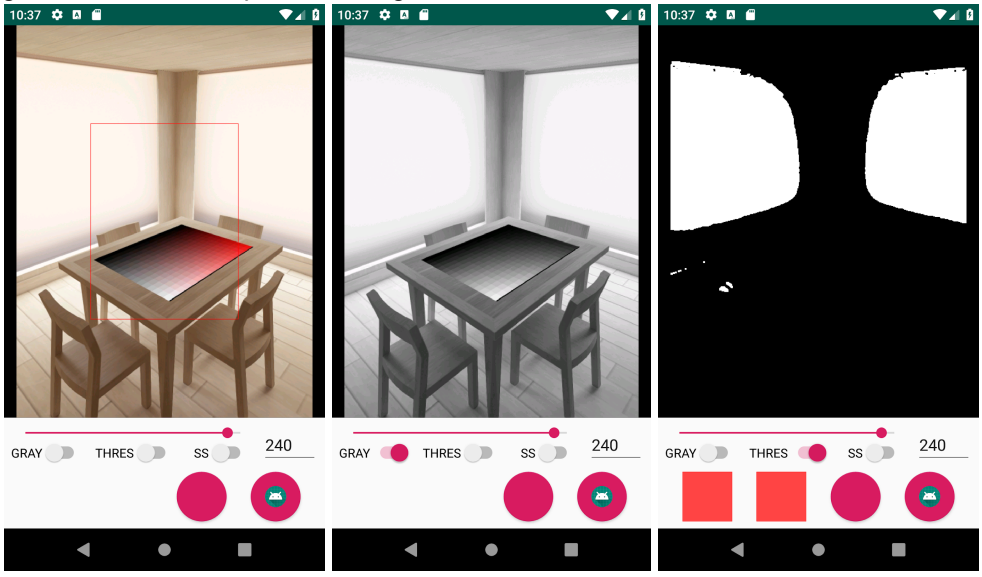
### 4.2.1 Captura de imagem

Ao abrir o aplicativo, o mesmo realiza o acesso à câmera e inicia a exibição do conteúdo capturado pela mesma na tela através de uma instância do componente *JavaCameraView*. Na parte central da tela, há o contorno em vermelho de um retângulo que tem a mesma proporção de largura e comprimento do conteúdo capturado pela câmera e um quarto de sua área para indicar a *RDI*.

Na parte inferior da tela, há uma barra de busca e um campo de texto editável que mostram e permitem alterar o valor do limiar de binarização. Abaixo da barra de busca, há a chave *GRAY* (em português, cinza) que, quando acionada, faz com que a exibição feita pelo *JavaCameraView* seja a captura da câmera em nível de cinza. À direita da chave *GRAY* há a chave *THRES* (*threshold*, em português, limiar) que, quando acionada, faz com que a exibição feita pelo *JavaCameraView* seja a captura da câmera binarizada com o valor de limiar de binarização definido campo de texto editável. À direita da chave *THRES*, há a chave *SS* (*Save Screenshot*, em português, salvar captura de tela) que, quando acionada, salva na memória interna do smartphone a captura de tela de todas as telas do fluxo, assim que é solicitado a mudança de tela.

Mais abaixo há dois botões, o de flash e o de classificação: o botão de flash aciona ou desativa o flash do dispositivo; o de classificação inicia o fluxo de extração e classificação da *RDI* e muda para a tela de gráfico de cores. A figura 16 mostra uma captura dessa tela utilizando o emulador do SO *Android*.

Figura 16 – Tela de captura de imagem nos modos normal, nível de cinza e binarizada

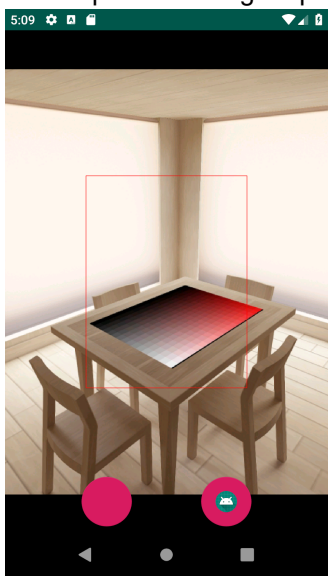



Fonte: elaborada pelo autor

#### *4.2.1.1 Captura de imagem para usuário final*

A tela de captura de imagem para o usuário final (figura 17) funciona como a anterior, porém o componente *JavaCameraView* ocupa a tela toda e possui apenas os botões, que estão sobrepostos à *JavaCameraView*, posicionados diferentemente e, ao acionar o botão de classificação, após o término da mesma, a tela seguinte será a tela de descrição de resultados, não passando pela tela de gráfico de cores.

Figura 17 – Tela de captura de imagem para usuário final

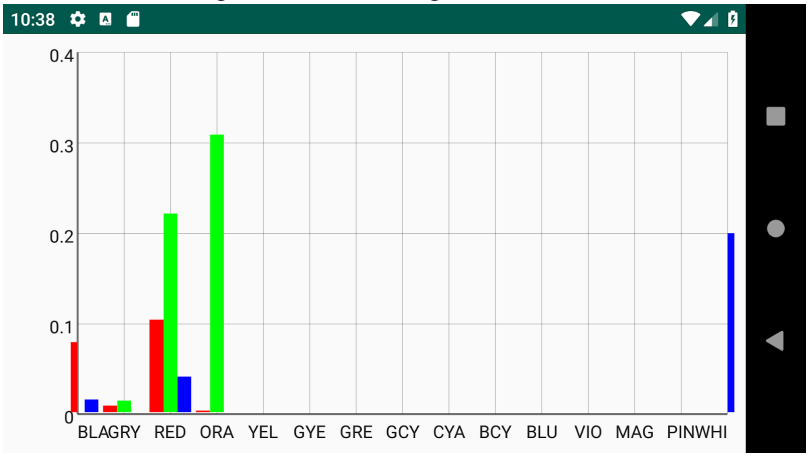



Fonte: elaborada pelo autor

### 4.2.2 Gráfico de cores

Após a geração do *MCI*, o mesmo é usado para construir um gráfico de barras que descreve a proporção de cada cor presente na *RDI*. Esse gráfico é exibido através do único componente presente nessa tela, um *GraphView*. Na versão para usuário final, essa tela não existe, e o fluxo segue para a próxima tela. A figura 18 mostra uma captura dessa tela, correspondente à classificação da *RDI* feita na figura 16, utilizando o emulador do SO *Android*.

Figura 18 – Tela de gráfico de cores




Fonte: elaborada pelo autor

Esse gráfico possui 15 grupos, cada um representando uma cor, e identificado por um trio de letras maiúsculas, derivadas de seu nome em inglês. Com exceção dos grupos *BLA*, *GRY* e *WHI*, cada grupo possui três categorias diferentes que representam a tonalidade daquela cor. Essas categorias são identificadas pela cor da barra, sendo vermelho para tonalidade escura, verde para tonalidade clara e azul para tonalidade saturada. A relação dos grupos, com a cor que ele representa e suas categorias estão descritos no quadro 2.

Quadro 2 – Relação de cor e categoria do gráfico

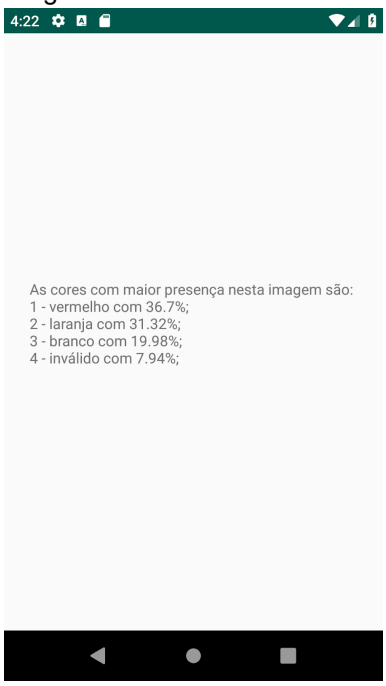
| **IDENTIFICAÇÃO** | **VERMELHO** | **VERDE** | **AZUL** |
|---|---|---|---|
| *BLA* | cor inválida | - | normal |
| *GRY* | escuro | claro | - |
| *RED* | escuro | claro | normal |
| *ORA* | escuro | claro | normal |
| *YEL* | escuro | claro | normal |
| *GYE* | escuro | claro | normal |
| *GRE* | escuro | claro | normal |
| *GCY* | escuro | claro | normal |
| *CYA* | escuro | claro | normal |
| *BCY* | escuro | claro | normal |
| *BLU* | escuro | claro | normal |
| *VIO* | escuro | claro | normal |
| *MAG* | escuro | claro | normal |
| *PIN* | escuro | claro | normal |
| *WHI* | - | - | normal |

Fonte: elaborada pelo autor

### 4.2.3 Descrição do resultado

Nessa tela, o *MCI* (que foi previamente ordenado) é usado para construir um texto que descreve as cores e suas quantidades, ditando as cores presentes na *RDI* com suas respectivas proporções em ordem decrescente e que tenham mais que 4.5% de proporção. Esse texto é construído de tal maneira a possibilitar a leitura correta pelo *TalkBack*. A imagem da figura 19 mostra uma descrição.

Figura 19 – Tela de resultado




Fonte: elaborada pelo autor

## 4.3 RESULTADOS OBTIDOS E DISCUSSÃO

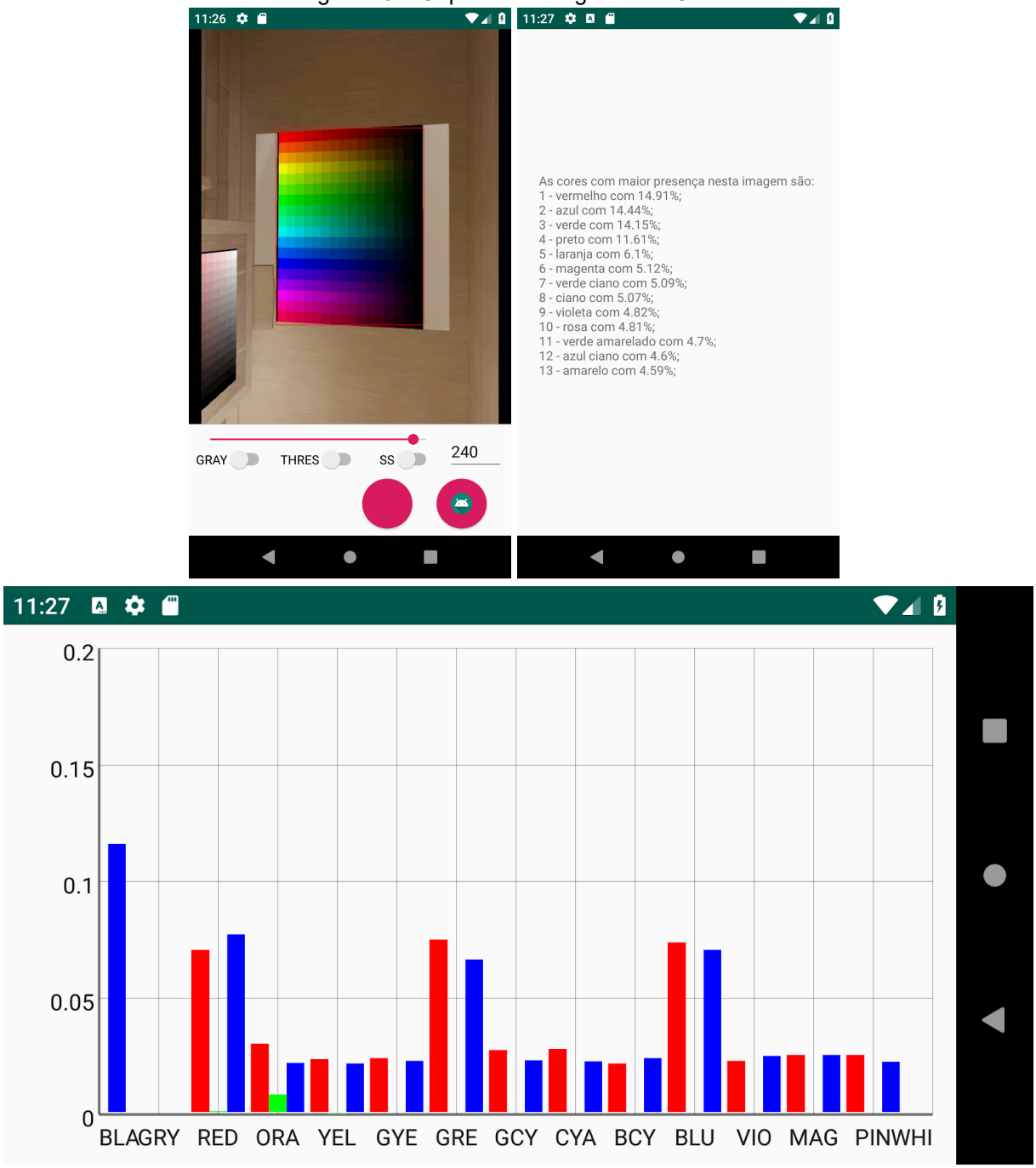
Com o objetivo de aferir a validade, o desempenho e as limitações tanto dos algoritmos quanto do aplicativo como um todo, diversos testes foram feitos considerando uma série de condições. Esses testes ocorreram executando o aplicativo em dois ambientes distintos: no emulador de dispositivo *Android Emulator* e num *smartphone Xiaomi Mi A2*.

### 4.3.1 *Android Emulator*

O *Android Emulator* é um programa emulador de dispositivo com o SO *Android* que vem no SDK do SO *Android*. Nele é possível instalar e executar aplicativos para o SO *Android*, para fins de teste, diretamente do computador onde está sendo realizado o desenvolvimento (GOOGLE, [s.d.]).

Nesse emulador, há um ambiente simulado de teste de câmera, no qual se insere uma imagem que se deseja usar e ela aparecerá no ambiente de teste da câmera. Por conta dessa característica, os testes para validar o funcionamento do algoritmo classificador de cor foram realizados nesse emulador. Para fazer a validação, foram usadas duas imagens: uma com *S* fixo no valor máximo, variando *H* nas colunas e *V* nas linhas; outra com *H* fixo em 0º (vermelho), variando *S* nas colunas e *V* nas linhas. As figuras 18 e 19 representam algumas capturas de tela obtidas com o aplicativo rodando no *Android Emulator*.

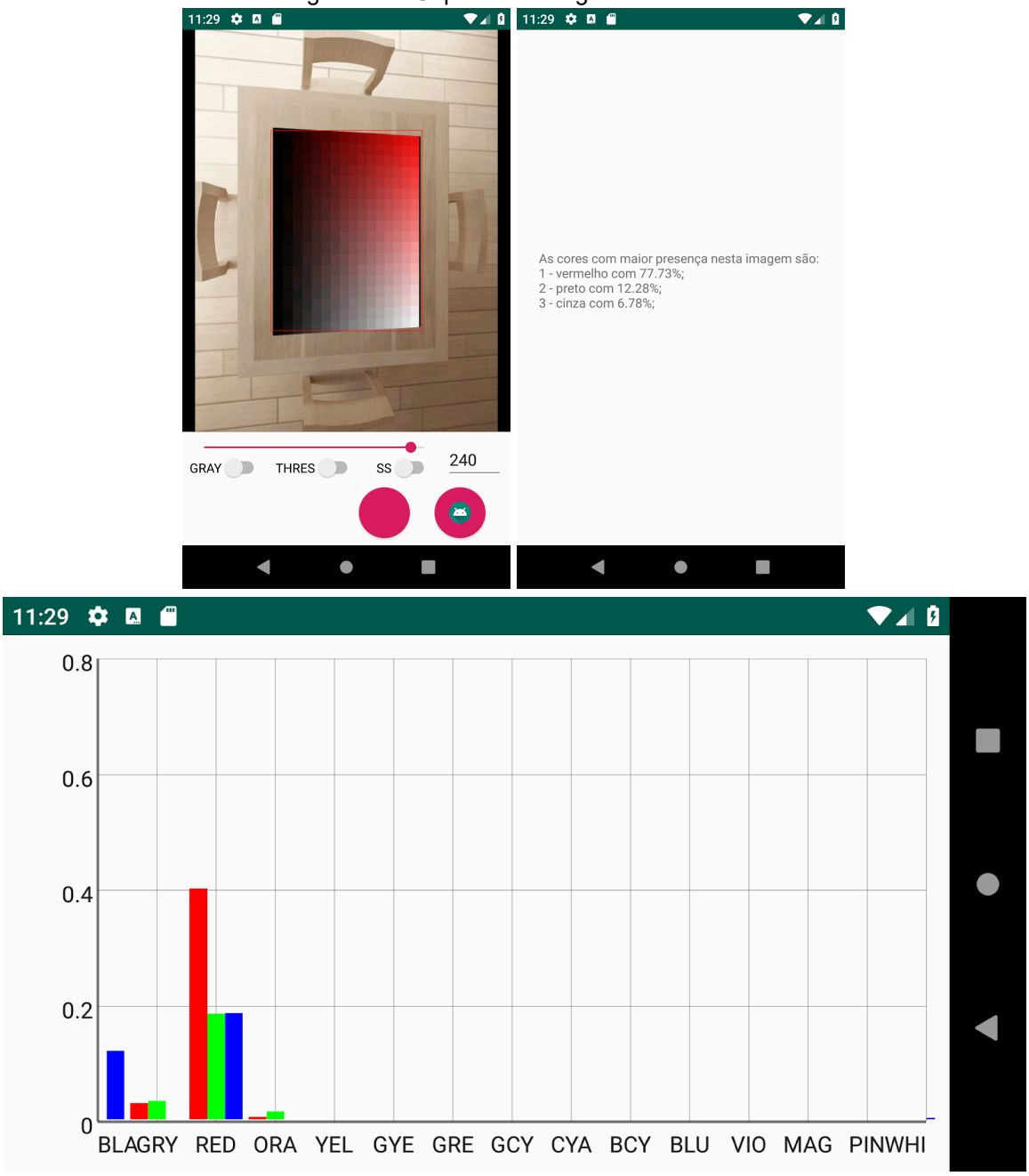
Figura 20 – Captura da imagem com *S* fixo




Fonte: elaborada pelo autor

A partir do gráfico da figura 20, percebe-se que o algoritmo afere de acordo com o esperado em relação ao matiz. As cores primárias (vermelho, verde e azul), possuem entre si aproximadamente a mesma proporção nas duas categorias, assim como as cores secundárias (todas as outras). A proporção de cor primária para cor secundária é de aproximadamente 3 para 1, validando o que fora desenvolvido. Uma quantidade de laranja claro foi contabilizada a partir da borda da *RDI*.

Figura 21 – Captura da imagem com *H* fixo




Fonte: elaborada pelo autor

Na captura da imagem com *H* fixo (figura 21), observam-se também as proporções esperadas das cores detectadas, de acordo com o algoritmo desenvolvido. Novamente observa-se uma quantidade de laranja claro proveniente das bordas da *RDI*.

### 4.3.2 *Xiaomi Mi A2*

O *smartphone* utilizado no desenvolvimento para realizar testes foi o modelo *Mi A2* da marca *Xiaomi*. Trata-se de um *smartphone* de nível médio, mas com especificação robusta, portando 4GB de memória RAM e duas câmeras traseiras de 12MP e 20MP. Mais detalhes da especificação desse aparelho encontram-se na quadro 3.

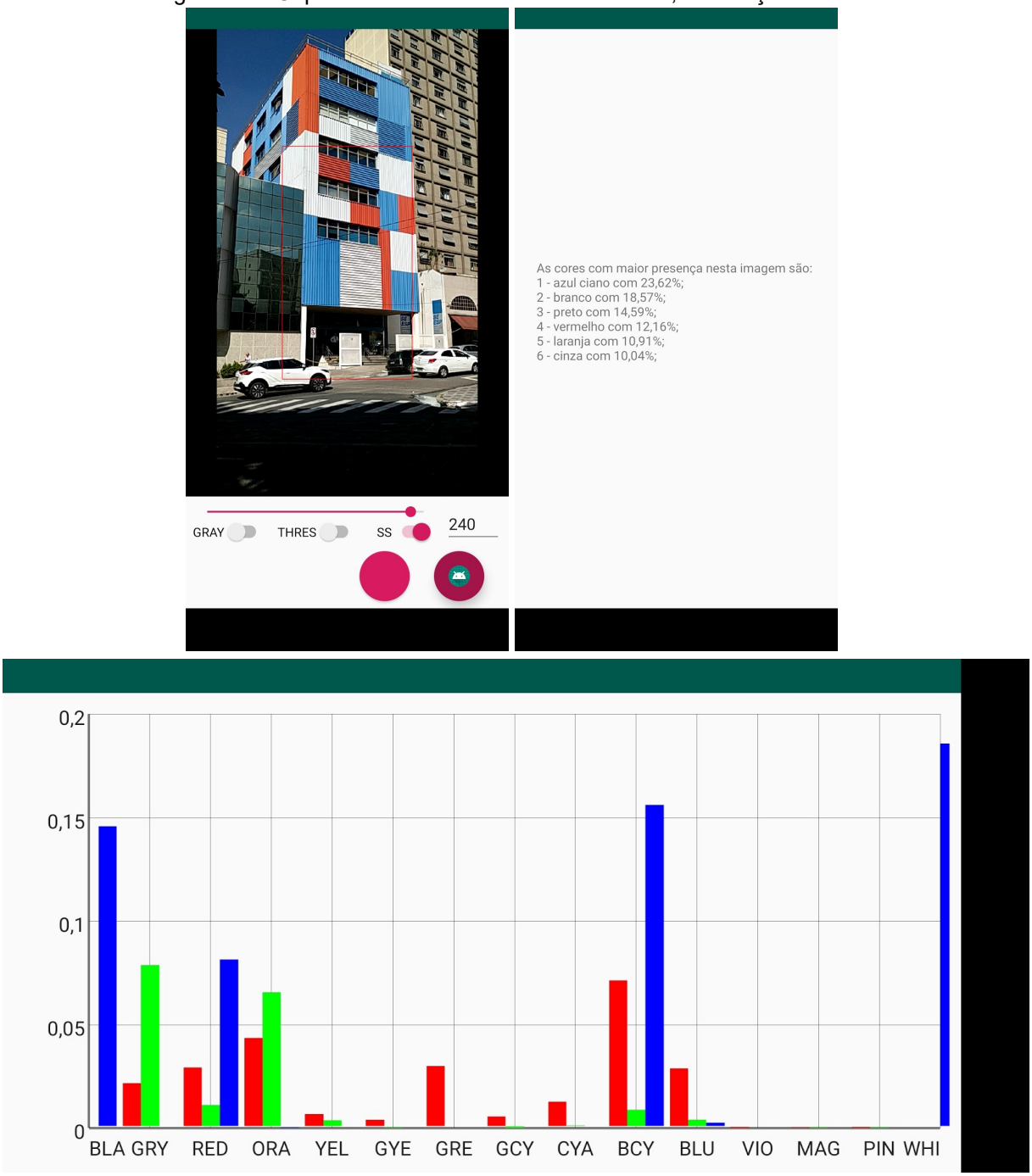
Quadro 3 – Especificações *Xiaomi Mi A2*

| **Característica** | **Descrição** |
|---|---|
| Fabricante | *Xiaomi* |
| Modelo | *Mi A2* |
| SO | *Android* 9.0 (*Pie*) versão *Android One* |
| Tela | 5,99” (15,21cm); 2160x1080px |
| Chipset | *Qualcomm Snapdragon 660* |
| Memória RAM/interna | 4GB/64GB |
| Câmera traseira | 12MP/20MP |

Fonte: adaptado de XIAOMI ([s.d.])

Nas figuras 20 a 26 estão apresentadas algumas capturas realizadas com esse aparelho, ilustrando o desempenho do aplicativo em diversas situações.

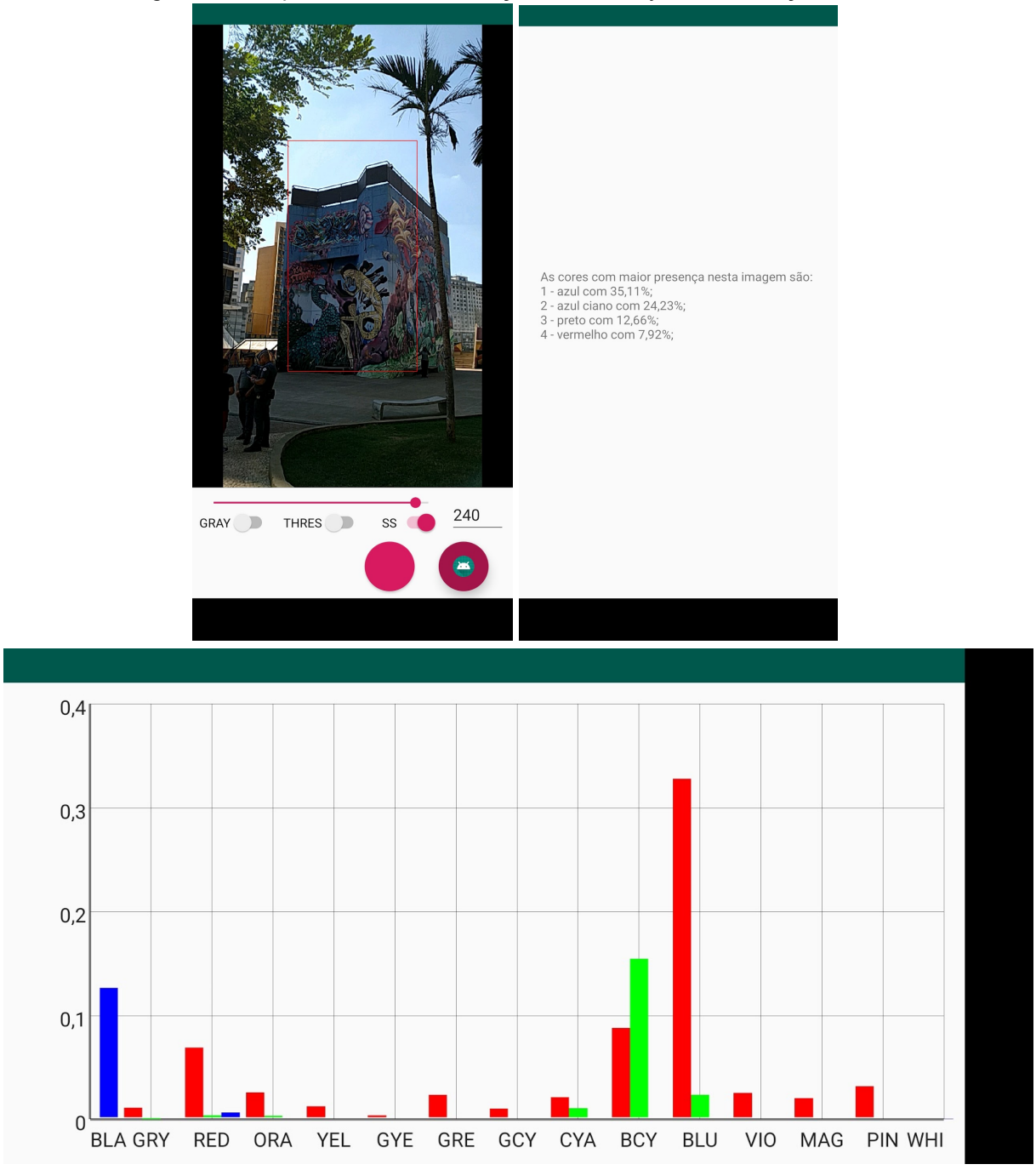
Figura 22 – Captura da fachada da sede da SEESP, iluminação natural




Fonte: elaborada pelo autor

Na figura 22, da fachada de um prédio, nota-se o branco, o azul e o vermelho; na projeção da área coberta há o preto.

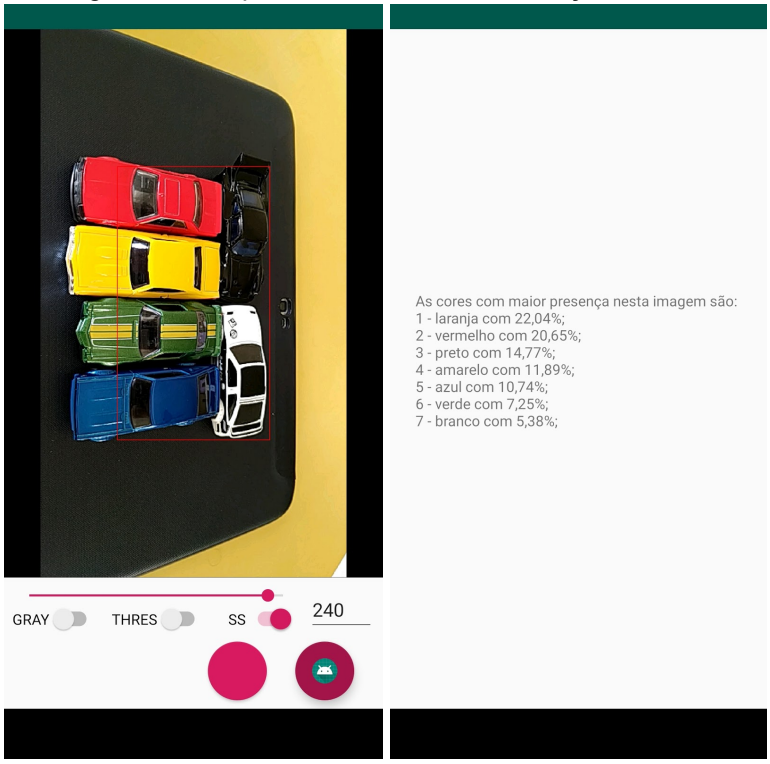
Figura 23 – Captura do mural da Praça Paulo Kobayashi, iluminação natural

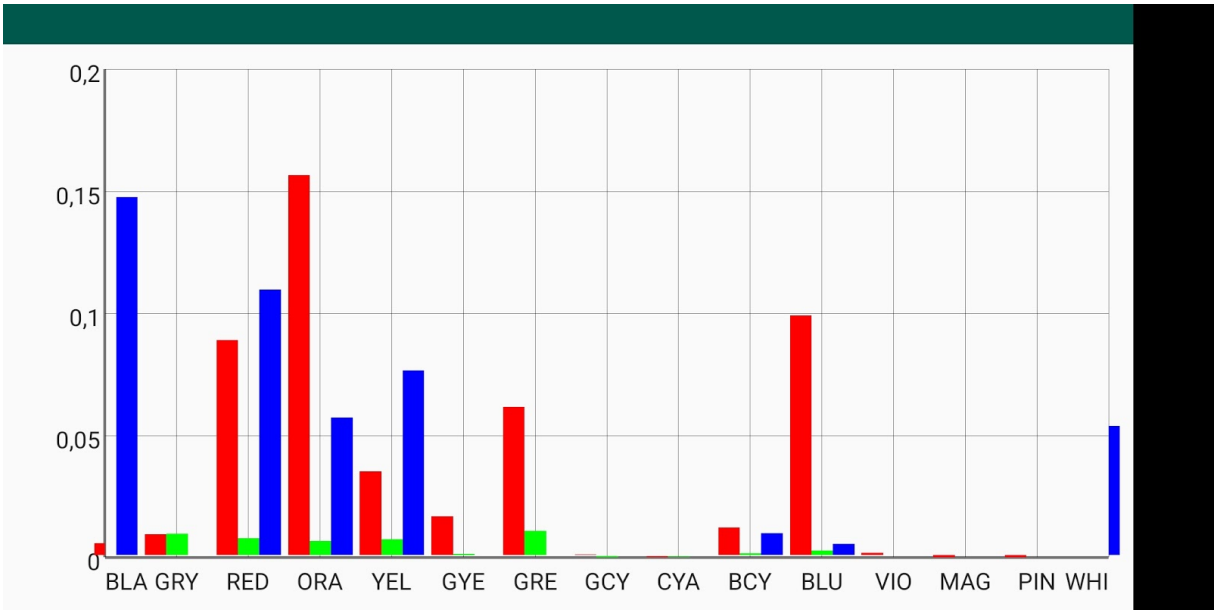



Fonte: elaborada pelo autor

Na figura 23, nota-se a predominância das cores em tonalidades escuras por conta do mural estar sombreado. Destaque para o azul escuro que cobre a maior parte do grafite do mural, o preto da grade da imagem e o azul-ciano claro do céu.

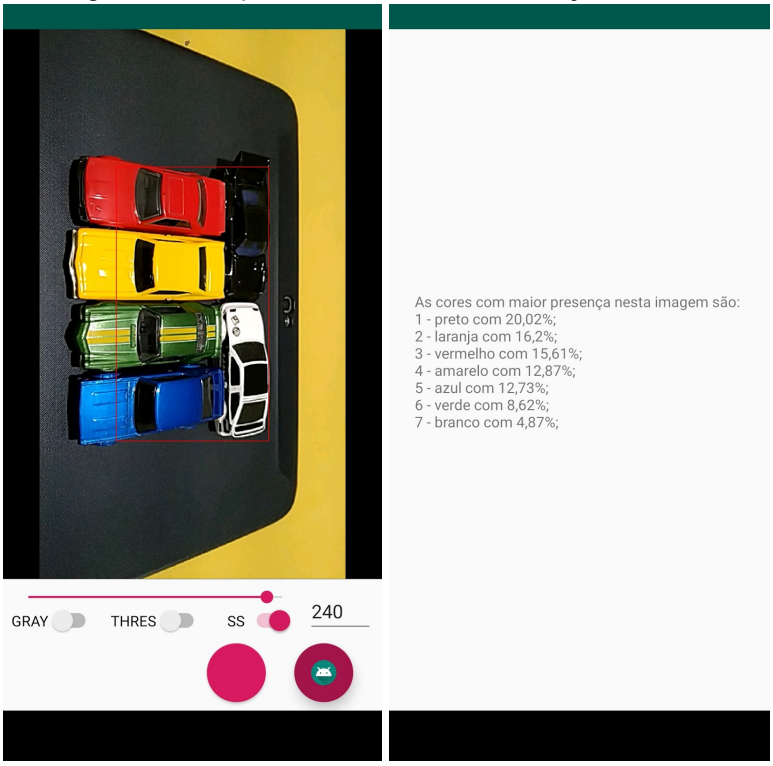
Figura 24 – Captura de miniaturas, iluminação natural

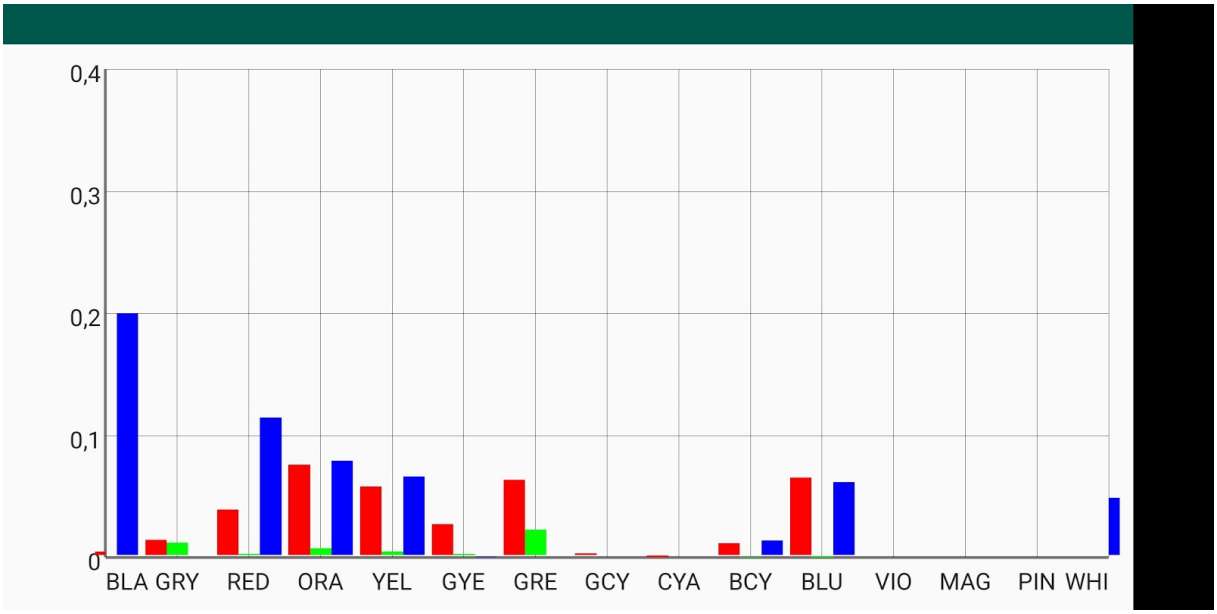



Fonte: elaborada pelo autor

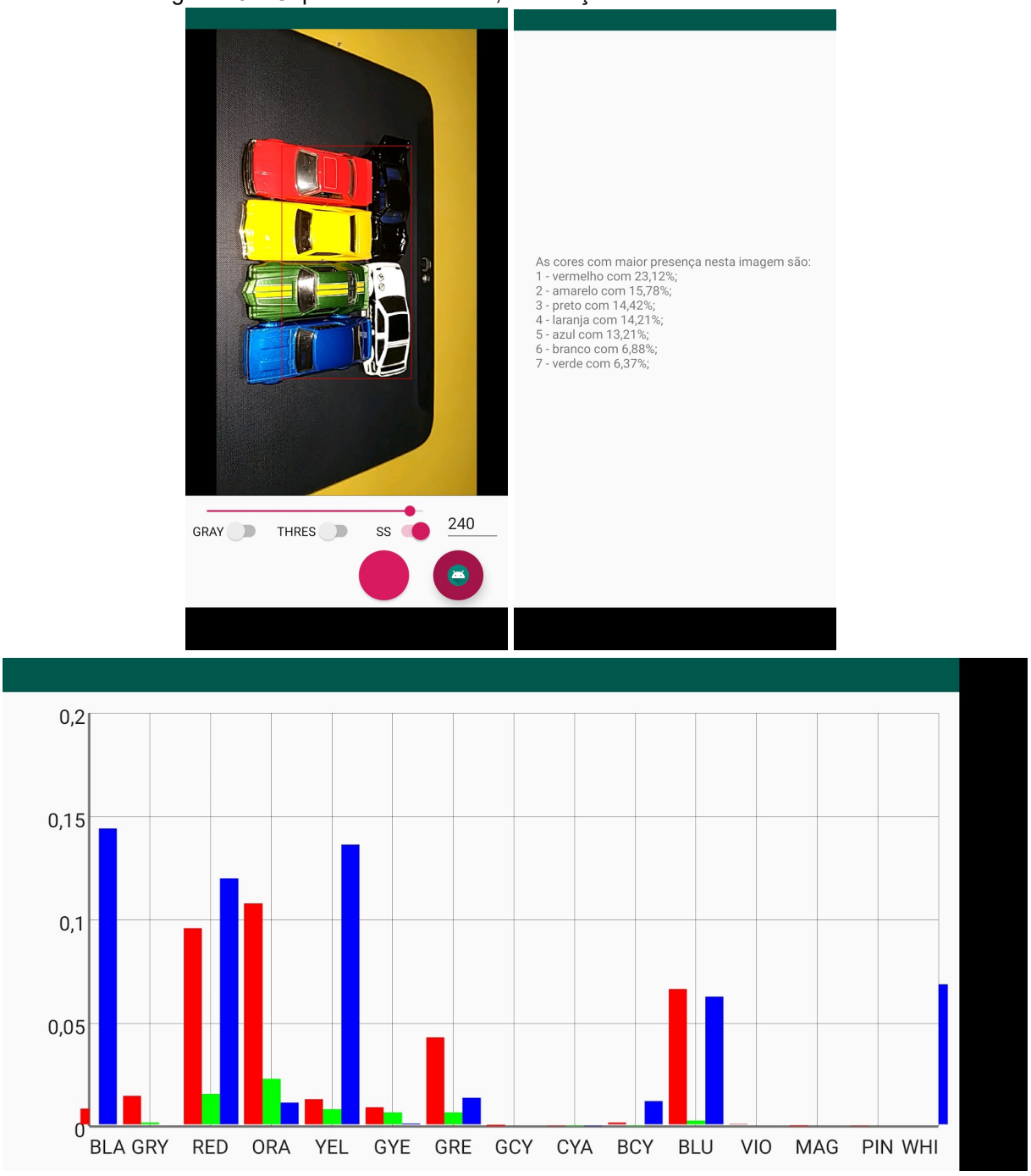
Figura 25 – Captura de miniaturas, iluminação artificial




Fonte: elaborada pelo autor

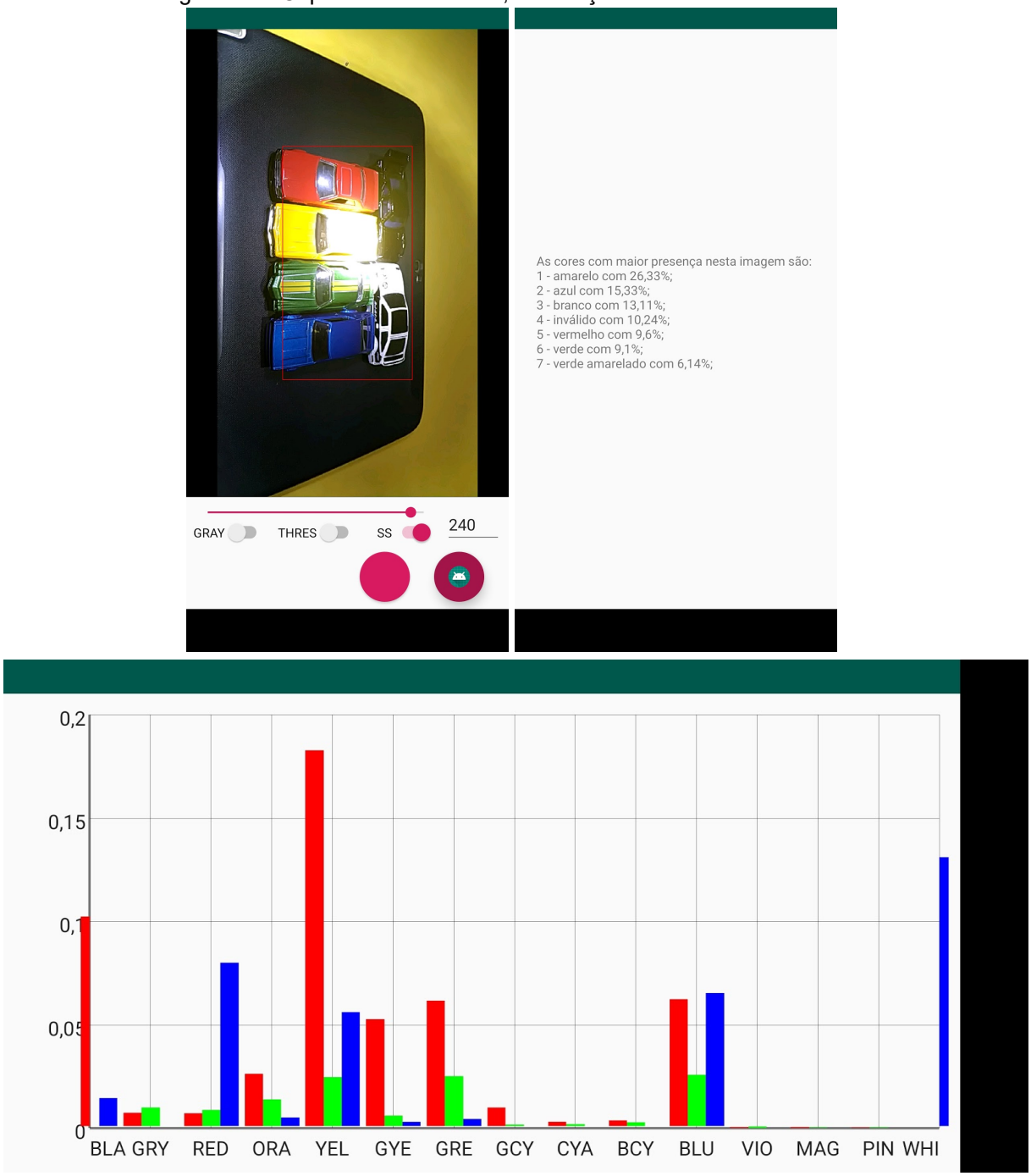
Figura 26 – Captura de miniaturas, iluminação artificial e flash acionado




Fonte: elaborada pelo autor

Nas figuras 24, 25 e 26, observa-se certa regularidade das cores detectadas, sendo que, nas figuras 24 e 26, até as proporções das cores são semelhantes, diferenciando apenas na proporção das tonalidades. Já na figura 25, não há luz refletindo nos parabrisas das miniaturas que estão à frente, isso faz com que mais preto seja detectado.

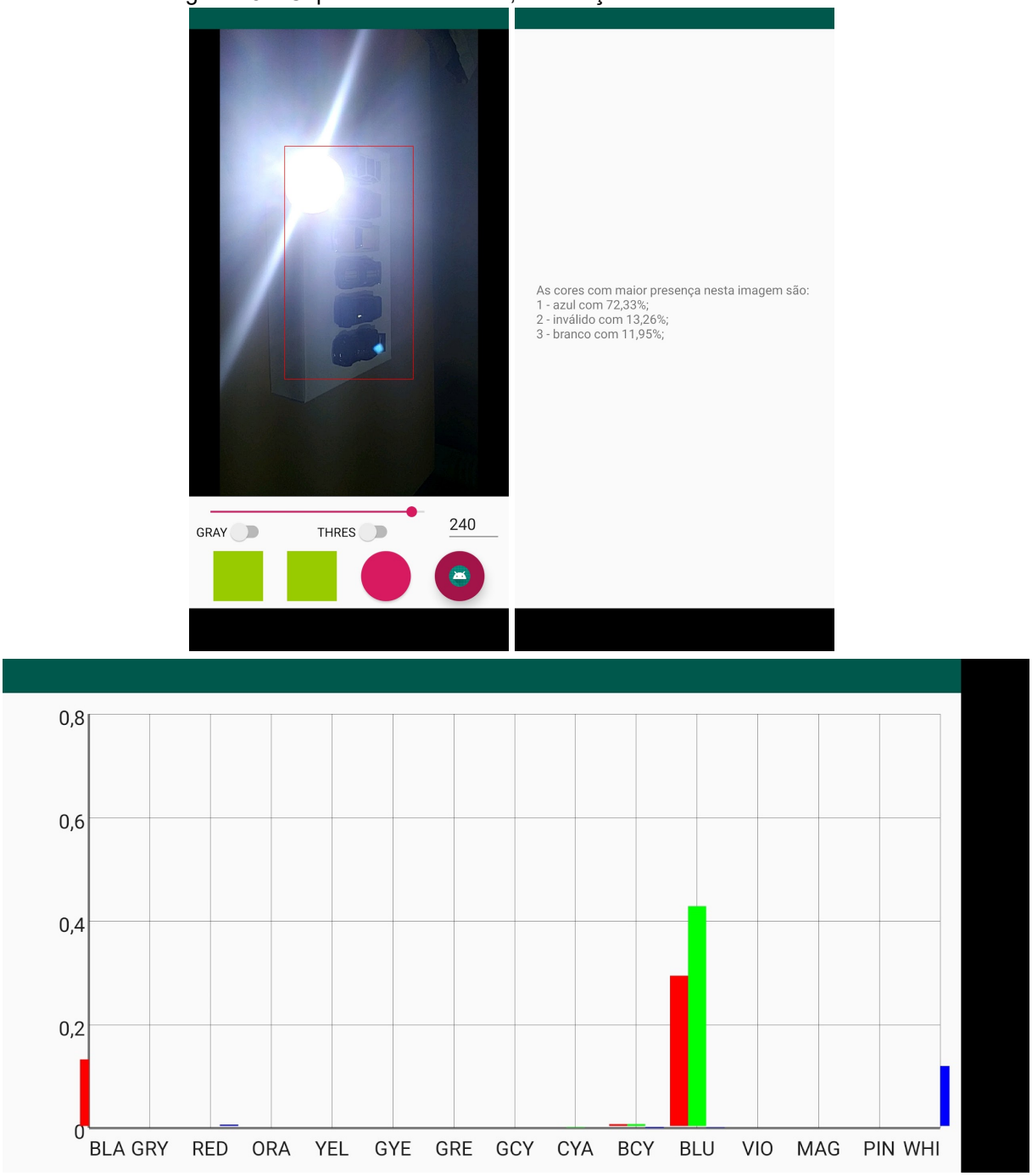
Figura 27 – Captura de miniaturas, iluminação artificial e reflexo de luz




Fonte: elaborada pelo autor

Na figura 27 percebe-se o impacto da presença de uma fonte de luz ou de seu reflexo na detecção. Cerca de 10% dos píxeis foram invalidados e houve uma detecção maior de amarelo e branco.

Figura 28 – Captura de miniaturas, iluminação artificial e fonte de luz




Fonte: elaborada pelo autor

A captura feita com uma fonte de luz apontando diretamente para a câmera, como na figura 28, acaba por degradar a imagem e, por consequência, o resultado da análise.

### 4.3.3 Discussão

Ao comparar as imagens das capturas com os seus respectivos gráficos, verifica-se que o algoritmo desenvolvido consegue fazer a classificação e contagem das cores da maneira como foi planejada. Além disso, a interface da aplicação para o usuário final também funciona como planejado, com os itens sendo corretamente focalizados e descritos pelo *TalkBack*, e ocorrendo as vibrações para indicar a presença de luz intensa na imagem a ser analisada.

Contudo, sendo a cor definida no aplicativo como uma tonalidade de um matiz, e os matizes sendo definidos levando-se em conta apenas critérios matemáticos, foram observados os seguintes fatos: o rol de cores definidas no aplicativo ora possui cores não usuais (como o azul ciano), ora não possui cores relevantes (como o marrom, por exemplo); certo píxel pode ter seu matiz classificado erroneamente (por exemplo, trocando amarelo por laranja e vice-versa) se o *H* medido estiver próximo de um *HUT*.

Como não foi encontrado nenhum outro produto ou trabalho, científico ou comercial, que realize as mesmas tarefas de contabilizar as cores presentes numa imagem, não foi possível fazer uma análise mais detalhada entre sistemas. Esse fato faz com que se destaque a inovação trazida por este projeto em relação às outras soluções disponíveis atualmente, e abre espaço para que se estudem soluções para outros problemas mediante desdobramentos desta.

# 5 CONSIDERAÇÕES FINAIS

## 5.1 CONCLUSÕES

À luz do aumento do número de usuários de *smartphones* e das frequentes melhorias pelas quais passam os *smartphones* e seus ambientes de desenvolvimento ao longo do tempo, este trabalho foi desenvolvido visando prover Tecnologia Assistiva através de uma solução que pudesse descrever quantitativamente as cores de um objeto. Além de se tratar de algo ainda não existente no mercado nacional e com interface em língua portuguesa do Brasil, esse trabalho possui um grande potencial de utilidade prática para as pessoas com deficiência visual.

Inicialmente foi feito um levantamento dos aplicativos que também utilizam imagens obtidas pela câmera do smartphone para extrair alguma informação sobre cor, e como esses aplicativos fazem a descrição falada dessa informação. O principal resultado desse levantamento foi a decisão sobre qual tipo de informação o aplicativo desenvolvido proporcionaria ao usuário: a descrição apresentada não seria somente sobre a cor predominante do objeto em foco, mas sim um mapa quantitativo da proporção de cores existentes na imagem analisada. Esta característica consiste numa inovação introduzida por esse trabalho no rol de aplicativos disponíveis atualmente .

Foi definido que o aplicativo seria desenvolvido para dispositivos com SO *Android* devido à experiência e conhecimento do autor nessa área, logo, a linguagem de programação usada foi o *Java*. Escolheu-se a biblioteca OpenCV dada a diversidade de técnicas de processamento digital de imagem implementadas, além de uma grande base de usuários e conhecimento.

A partir do estudo sobre processamento digital de imagem, combinando os conhecimentos de imagem digital e modelos de cor e as técnicas de conversão de espaço de cor, limiarização e detecção de borda, foram desenvolvidos: um algoritmo para contabilizar a área com luz presente numa imagem; e um algoritmo para contabilizar as cores (de uma lista de cores predefinida) presentes numa imagem.

Combinando ambos algoritmos, cujas técnicas foram providas pela biblioteca OpenCV, ao conjunto de soluções de desenvolvimento de aplicativos do SDK do SO *Android*, tendo em mente a integração com o leitor de tela *TalkBack*, foi então desenvolvido o aplicativo objeto deste trabalho.

Testes foram feitos num ambiente virtual e num ambiente real e, em ambos os casos, o aplicativo funcionou conforme esperado, tanto na questão da classificação das cores quanto na questão da integração com o *TalkBack*.

As principais áreas de estudo que estiveram envolvidas no desenvolvimento desse projeto foram: teoria das cores; processamento digital de imagens; linguagem de programação *Java*; experiência do usuário; tecnologia assistiva.

## 5.2 TRABALHOS E MELHORIAS FUTURAS

Através de vários testes realizados com o aplicativo, foram identificadas algumas limitações que justificam melhorias e trabalhos futuros para seu aperfeiçoamento.

Em relação aos algoritmos, há a necessidade de se introduzir aspectos fisiológicos, psicofísicos e culturais à definição das cores do aplicativo, de maneira que essa definição seja de fato mais prática e clara para o usuário. Além disso, faz-se necessário elaborar uma solução melhor para a detecção de fonte de luz na captura, incluindo algum método de detecção de superfícies especulares e/ou um ajuste do limiar de binarização.

No âmbito do aplicativo em si, há a necessidade de se utilizar alguns conceitos de engenharia de *software*, principalmente em relação ao padrão de arquitetura de *software*, de modo a facilitar a implementação de melhorias futuras e a manutenção do código. Além disso, também há a necessidade de se implementar um tutorial para ser executado no primeiro uso.

É possível também, com a estrutura desenvolvida, agregar mais funcionalidades relacionadas ao aplicativo atual, como por exemplo: descritor de cédulas de dinheiro; descritor de vestimenta; reconhecedor ótico de caracteres, dentre outros.

# 6 CRONOGRAMA DE ATIVIDADES

No quadro 4 está representado o cronograma que foi seguido no decorrer do desenvolvimento do projeto de pesquisa.

Quadro 4 – Cronograma de atividades

| **Etapas do projeto** | **Quadrimestres** | | |
|---|---|---|---|
| | **2q2018** | **3q2018** | **1q2019** |
| Revisão bibliográfica | X | X | X |
| Testes com aplicativos disponíveis no mercado para levantamento das características | X | | |
| Estudo da linguagem de programação *Java* | X | X | |
| Estudo do desenvolvimento de aplicativos para o SO *Android* | X | X | |
| Estudo do desenvolvimento de interfaces gráficas em aplicativos móveis para pessoas com deficiência visual (com uso do *Google TalkBack*) | X | X | |
| Codificação | | X | X |
| Testes e correções da aplicação | | X | X |
| Redação do relatório técnico | | X | X |

Fonte: elaborada pelo autor

# REFERÊNCIAS


APPLE INC. **VoiceOver**. Acessibilidade. Disponivel em: <https://support.apple.com/kb/sp706?locale=pt_BR>. Acesso em: 28 jul. 2018.

AUIRE TECNOLOGIAS ACESSÍVEIS. **Auire Prisma**. Auire Tecnologias Acessíveis. Disponivel em: <http://www.auire.com.br/prisma/>. Acesso em: 02 abr. 2016.

BERSCH, Rita. **Introdução à Tecnologia Assistiva**. Assistiva - Tecnologia e Educação, 2013. Disponivel em: <http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf>. Acesso em: 02 abr. 2016.

BRASIL. Ministério da Ciência, Tecnologia e Inovação. **Sobre o Catálogo**. Catálogo Nacional de Produtos de Tecnologia Assistiva. Disponivel em: <http://assistiva.mct.gov.br/sobre-o-catalogo>. Acesso em: 02 abr. 2016.

BRIGGS, David. **The Dimensions of Colour, lightness, value, tone**. The Dimensions of Colour. Disponível em: <http://www.huevaluechroma.com/013.php#hls>. Acesso em: 4 mar. 2019.

BRIGGS, David. **The Dimensions of Colour, saturation**. The Dimensions of Colour. Disponível em: <http://www.huevaluechroma.com/017.php>. Acesso em: 5 jan. 2019.

HEARN, Donald; BAKER, M. Pauline. Color Models and Color Application. In: HEARN, Donald; BAKER, M. Pauline. **Computer Graphics, C Version.** 2. ed. Upper Saddle River: Prentice Hall, 1996. Cap. 15. p. 564-582.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE), BRASIL. Ministério do Planejamento, Orçamento e Gestão. Instituto Brasileiro de Geografia e Estatística . **Censo demográfico 2010: Características gerais da população, religião e pessoas com deficiência**. Biblioteca do IBGE, 2010. ISSN 1676-4935. Disponivel em: <http://servicodados.ibge.gov.br/Download/Download.ashx?http=1&u=biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf>. Acesso em: 02 abr. 2016.

BRASIL. Secretaria de Direitos Humanos da Presidência da República. Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência - SNPD. **Acessibilidade**. Site da Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência. Disponivel em: <http://www.pessoacomdeficiencia.gov.br/app/acessibilidade-0>. Acesso em: 02 abr. 2016.

BRADSKI, Gary. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BRADSKI, Gary; KAEHLER, Adrian. Image Processing. In: BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV.** Sebastopol: O'Reilly Media, 2008. Cap. 5. p. 109-143.

BRILL, Jonathan. **Jonathan Brill's answer to How is technology affecting the lives of people with disabilities?**. Quora, 2016. Disponível em: <https://www.quora.com/How-is-technology-affecting-the-lives-of-people-with-disabilities/answer/Jonathan-Brill>. Acesso em: 02 abr. 2016.

BROWN, Blain. **Cinematography: Theory and Practice: Image Making for Cinematographers and Directors**. [s.l.]: CRC Press, 2016.

GEHRING, Jonas. **Bar Chart**. GraphView. Disponível em: <http://www.android-graphview.org/bar-chart/>. Acesso em: 5 mar. 2019.

GERBER, Adam; CRAIG, Clifton. Introducing Android Studio. In: GERBER, Adam; CRAIG, Clifton. **Learn Android Studio**. [S.l.]: Apress, 2015. cap. 1, p. 1-15.

GOOGLE. **Android Studio Overview**. Android Developers. Disponivel em: <http://developer.android.com/intl/pt-br/tools/studio/index.html>. Acesso em: 02 abr. 2016.

GOOGLE. **Bitmap**. Android Developers. Disponivel em: <http://developer.android.com/intl/pt-br/reference/android/graphics/Bitmap.html>. Acesso em: 02 abr. 2016.

GOOGLE. **Color**. Android Developers. Disponivel em: <http://developer.android.com/intl/pt-br/reference/android/graphics/Color.html>. Acesso em: 02 abr. 2016.

GOOGLE. **Color Detector**. Google Play. Disponivel em: <https://play.google.com/store/apps/details?id=com.keesadens.colordetector>. Acesso em: 24 jun. 2018.

GOOGLE. **Controlling the Camera**. Android Developers. Disponivel em: <http://developer.android.com/intl/pt-br/training/camera/cameradirect.html>. Acesso em: 02 abr. 2016.

GOOGLE. **Detector de Cor**. Google Play. Disponivel em: <https://play.google.com/store/apps/details?id=com.rameltec.falacor>. Acesso em: 24 jun. 2018.

GOOGLE. **Executar aplicativos no Android Emulator**. Android Developers. Disponível em: <https://developer.android.com/studio/run/emulator>. Acesso em: 5 mar. 2019.

GOOGLE. **Introdução ao Android**. Android Developers. Disponivel em: <http://developer.android.com/intl/pt-br/guide/index.html>. Acesso em: 02 abr. 2016.

GOOGLE. **Primeiros passos no Android com o TalkBack**. Ajuda do Android Accessibility. Disponivel em: <https://support.google.com/accessibility/android/answer/6283677?hl=pt-BR&ref_topic=3529932>. Acesso em: 02 abr. 2016.

CONTRIBUIDORES DA WIKIPÉDIA. HSL and HSV. In: **WIKIPEDIA, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=890683083>. Acesso em: 5 mar. 2019.

GONZALES, Rafael C.; WOODS, Richard E. In: GONZALES, Rafael C.; WOODS, Richard E. **Digital Image Processing**. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2002. Cap. 1. p. 1-33.

HUANG, Thomas Shi-tao. **Computer Vision: Evolution And Promise**. In: CERN SCHOOL OF COMPUTING, 19., 1996, Egmond Aan Zee. Proceedings... . Genebra: Cern, 1996. p. 21 - 25. Disponível em: <http://cds.cern.ch/record/400313/files/p21.pdf>. Acesso em: 13 jan. 2019.

INTEL CORPORATION. **Intel Integrated Performance Primitives Developer Reference:** Volume 2: Image Processing. 2018. Disponível em: <https://software.intel.com/sites/default/files/ippi_0.pdf>. Acesso em: 30 nov. 2018.

INTERNATIONAL TELECOMMUNICATION UNION - RADIOCOMMUNICATION SECTOR (ITU-R) (Org.). **ITU-R Recommendation BT601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios**. 2011. Disponível em: <https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf>. Acesso em: 30 nov. 2018.

JACK, Keith. Color Spaces. In: JACK, Keith. **Video demystified: a handbook for the digital engineer**. 4th ed. Amsterdam; Boston: Elsevier, 2005. Cap. 3. p. 15-34.

JAIN, Ramesh; KASTURI, Rangachar; SCHUNCK, Brian G. Binary Image Processing. In: JAIN, Ramesh; KASTURI, Rangachar; SCHUNCK, Brian G. **Machine vision**. New York: McGraw-Hill, 1995. Cap. 2. p. 25-72.

LEAL, Nelson Glauber de Vasconcelos. Multimídia: Câmera. In: LEAL, Nelson Glauber de Vasconcelos. **Dominando o Android**. 2. ed. São Paulo: Novatec Editora Ltda, 2015. cap. 19, p. 632-636.

MICROSOFT. **Seeing AI**. Seeing AI. Disponivel em: <https://www.microsoft.com/en-us/seeing-ai>. Acesso em: 28 jul. 2018.

MOBIALIA. **Color Detector**. Mobialia. Disponivel em: <https://www.mobialia.com/apps/color-detector/>. Acesso em: 24 jun. 2018.

MORRIS, J.; MUELLER, J.. Blind and Deaf Consumer Preferences for Android and iOS Smartphones. In: LANGDON, P. M. et al. **Inclusive Designing Joining Usability, Accessibility, and Inclusion**, [s.l.], p.69-79, 2014. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-05095-9_7

NOVARTIS PHARMA AG. **ViaOpta Daily**. ViaOpta. Disponivel em: <http://viaopta-apps.com/ViaOptaDaily.html>. Acesso em: 02 abr. 2016.

SHIRLEY, Peter; MARSCHNER, Steve. Raster Images. In: SHIRLEY, Peter et al. **Fundamentals of Computer Graphics.** 3. ed. Natlick: Crc Press, 2009. Cap. 3. p. 53-68.

SMITH, Alvy Ray. Color gamut transform pairs. **Acm Siggraph Computer Graphics**, [s.l.], v. 12, n. 3, p.12-19, 23 ago. 1978. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/965139.807361.

SZELISKI, Richard. Image processing. In: SZELISKI, Richard. **Computer Vision**. London: Springer London, 2011. Cap. 3. p. 87-180.

VILLEGAS, Alex. Como Funciona a Captura Digital. In: VILLEGAS, Alex. **O Controle da Cor:** Gerenciamento de Cores para Fotógrafos. Balneário Camboriú: Photos, 2009. Cap. 2. p. 29-46.

WIRELESS RERC (Estados Unidos da América). Georgia Institute Of Technology. **SUNspot – Wireless Device Operating Systems of People with Disabilities.** 2016. Disponível em: <http://www.wirelessrerc.gatech.edu/sites/default/files/publications/sunspot_2016-02_wireless_device_operating_systems_all_disabilities_2016-12-18.pdf>. Acesso em: 15 ago. 2018.

XIAOMI. **Mi A2**. Mi Global Home. Disponível em: <https://www.mi.com/global/mi-a2/specs/>. Acesso em: 5 mar. 2019.