UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO: CIÊNCIAS DA COMPUTAÇÃO

BENE LEMUEL DANTAS GONDIM

DESENVOLVIMENTO DE UM SOFTWARE PARA CORREÇÃO DE AVALIAÇÕES OBJETIVAS

MOSSORÓ - RN
2014

BENE LEMUEL DANTAS GONDIM

DESENVOLVIMENTO DE UM SOFTWARE PARA CORREÇÃO DE AVALIAÇÕES OBJETIVAS

Trabalho de conclusão de curso apresentado a Universidade Federal Rural do Semi-Árido-UFERSA, Departamento de Ciências Exatas e Naturais, como parte dos requisitos para a obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Marcelo Roberto Bastos Guerra Vale –UFERSA

Mossoró - RN
2014

O conteúdo desta obra é de inteira responsabilidade de seus autores

Dados Internacionais de Catalogação na Publicação (CIP)

Biblioteca Central Orlando Teixeira (BCOT)

Setor de Informação e Referência

G586d Gondim, Bene Lemuel Dantas.

Desenvolvimento de um software para correção de avaliação objetivas / Bene Lemuel Dantas Gondim. -- Mossoró, 2014.

105f.: il.

Orientador: Prof. Dr. Marcelo Roberto Bastos Guerra Vale.

Monografia (Graduação em Ciência da Computação) – Universidade Federal Rural do Semi-Árido. Pró-Reitoria de Graduação.

1. Desenvolvimento de software . 2 . Dispositivo móveis. 3. Java. 4. MATLAB. I. Título.

RN/UFERSA/BCOT /685-14 CDD: 005.12

Bibliotecária: Vanessa de Oliveira Pessoa

CRB-15/453

BENE LEMUEL DANTAS GONDIM

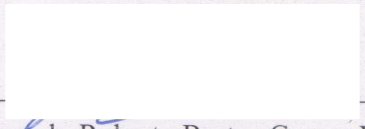
DESENVOLVIMENTO DE UM SOFTWARE PARA CORREÇÃO DE AVALIAÇÕES OBJETIVAS

Trabalho de conclusão de curso apresentado a Universidade Federal Rural do Semi-Árido-UFERSA, Departamento de Ciências Exatas e Naturais, como parte dos requisitos para a obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Marcelo Roberto Bastos Guerra Vale – UFERSA

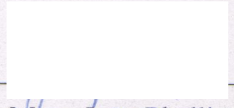
APROVADO EM: 06 / 08 / 2014

BANCA EXAMINADORA



Prof. Dr Marcelo Roberto Bastos Guerra Vale – UFERSA

Orientador



M.sc. João Phellipe de Freitas Pinto

Primeiro Membro

Prof. Dênis Freire Lopes Nunes - UFERSA

Segundo Membro

DEDICATÓRIA

Dedico este trabalho de conclusão de curso a Deus por todo o que me proporciona; aos meus pais, Maria Dalva e Antônio Aldemir pelo incentivo e amor; à minha namorada pelo amor, paciência e grande ajuda.

AGRADECIMENTO

A Deus por todos os dias de minha vida e por tudo que nos concede, como bem canta a letra de uma música “Por tudo o que tens feito, por tudo o que vais fazer, por tuas promessas e tudo que És eu quero te agradecer com todo o meu ser” OBRIGADO MEU DEUS!

Aos meus pais, Maria Dalva e Antônio Aldemir que me incentivaram e apoiaram incondicionalmente, se não fosse por vocês esse momento não seria possível, o meu muito obrigado.

À minha namorada, Dinara Fernanda, pelo grande incentivo, conselhos e imenso amor com que sempre me estimulou a lutar pelas coisas que considerar importante; a você, o meu eterno obrigado.

Aos professores pelas experiências e conhecimentos que compartilhamos vamos levá-los pra sempre em nossas vidas profissionais e pessoais.

Ao meu orientador Marcelo Guerra pela contribuição no transcorrer da elaboração deste trabalho de conclusão do curso.

Aos professores João Phellipe e Dênis Freire, por aceitarem o convite para participar da banca do meu trabalho.

Aos amigos e amigas pelo incentivo e carinho de todos os momentos.

A todos que de perto ou de longe incentivaram e torceram pelo meu sucesso o meu imenso OBRIGADO!

RESUMO

Este trabalho propõe uma solução tecnológica de correção de provas com questões objetivas. A princípio o trabalho foi desenvolvido utilizando a ferramenta MATLAB e em seguida foi desenvolvida uma versão do software para dispositivos móveis, especificamente para dispositivos que rodem a plataforma Android. O software trabalhará com elementos de obtenção de imagens e processamento digital de imagens dos gabaritos impressos. Contempla ainda a geração de relatórios que fornecerá informações estatísticas sobre as aplicações das provas. O trabalho discorrerá todo o processo de fundamentação teórica onde é contemplada desde a apreciação pedagógica sobre o modelo avaliativo à toda a tecnologia utilizada durante o desenvolvimento da solução.

Palavras chave: Correção de provas, solução tecnológica, MATLAB, Java, dispositivos móveis, Android

ABSTRACT

This work proposes a technological solution to revision of objective tests. At first, the work was developed using MATLAB and then was developed a version of software for mobile devices, specifically for devices that run the Android platform. The software will work with elements of imaging and digital image processing of printed templates. Also includes the generation of reports that provide statistical information about the applications of tests. The work will talk the whole process of theoretical foundation, which contemplates since the pedagogical assessment of the evaluation, as the technology used during the development of the model solution.

Keywords: Test revision, technological solution, MATLAB, Java, Mobile, Android

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Em todos os seguimentos é utilizado a aplicação de avaliações, pois sua necessidade e pratica estão presentes nos mais variados segmentos, sejam nas empresas, em órgãos públicos, em ambientes escolares; sendo utilizada nas mais variadas situações, como para ingressar em um concurso, ou em um emprego.

As avaliações podem ser realizadas de diversas formas, como em forma de dinâmica e de entrevista. Em especial nas avaliações diretas, que se materializam em respostas em papel, que são o foco do trabalho, diversos são os esforços para otimizar a forma de correção de tais avaliações, pois o processo manual é cansativo e enfadonho.

As provas podem ser projetadas em diversos formatos, como provas objetivas, subjetivas ou orais. O formato mais utilizado é o da prova objetiva, a qual contém questões de múltiplas escolhas que devem ser respondidas de forma única. Esse tipo de avaliação é utilizado na grande maioria dos processos seletivos realizados no Brasil, compondo parte ou totalidade da prova, como bem exemplifica o Exame Nacional de Ensino Médio (ENEM), o exame de Ordem dos Advogados do Brasil (OAB), concursos e vestibulares, bem como as provas feitas em sala de aula por professores.

As provas objetivas são mais rápidas de serem corrigidas, se comparada com as provas subjetivas, porém ainda requerem muito tempo dos avaliadores, mesmo possuindo gabaritos idênticos, pois o número de questões e/ou provas são grandes, tornando-se um processo cansativo e rotineiro devido a sua grande repetição.

Objetivando otimizar o tempo de correção das provas objetivas é que este trabalho se desenvolveu, propondo uma solução tecnológica para tal problemática, onde foi desenvolvido um software de correção para dispositivos móveis.

O software desenvolvido trabalha com elementos de obtenção de imagens e processamento digital de imagens dos gabaritos impressos. Contempla ainda a geração de relatórios que fornece informações estatísticas sobre as aplicações das provas.

O trabalho é estruturado em cinco capítulos: o primeiro contém a introdução, na qual se apresenta os aspectos mais relevantes do estudo; o segundo capítulo contém a revisão bibliográfica, onde se aborda os referenciais teóricos que embasa o trabalho; no terceiro capítulo é explanada a metodologia, como foi desenvolvido o software de correção de avaliações objetivas; o capítulo quatro relata os resultados obtidos com a realização os testes

do software, bem como as discussões de tais resultados obtidos; e no último capítulo, o cinco, é feito a conclusão do trabalho e as sugestões para o desenvolvimento de futuros trabalhos.

1.1 JUSTIFICATIVA

Diante dos argumentos já expostos, da importância de se ter um corretor para diminuir o tempo e esforço de correção é que este trabalho se desenvolve, esperando contribuir, também, para a que as pesquisas nessa área possam aumentar.

1.2 MEDODOLOGIA

Para que fosse possível concretizar o objetivo deste trabalho, ou seja, desenvolver um software de correção de provas objetivas, foram esquematizados como metodologia os seguintes procedimentos: inicialmente foi realizada uma revisão bibliográfica das linguagem utilizas no processo de criação do software, bem como a apreciação pedagógica sobre as avaliações objetivas; em seguida, foi desenvolvido o software de correção, o qual, a princípio, foi implementado utilizando a ferramenta MATLAB, para se avaliar viabilidade do desenvolvimento e seu funcionamento, sendo posteriormente desenvolvido para uma versão do software para dispositivos móveis, especificamente para dispositivos que rodem a plataforma Android.

1.3 OBJETIVOS

Os objetivos a que se propõe este Trabalho estão divididos em geral e específicos.

1.3.1 Objetivo Geral

Desenvolver um software específico para correção de avaliações objetivas.

1.3.2 Objetivos específicos

Para que fosse possível alcançar o objetivo geral a que se destina esse trabalho, foram traçados alguns objetivos específicos:

- Especificar o contexto e o que se necessita para desenvolver o software;
- Desenvolver o protótipo utilizando o MATLAB;
- Desenvolver o aplicativo para Android;
- Testar se o software está cumprindo o seu objetivo;
- Validar os resultados.

2 REVISÃO DA LITERATURA

2.1 SOLUÇÕES DISPONÍVEIS NO MERCADO

Antes da proposição de desenvolvimento de uma nova ferramenta computacional, realizou-se uma pesquisa bibliográfica visando buscar a existência de algum sistema ou software que tenham funções correspondentes ao proposto. Isto foi feito no intuito de auferir, e consequentemente justificar, a viabilidade de proposição e implementação de um novo aplicativo, com funcionalidades diferenciadas e que se encaixam às necessidades apresentadas.

Algumas ferramentas de gerência de provas foram encontradas, são elas:

- Corretor de prova: O Sistema Corretor de Prova é um produto integrado ao Sistema Mais Escola e que possui a solução completa para corrigir as provas com base num gabarito pré-cadastrado.
- SiaaE: Um sistema para correção, análise e avaliação de dados estatísticos voltados para escolas, universidades e cursos em geral dando total apoio na correção de avaliações por meio de escaneamento de imagens e processamento de resultados.

Os sistemas encontrados são eficientes quando se trata de cadastro, produção e correção de questões. Estas ferramentas possuem desvantagens como o fato da obrigatoriedade de ter o programa instalado em uma máquina específica, limitando sua instalação, algumas com manutenção financeira de custo elevado, o que prende o professor à um local para aplicação e correção das provas, pois utilizam um *scanner* para realizar a correção.

Além do fato de nenhuma das soluções encontradas oferecerem o recurso da correção do gabarito a partir de um dispositivo móvel.

Baseado nessa análise pode-se afirmar que é válido o desenvolvimento de uma nova ferramenta, que não necessite de um computador para que se possa fazer correções das provas**.**

2.2 MODELO AVALIATIVO

A necessidade de avaliar está presente em inúmeras instituições, organizações públicas e privadas, empresas, entre outras, se materializando das mais variadas formas, sendo a avaliação direta a mais utilizada, porém não única forma de provar conhecimentos, podendo-se também desenvolver-se em meio a uma discussão, uma entrevista ou até em uma dinâmica, onde o processo condutor varia de acordo com cada pessoa, dependendo do contexto e da temática que se tem para avaliar.

Contudo é dentro do ambiente escolar que a avaliação se apresenta de forma mais contundente, diversificada e impactante, pois a mesma utiliza as provas como meio para avaliar o desenvolvimento do conhecimento e do aprendizado escolar.

Segundo Bloom, Hastings e Madausa (1983) a avaliação se apresenta tão somente como forma de medir desempenho individual, mas também, e principalmente como uma forma de verificar se os alunos estão compreendendo os assuntos, se existem falhas nos métodos de ensino ou no conteúdo transmitido e se os mesmos estão aproveitando de forma satisfatória o conteúdo que está sendo transmitido, de forma que o aprendiz consiga ser utilizado posteriormente.

As avaliações podem ser realizadas utilizando os mais variados métodos, podendo mudar de acordo com o perfil do professor, de acordo com o conteúdo que está sendo trabalhado e do perfil do público alvo.

É de fácil percepção, que em várias instituições o método de coletar e analisar o processo de ensino-aprendizagem dos alunos no decorrer dos períodos é as provas, as quais são consideradas um instrumento de avaliação do estágio de aprendizado do aluno.

A utilização de provas como instrumento de avaliação do processo de aprendizagem dos alunos, proporciona ao professor poder avaliar se o conteúdo das aulas estão sendo apreendido de forma adequada, em que parte do conteúdo o aluno está com dificuldades, se a forma de trabalhar o conteúdo é mais adequada, e assim, trabalhar para melhorar o processo ensino-aprendizagem.

As provas podem ser diferentes quanto a sua forma, como por exemplo: prova objetiva, descritiva, oral, prova com consulta, seminários, dentre outras. A forma utilizada neste trabalho é a prova objetiva e por tal nos deteremos falando um pouco dele.

2.1.1 Avaliação objetiva

A prova objetiva se destaca por possuir características que permite a sua utilização não somente no contexto escolar, mas também em processos seletivos, certificação profissional, ou avaliações em larga escala, pois a mesmo fornece uma ampla amostra de conhecimento do aluno, já que é formada por inúmeras questões, possibilitando um julgamento objetivo e rápido. Além disso, possibilita uma correção relativamente simples e rápida e elimina aspectos subjetivos dos alunos e também dos corretores. Porém, possui como desvantagem o fato de que às provas de caráter objetivo são difíceis de serem elaboradas, restringem a capacidade de argumentação dos alunos e necessitam de atenção durante a aplicação.

No processo de elaboração da prova objetiva o autor aborda as questões objetivas em diferentes vertentes, focos, aspectos ou interpretações, podendo variar e diversificar esses formatos, mesmo mantendo a rigidez da estrutura do modelo direto de perguntas e respostas. As questões para esse tipo de abordagem podem ser de verdadeiro ou falso, de múltipla escolha, preenchimento de lacunas e de associações.

As provas objetivas com questões do tipo verdadeira ou falsa possuem assertivas com valores lógicos verdadeiros ou falsos em relação ao argumento utilizado. A Figura 1 apresenta uma questão desse tipo.

Figura 1 - Questão do tipo verdadeiro ou falso

(QUESTÃO – 1): Assinale Verdadeiro (V) ou Falso (F)

( ) A luminosidade é o quanto a cor se afasta da cor neutra.
( ) As cores complementares são amarelo, magenta e ciano.
( ) As cores primárias são tais que, se misturadas 2 a 2, não resultam na terceira, e misturadas entre si resultam no branco.
( ) Na síntese subtrativa, 3 cores primárias, quando somadas, resultam na cor branca.
( ) No modelo de Munsell as variações de cor do modelo representam as variações perceptivas de cor
( ) O modelo RGB para especificação da cor não tem relação com a técnica de produção de cores das telas de computadores
( ) O sistema de Munsell é descrito em termos de Tom, Valor e Saturação
( ) Quanto mais saturada é a cor, menos presença de branco ou preto.
( ) Tom de uma cor é definido pelo seu comprimento de onda no espectro visível.

Fonte:http://www.ifsuldeminas.edu.br/~concurso/attachments/article/143/Prova%20Agrimensura%20e%20Cartografia%20II.pdf

As questões de múltiplas escolhas são aquelas na qual o enunciado pede uma resposta e somente uma opção está correta. Este tipo de questões são de difícil elaboração, pois o autor tem que procurar fazer com que todas as alternativas estejam muito próximas da verdade, porém somente uma esteja totalmente correta, para evitar acertos somente por indução. Na Figura 2 é exposto esse tipo de questão.

Figura 2 - Questão do tipo múltiplas escolhas

(QUESTÃO – 4): Em território nacional, o indicador para trabalhar com controle de precisão cartográfica é:
a) PEC – Padrão de Exatidão Cartográfica
b) CEC – Controle Estatístico Cartográfico
c) Desvio-padrão
d) IEC – Indicador Estatístico Cartográfico

Fonte:http://www.ifsuldeminas.edu.br/~concurso/attachments/article/143/Prova%20Agrimensura%20e%20Cartografia%20II.pdf

As provas com questões associativas apresentam itens que devem ser associados por meio de definições. Tais questões possuem muitas maneiras de associação dificultando o acerto ao acaso, sem o devido estudo. A Figura 3 apresenta esse tipo de questão.

Figura 3 - Questão do tipo associativa

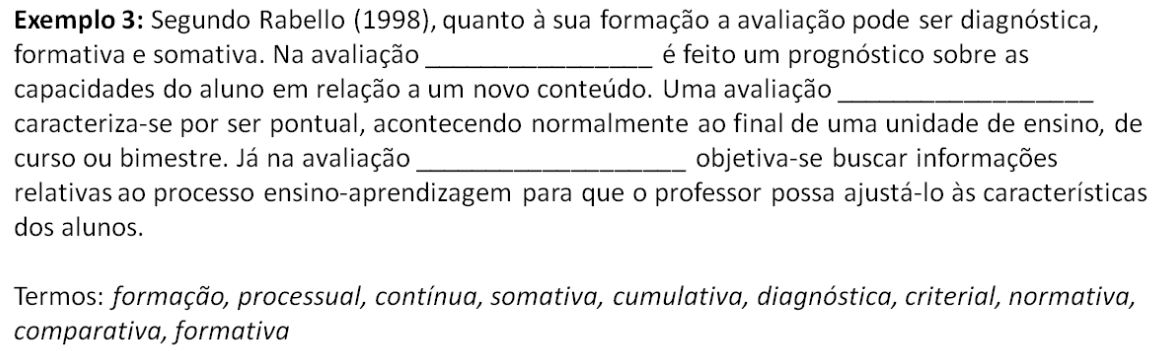
Exemplo 4: Relacione as obras literárias com seus respectivos autores.

a - Érico Verissimo	( ) O Dom Supremo
b - Gabriel Garcia Marques	( ) Dom casmurro
c - Paulo Coelho	( ) O Tempo e o Vento
d - Liz Fernando Veríssimo	( ) Cem Anos de Solidão
e - Machado de Assis	( ) O Analista de Bagé

Fonte:http://www.ifsuldeminas.edu.br/~concurso/attachments/article/143/Prova%20Agrimensura%20e%20Cartografia%20II.pdf

As questões que pedem o preenchimento de lacunas desejam que os candidatos preencham de forma satisfatória os espaços em branco, sendo as opções de respostas dadas em seguida do texto. A Figura 4 exemplifica esse tipo de questão.

Figura 4 - Questão do tipo preenchimento de lacunas


Exemplo 3: Segundo Rabello (1998), quanto à sua formação a avaliação pode ser diagnóstica, formativa e somativa. Na avaliação _____ é feito um prognóstico sobre as capacidades do aluno em relação a um novo conteúdo. Uma avaliação _____ caracteriza-se por ser pontual, acontecendo normalmente ao final de uma unidade de ensino, de curso ou bimestre. Já na avaliação _____ objetiva-se buscar informações relativas ao processo ensino-aprendizagem para que o professor possa ajustá-lo às características dos alunos.

Termos: *formação, processual, contínua, somativa, cumulativa, diagnóstica, criterial, normativa, comparativa, formativa*

Fonte: http://sitenotadez.net/portugues-gramatica/

2.3 TECNOLOGIAS PARA O DESENVOLVIMENTO

A proposta apresentada neste trabalho requer o desenvolvimento de um protótipo, para análise da viabilidade da sua implementação, e por fim o desenvolvimento da solução para dispositivos móveis. O texto que se segue não é uma análise das tantas ferramentas de desenvolvimento existentes, ao contrário, é uma justificativa da escolha de apenas uma delas em cada contexto, que neste caso são: Matlab (para o desenvolvimento do protótipo) e Android (para o desenvolvimento em dispositivos móveis).

2.3.1 MATLAB e sua linguagem.

O MATLAB original foi desenvolvido em linguagem Fortran por Clever Moler. A partir da versão 5 foi desenvolvido em linguagem C por: Steve Bangeret, Steve Kleiman e Clever Moler – Stanford University.

Desde a sua primeira versão, o MATLAB é tido como produto líder na área de computação numérica e científica. Mais do que um software, o MATLAB é um ambiente integrado de modelagem de sistemas e algoritmos, ideal para implementação de projetos complexos, e que por esta razão vem sendo adotado como ferramenta de desenvolvimento padrão pelas principais universidades do Brasil e do mundo. O MATLAB é um software

destinado a fazer cálculos com matrizes (matriz é o seu elemento essencial). O nome MATLAB é derivado de MATrix LABoratory, ou seja, um laboratório de matrizes.

O MATLAB é um sistema interativo cujo elemento básico da informação é uma matriz que não requer dimensionamento. Esse sistema permite a resolução de muitos programas numéricos em apenas uma fração do tempo que se gastaria para escrever um programa semelhante em linguagem tradicional como Fortran, Basic, C/C++, Delphi, Visual Basic, etc.

Assim o MATLAB é uma ferramenta e uma linguagem de programação de alto nível, e tem como principais funções: construção de gráficos e compilação de funções, manipulação de funções específicas de cálculo e variáveis simbólicas.

Além disso, o MATLAB possui uma grande quantidade de bibliotecas auxiliares ("Toolboxes") que otimizam o tempo gasto para realizar tarefas, uma vez que, o usuário poderá utilizar muitas funções já definidas, poupando o tempo de criá-las. Por outro lado, os programas desenvolvidos são difíceis de serem executados em um ambiente fora do MATLAB.

2.3.2 Android

O mercado de dispositivos móveis vem crescendo admiravelmente. De acordo com Ávila (2013), a penetração de celulares no mundo é de 87%. De acordo com a pesquisa, o serviço de telefonia móvel vem barateando gradativamente, existem atualmente cerca de 8 bilhões de linhas móveis ativas no mundo, ou seja, praticamente existe um aparelho móvel para cada pessoa no mundo, incluindo crianças.

O que antes eram diferenciais, só em aparelhos de custo elevado, hoje em dia, usuários comuns estão buscando cada vez mais celulares com recursos como câmeras de alta definição, musicas, bluetooth, capacidade para jogos, GPS, acesso à internet e principalmente a redes sociais e TV digital (LECHETA, 2013, p. 20).

Na área corporativa a procura por dispositivos móveis, como *tablets* ou *smartphones*, vem crescendo também. Diversas empresas procuram incorporar suas aplicações convencionais para dispositivos móveis, tornando seus negócios mais ágeis e flexíveis e alimentando cada vez mais o conceito "mobilidade" (FARTO, 2010).

Para acompanhar o nicho de mercado e a evolução tecnológica, satisfazendo usuários, corporações e desenvolvedores, o Android é uma resposta da Google para ocupar este espaço. Atraindo a atenção de milhares de empresas e usuários, o anuncio da tecnologia causou muito impacto por ser uma plataforma Open Source e liderada pela Google, empresa revolucionaria e líder na internet. Além da Google, outras empresas de telefonia móvel investem e apoiam esta tecnologia, o grupo foi chamado de OHA (Open Handset Alliance).

2.3.2.1 Maquina virtual Dalvik

Para desenvolver um aplicativo Android, o desenvolvedor terá que utilizar Java como linguagem de programação, sendo a arquitetura não utiliza a JVM (Java Virtual Machine) como sua máquina virtual, mas uma máquina virtual otimizada para execuções em dispositivos móveis, chamada Dalvik.

Ao desenvolver aplicativos Android, o desenvolvedor contará com todos os recursos Java, mais quando o *bytecode* (class) é submetido à compilação, ele é convertido para um formato (.dex), isto é, Dalvik Executable, que representa uma aplicação Android compilada.

Em seguida, todos os arquivos com extensão *.dex*, outros recursos como imagens, sons e vídeos são compactados em um único arquivo com a extensão *.apk* - *Android Package File*, que representa a aplicação final, pronta para ser instalada e distribuída a qualquer dispositivo que execute o sistema operacional Android (LECHETA, 2013, p. 20).

2.3.2.2 Plataforma Android

A Figura 5 demonstra toda arquitetura da plataforma Android. Será feita uma descrição detalhada de cada nível/camada da arquitetura, pois é de fundamental importância um entendimento detalhado para o desenvolvimento de um aplicativo mais eficiente.

Figura 5 – Arquitetura do Android

Fonte: http://pt.slideshare.net/TheRonildoOliveira/arquitetura-da-plataforma-android

A ilustração da arquitetura Google Android, foi dividida na ilustração da Figura 5 em níveis, sendo que o nível mais baixo presta serviço e da base ao nível acima, cada nível será discutido nas seções seguintes.

2.3.2.2.1 Camada Núcleo Linux

Utilizando a versão 2.6 do *kernel* do Linux, este nível é responsável pelos serviços centrais do sistema Android, como gestão de memória, processos, segurança e tarefas. Este nível também opera como uma camada de *middleware* entre as camadas das plataformas Android, abstraindo toda a complexidade do hardware (PEREIRA; SILVA, 2009, p. 9).

2.3.2.2.2 Camada Android Execução/Tempo

Cada aplicativo do Android executa seu próprio processo, instanciado previamente pela máquina virtual Dalvik. Os processos são executados no formato *Dalvik Executable* (*.dex*) e otimizados para ocupar uma pequena quantidade de memória. Esta camada tem o objetivo de controlar as bibliotecas centrais no núcleo (*core libraries*), e consequentemente controlar as instancias da máquina virtual Dalvik e seus processos otimizados (PEREIRA; SILVA, 2009, p. 8).

2.3.2.2.3 Camada Bibliotecas

A arquitetura Android possui um conjunto de bibliotecas C/C++ usadas por diversos componentes. Tais bibliotecas permitem trabalhar com arquivos de mídia comuns como MPEG4, H.264, MP3, AAC, AMR, JPG e PNG. Componentes para elaboração de projetos em 2D como em 3D, inclusive, uma biblioteca cuja implementação foi baseada em Open Graphics Library (OpenGL), fornecendo acesso a praticamente todos os recursos do hardware de vídeo. Juntamente com esta camada, foi disponibilizado o banco de dados SQLite (PEREIRA; SILVA, 2009, p. 7).

2.3.2.2.4 Camada Quadro de aplicações

A arquitetura deste framework foi desenvolvida com o objetivo de simplificar a programação e a reutilização dos componentes, sendo possível qualquer desenvolvedor construir um aplicativo e permitindo que este seja utilizado por outros programas. Sendo assim, está camada é responsável por disponibilizar todas as API (do inglês Interface de Programação de Aplicações e recursos necessários para pacotes e aplicativos (PEREIRA; SILVA, 2009, p. 6).

2.3.2.2.5 Camada Aplicações

É a camada onde se encontram todos os aplicativos Google Android, como clientes de e-mail, contatos, navegador web, aplicativos GPS. Todos os projetos implementados encontram-se neste nível (PEREIRA; SILVA, 2009, p. 6).

2.3.2.3 Android SDK

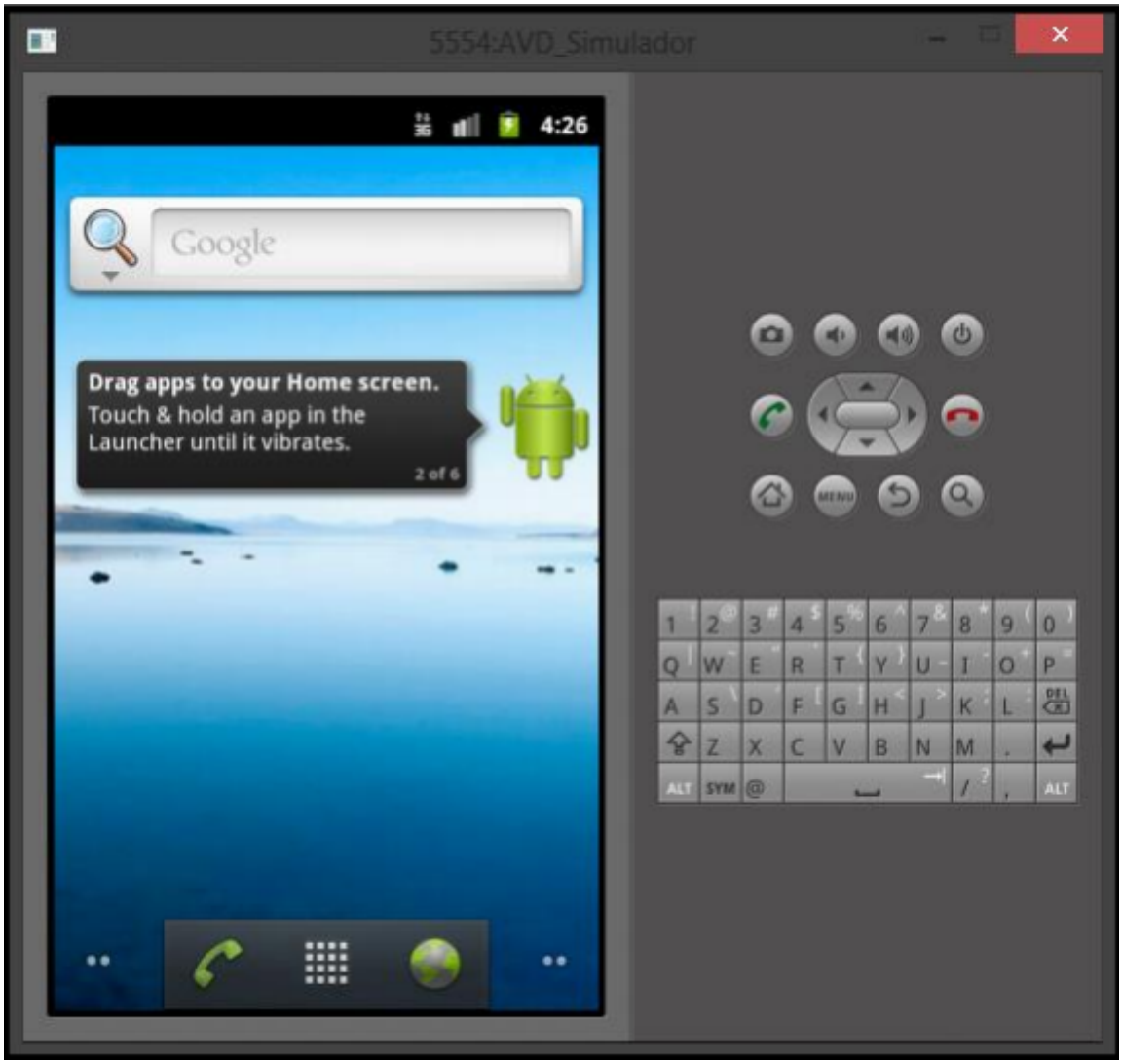
O Android SDK é um software que fornece as bibliotecas da API e instrumentos de desenvolvimento necessários para construir, testar e depurar aplicativos para a plataforma móvel Google Android (LECHETA, 2013, p. 27).

A plataforma possui um emulador de simulação de um dispositivo móvel, fornecendo praticamente todas as funções que um celular ou tablet, dando ao desenvolvedor total condição de realizações de testes.

Apesar de o Android SDK possuir o emulador que é executado normalmente no computador, como um aplicativo convencional, a plataforma fornece um plug-in para o Ambiente de Desenvolvimento Eclipse, cujo o objetivo é integrar a programação em Java com o Emulador (LECHETA, 2013, p. 27).

A Figura 6 ilustra o emulador Android.

Figura 6 – Android SDK



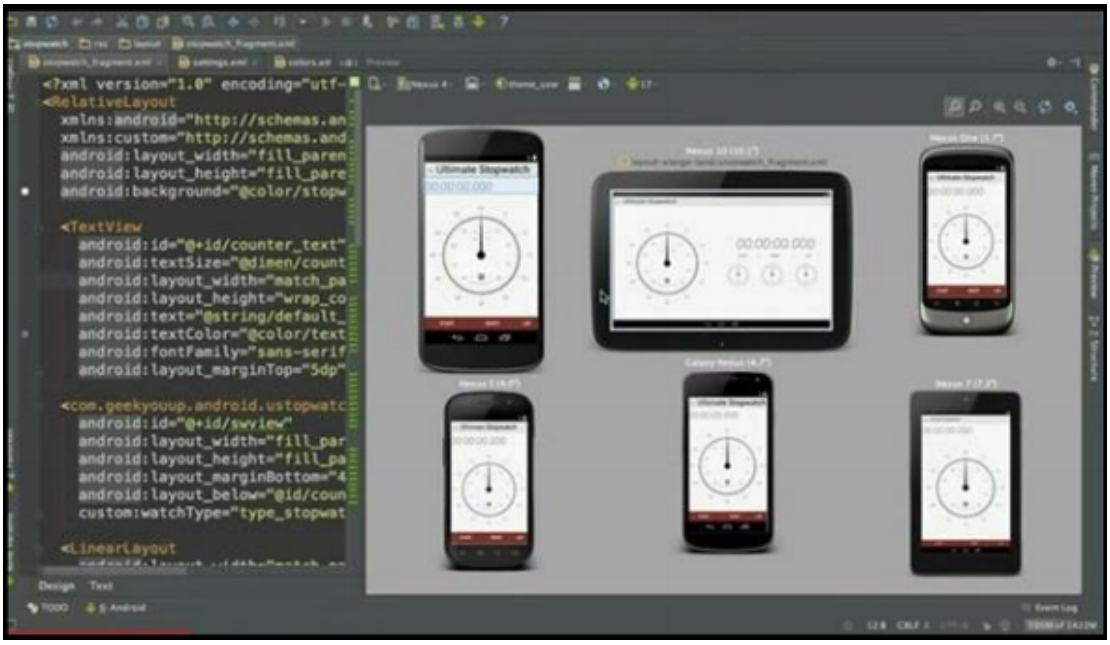
Fonte: http://pt.kioskea.net/download/baixaki-11598-android-sdk

A partir do plug-in, é possível iniciar o emulador junto ao Eclipse, dando ao mesmo a capacidade de realizar depurações no código de forma automática, como qualquer outra aplicação.

Atualmente, mas ainda como em fase de testes, a própria Google disponibilizou uma plataforma de desenvolvimento e depuração para aplicativos Android chamada Android Studio, cujo o objetivo é fornecer ferramentas integradas para o desenvolvimento de aplicativos na plataforma, e dar ao desenvolvedor uma vasta opção de edições de layouts e ferramentas de soluções rápidas, tudo já incluso dentro da suíte. O ambiente foi apresentado na última conferência anual da empresa, Google I/O (BARROS, 2013).

A Figura 7 mostra a ferramenta Android Studio.

Figura 7 – Android Studio



Fonte: http://imasters.com.br/mobile/android/android-studio-vantagens-e-desvantagens-com-relacao-ao-eclipse/

De acordo com Barros (2013), a plataforma tem grande possibilidade de ascensão, pois é totalmente Open Source e está sendo desenvolvida o que dará ao produto mais benefícios e facilidades ao programador.

Ainda que em fase testes, muitos desenvolvedores já estão migrando para este ambiente de desenvolvimento, pois poderão fazer sugestões de mudanças e indicações para a versão final, assim atraindo um público alvo de programadores ainda maior.

Uma das funções mais interessantes deste novo ambiente de desenvolvimento é a capacidade de exibir uma pré-visualização em tempo real do aplicativo em desenvolvimento, nos múltiplos formatos. A medida que se desenvolve, é possível avaliar como o layout da aplicação fica em smartphones e tabletes de vários tamanhos e resoluções, permitindo realizar ajustes e aproveitar espaços livres (no caso de telas maiores).

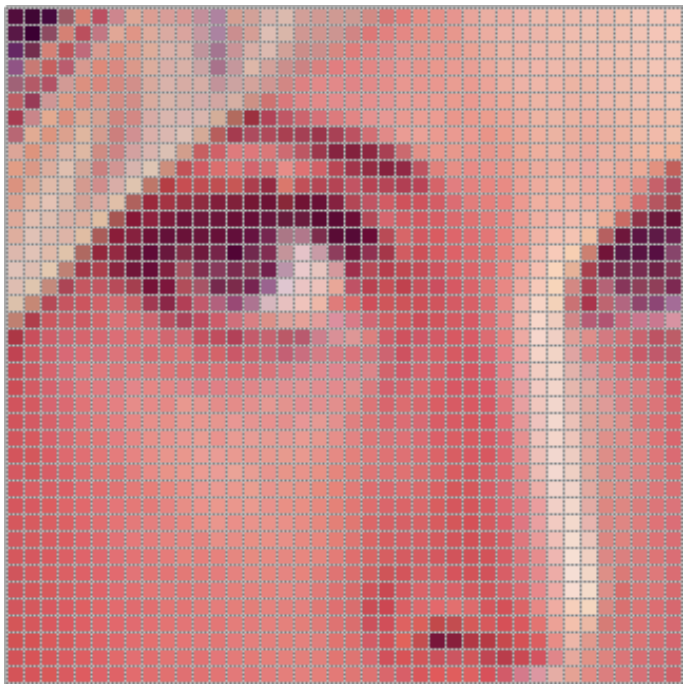
2.4 PROCESSAMENTO DIGITAL DE IMAGENS

Diferente da visão humana, o computador visualiza uma imagem em forma de células, ou seja, de forma matricial, sendo que cada ponto desta matriz, contém informações, e a soma

de todas as informações forma-se uma imagem (ALBUQUERQUE; ALBUQUERQUE, 2009).

Uma imagem digital linearizada ou monocromática é uma função bidimensional (x, y) que representam intensidades luminosas, onde x e y são coordenadas espaciais, que por convenção x = [1, 2, ..., N] e y = [1, 2, ..., N], sendo que f na posição (x, y) é regular ao nível de cinza (ou brilho) da imagem nesta localidade. A Figura 8 demonstra a ideia de um pixel em uma imagem.

Figura 8 – Ideia de um pixel de uma imagem



Fonte: http://jamarsmuniz.blogspot.com.br/2012/12/mitos-dos-pixels.html

Um pixel é o elemento mais básico de uma imagem. A maneira mais básica para o pixel é a forma retangular ou quadrada. Ele também é um componente de dimensões finitas na reprodução de uma imagem digital (SERRANO, 2010).

A Figura 8 foi dividida em três partes: A parte 1, demonstra a figura inteira, na integra, juntamente com o plano cartesiano (x, y), representando a localização dos pixels da imagem. Ainda na parte 1, pode-se verificar que a imagem foi selecionada. A parte 2 da figura, é uma imagem ampliada da seleção da parte 1. Já a parte 3, é a ampliação da imagem 2, sendo que nela é possível verificar a representação de um pixel da imagem.

A partir de uma imagem digital, pode-se aplicar operações aritméticas e lógicas, pois sua representação é totalmente numérica, uma matriz de pixels, ou seja, sua localização, intensidade de luz, tons e cores são números armazenados em cada pixel e diante deste conceito, são criados filtros, melhoria na qualidade das imagens, correções, e criação de efeitos artísticos.

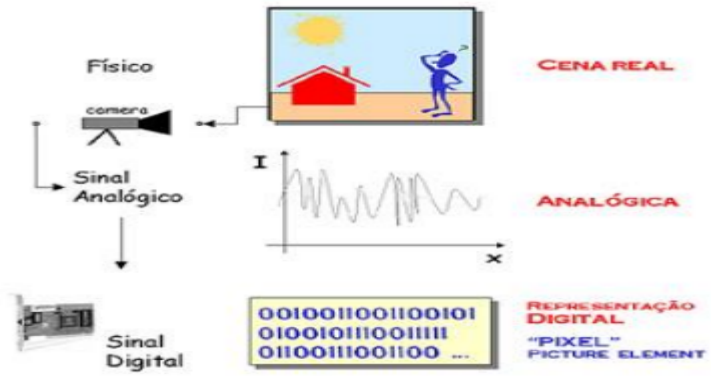
O Processamento Digital de Imagens (PDI) parte da imagem (de uma informação inicial que é geralmente captada por uma câmera) ou de uma sequência de imagens, para a obtenção da informação. Caso a imagem seja analógica, é necessário primeiramente converte-la para digital, em seguida, realizar todo o processamento da mesma.

2.4.1 Aquisição de imagens

Para obter uma imagem digital é necessário um dispositivo fisico, para capturar a imagem e armazena-la em algum local previamente delimitado. Se a imagem for uma imagem analógica, ela deve ser convertida para digital. Para este tipo de conversão, usa-se scaners, que transformam um sinal totalmente continuo, para binário (0 ou 1).

O conversor analógico/digital converte o sinal elétrico que produz o foto-sensor em pulsos digitais em formato binário, que são convertidas pelo computador. Os softwares embarcados nos dispositivos conversores fazem o tratamento e armazenam no computador em um local especifico. A Figura 9 ilustra esta fase:

Figura 9 – Aquisição de imagens

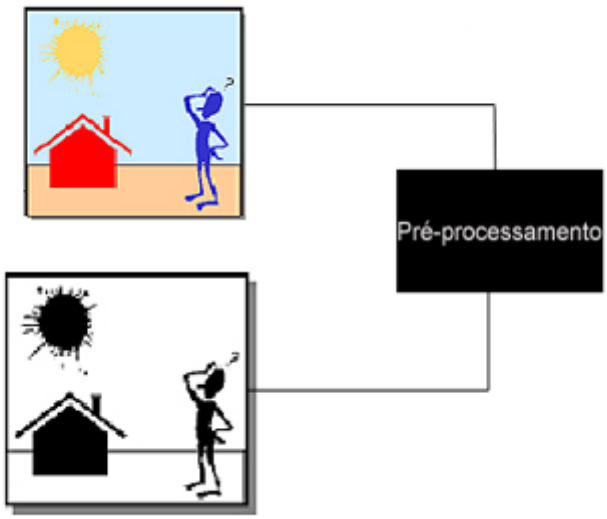


Fonte: http://www.cbpf.br/cat/pdsi/visao/

2.4.2 Pré-processamento

O Pré-Processamento consiste em melhorar a imagem de forma a aumentar as chances de sucesso dos processos seguintes. O processo mais simples para esta fase consiste em duas técnicas: para monocromatização da imagem (transformá-la em escala cinza) e a binarização que consiste em transformar a imagem em preto e branco para diminuir o número de possibilidades das cores de cada pixel de quase 17 milhões para 2 (0 ou 1). A Figura 10 mostra este conceito:

Figura 10 – Pré-processamento de imagens



Fonte: http://www.cbpf.br/cat/pdsi/visao/

2.4.3 Segmentação

A segmentação consiste no processo de separar na imagem os objetos a serem analisados. É usual denominar objetos da imagem os grupos de pixels de interesse, ou que fornecem alguma informação para o processamento digital de imagem (SERRANO, 2010).

Existem dois tipos de segmentação: baseados em pixel, ou baseados em formatos.

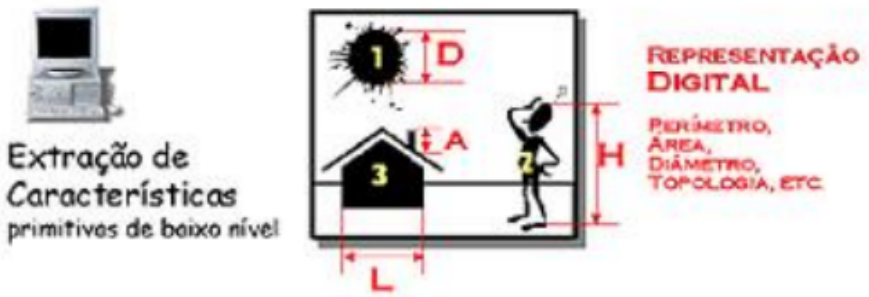
- Baseados em Pixel
    - Segmentação de Intensidade;
    - Segmentação de Cores.
- Baseados em Formatos
    - Detecção de bordas;
    - Detecção de descontinuidades;
    - Detecção de pontos;
    - Detecção de linhas.

A fase da segmentação é a etapa do PDI que deve ser mais eficiente, pois a partir dela que é feita a extração de atributos (NETO; FILHO, 1999).

2.4.4 Extração de atributos

A etapa da extração de atributos é a fase do que consiste em retirar características (informações) da imagem. Esta etapa retira da imagem informações como: tamanho de uma região, distância entre um pixel a outro, altura, área, perímetro, profundidade, entre outros. A Figura 11 ilustra esta etapa:

Figura 11 – Extração de atributos da imagem



Fonte: http://www.cbpf.br/cat/pdsi/visao/

2.4.5 Reconhecimento e interpretação

A etapa de reconhecimento e interpretação é a fase que consiste em comparar de forma "automática" todos os atributos retirados na fase de extração de atributos com as informações já pré-cadastradas ou armazenadas no sistema. Nesta fase, a imagem já passou por todo processamento. A Figura 12 demonstra esta etapa e ilustra um possível exemplo de interpretação:

Figura 12 – Reconhecimento e interpretação de imagens

Fonte: http://www.cbpf.br/cat/pdsi/visao/

A partir da comparação, é feito o reconhecimento, resultando numa resposta satisfatória se for encontrado características similares existentes na imagem processada com alguma armazenada em banco, ou insatisfatória se não encontrar nenhum resultado.

No PDI, a fase de pré-processamento é uma fase muito importante, pois todos os passos a partir desta fase dependem da eficácia e qualidade do pré-processamento.

2.5 BIBLIOTECA OPENCV

Desenvolvida nas linguagens C/C++, no ano 2000 pela Intel, é uma biblioteca totalmente livre ao uso acadêmico e comercial, bastando seguir o modelo de licença da BSD Intel (Berkeley Software Distribution), para o desenvolvimento de sistemas com ênfase em Visão Computacional, possuindo interfaces de C, C++, C#, Python e Java, suporta Windows, Linux, Mac OS, iOS e Android como plataformas. Adotada em todo o mundo, a biblioteca tem mais de 47 mil pessoas da comunidade de usuários e número estimado de download superiores a 7 milhões.

A biblioteca Open Source Computer Vision Library (OpenCV), consiste de vários módulos de processamento de imagens e vídeos, estruturas de dados baseados em álgebra linear, interface gráfica do usuário, controle independente de mouse, teclado e periféricos, além de mais de 500 implementações de Visão Computacional como: filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e outros. Por ser desenvolvida em C/C++, a biblioteca pode tirar proveito do processamento Multi-Core (OPENCV...2013).

Segundo Bradski (2013), a biblioteca OpenCV foi arquitetada para a eficiência computacional e com um forte foco em aplicações em tempo real, com o propósito de tornar a Visão Computacional próxima aos usuários e programadores nas áreas da computação e robótica. Um sistema implementado com OpenCV, ao ser executado, invoca sempre uma Dynamic-Link Library (DLL) que detecta o tipo de processador da máquina que executa a tarefa, realizando assim todas as otimizações de códigos. A biblioteca é dividida em cinco principais grupos de funções:

- "cv.h" – contém o básico de Processamento de Imagens e algoritmos de visão computacional;
- "ml.h" – é biblioteca de Aprendizagem de Máquina e Análise Estrutural, que inclui muitos classificadores estatísticos;
- "cxcore.h" – contém as estruturas de dados básicos;
- "highgui.h" – contém rotinas I/O e funções para armazenar e carregar imagens e vídeos;
- "cvaux.h" – contém as áreas de reconhecimento facial e algoritmos experimentais.

Como citado anteriormente, a grande preocupação para o desenvolvimento de um aplicativo ágil e eficaz é um estudo detalhado na fase de pré-processamento, mas o desenvolvimento de um aplicativo para um dispositivo móvel há também a apreensão quanto

a taxa de processamento, limitação de memória e quantidade de processos ativos, preocupação esta, que não representa um grande problema no desenvolvimento de um aplicativo convencional. A união desta biblioteca com a plataforma Android, trará ao aplicativo um benefício muito grande, pois a OpenCV trata de forma eficiente e rápida as imagens nas quais são submetidas a processamento, utilizando a menor quantidade de memória e taxa de processamento possível.

3 METODOLOGIA

Antes do desenvolvimento da versão final do software, foi desenvolvido um protótipo utilizando o MATLAB, por fim foi desenvolvido a versão final do software para dispositivos móveis que rodem a plataforma Android. Tanto o desenvolvimento em MATLAB como o desenvolvimento para Android estão descritos neste capítulo.

3.1 IMPLEMENTAÇÃO DO PROTÓTIPO

Para avaliar de maneira prática a solução proposta, foi implementado um protótipo do software utilizando-se o software MATLAB.

A linguagem Matlab "permite implementar e resolver problemas matemáticos muito mais rápida e eficientemente que através de outras linguagens como C, Basic, Pascal ou 37 Fortran." (CBPF, 2002). Devido a sua capacidade de processamento matemático, a linguagem é amplamente utilizada por diversas áreas das Ciências Exatas, desde engenheiros a economistas utilizam-se da facilidade de lidar com matrizes ou manipulação algébrica.

3.1.1 Tela de usuário

No protótipo desenvolvido em MATLAB existe somente uma tela de usuário. Essa tela é apresentada ao rodar o software, nela são exibidas três *axes* (elemento do GUI utilizado para exibição de imagens) onde são mostrados, respectivamente:

Axes 1: A imagem que está sendo capturada da câmera no momento;

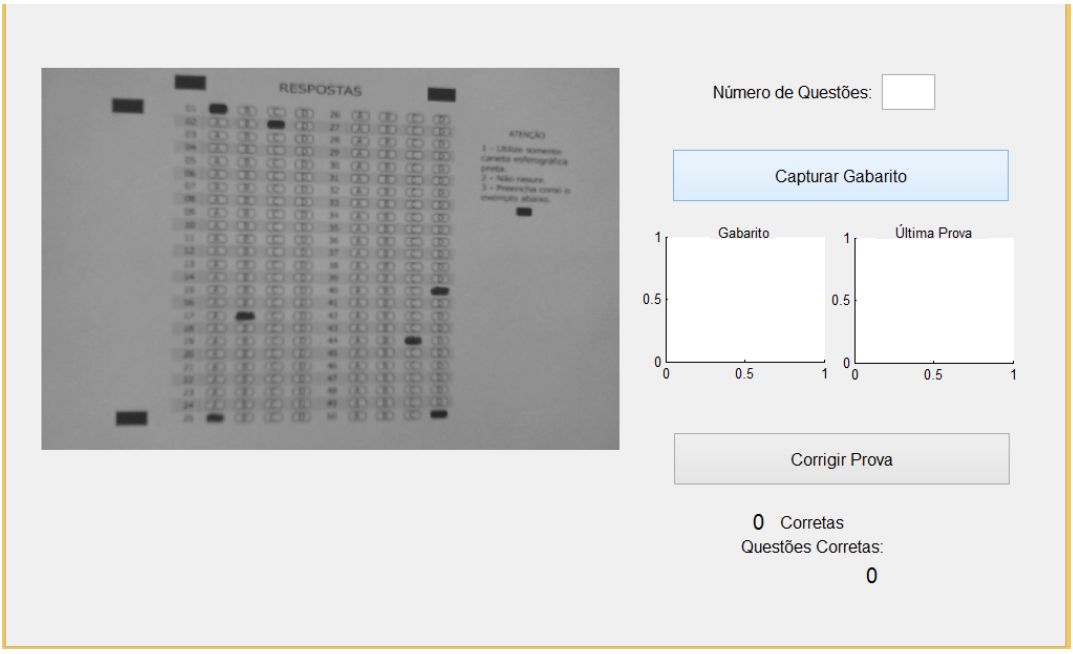
Axes 2: A imagem do gabarito que foi capturado para tomar como base para a correção as outras provas;

Axes 3: A imagem da última prova que foi corrigida;

Além dos *axes* onde exibem-se as imagem, existem um campo de texto e dois botões: O campo de texto é utilizado para determinar por quantas questões é composto o gabarito; O primeiro botão captura e realiza o processo sobre o gabarito que servirá de base para correção das demais provas; O segundo botão captura e realiza o processo sobre a prova que deve ser corrigida e exibe o total de questões corretas, como também quais questões estão corretas.

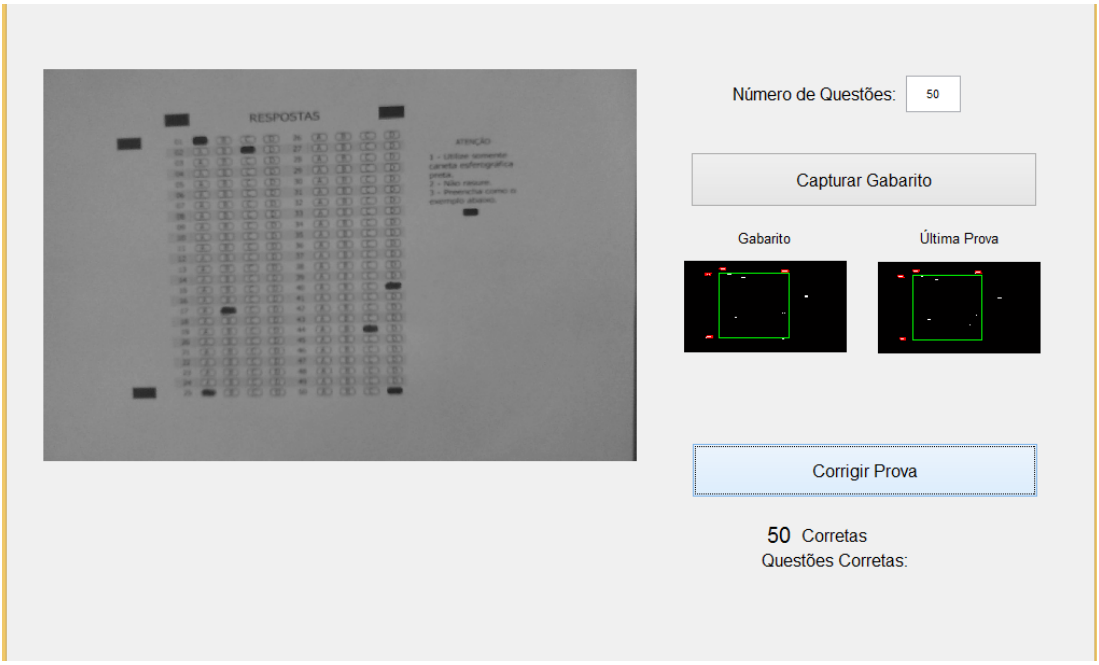
A Figura 13 mostra a tela inicial do protótipo, onde é exibido no *axes* 1 a imagem que está sendo capturada da câmera. A Figura 14 mostra o resultado da correção de uma prova, onde são exibidos o gabarito capturado e a prova capturada, no *axes* 2 e *axes* 3, respectivamente.

Figura 13 – Tela inicial do protótipo



Fonte: Autoria própria

Figura 14 – Tela do protótipo com uma prova corrigida



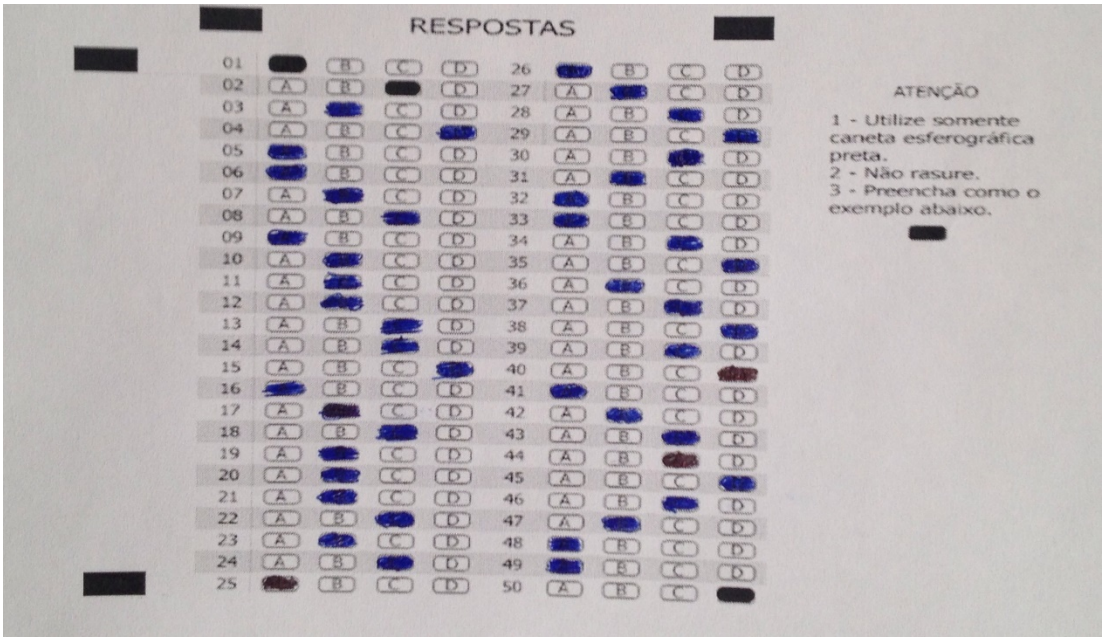
Fonte: Autoria própria

3.1.2 Correção através do protótipo desenvolvido em MATLAB

O sistema se baseia na obtenção de uma imagem de formulário de gabarito impresso, previamente definido e o processamento de seu conteúdo.

A Figura 15 apresenta a imagem de um formulário de gabarito impresso obtida através da WebCam que foi utilizada no desenvolvimento do protótipo.

Figura 15 – Imagem do gabarito capturado



Fonte: Autoria própria

3.1.3 Trechos de código do protótipo desenvolvido em MATLAB

A Figura 16 mostra o início da implementação do protótipo. Inicialmente é feita a definição da câmera que será utilizada para captura das imagens, a definição é feita pela função *videoinput*. Nessa função são passados três itens como parâmetro:

- *winvideo* – que é o nome do adaptador de vídeo que será utilizado para a aquisição da imagem;
- 1 – que indica qual câmera será escolhida entre as listadas pelo adaptador;
- *YUY2_1280x720* – que define qual o esquema de cores utilizado pela câmera, e resolução em que a mesma irá trabalhar.

Para o desenvolvimento da aplicação foi utilizado a câmera Microsoft Webcam LifeCam HD 3000.

Em seguida a imagem é transformada em tons de cinza e é exibida na tela pela função *preview* obedecendo a resolução (*VideoResolution*) e o número de bandas (*NuberOfBands*).

Figura 16 – Define câmera, transforma em tons de cinza e exibe.

```
79 -  global vid;
80 -  vid = videoinput('winvideo',1,'YUY2_1280x720');
81 -  set(vid,'ReturnedColorSpace','gray');
82 -  vidRes=get(vid,'VideoResolution');
83 -  Nbands=get(vid,'NumberOfBands');
84 -  himage=image(zeros(vidRes(2),vidRes(1),Nbands),'Parent',handles.camera);
85 -  preview(vid,himage);
86
```

Fonte: Autoria própria

Ao pressionar o botão “Capturar Gabarito”, inicia-se a segunda fase do processo, conforme mostrado na Figura 17, a fase onde é capturado e processado o gabarito que será utilizado como base para a correção das demais provas.

A função *getsnapshot* (linha 97) captura um frame da câmera e guarda na variável **gabarito**. A função *imwrite* (linha 98) salva o frame que foi capturado em um arquivo chamado “gabarito.jpg”. Logo após o arquivo é lido (*imread*) para a variável **I_cam** onde será processado.

Figura 17 – Captura imagem, salva e ler para variável.

```
89      % --- Executes on button press in pushbutton1.
90    function pushbutton1_Callback(hObject, eventdata, handles)
91    % hObject    handle to pushbutton1 (see GCBO)
92    % eventdata  reserved - to be defined in a future version of MATLAB
93    % handles    structure with handles and user data (see GUIDATA)
94 -  global vid;
95 -  global gabarito;
96
97 -  gabarito = getsnapshot(vid);
98 -  imwrite(gabarito,'gabarito.jpg','jpg');
99 -  I_cam = imread('gabarito.jpg');
100
```

Fonte: Autoria própria

O próximo passo consiste em transformar a imagem que foi captura em uma imagem binária (que será representada por zeros e uns).

Para tal é utilizada a função *im2bw*. *BW = im2bw(I, level)* (linha 108) converte uma imagem em escala de cinza para uma binária. A imagem de saída substitui todos os pixels da imagem de entrada por 1 (branco), onde os pixels tiverem maior luminância, e por 0 (preto) nos pixels de menor luminância. O level deve ser especificado entre 0 e 1. Esse intervalo é em relação aos níveis possíveis para a classe da imagem.

A função *B = blockproc (A, [MN], fun)* processa a imagem A, aplicando a função *fun* para cada bloco distinto M-por-N de A e concatenando os resultados em B, a matriz de saída.

Para cada bloco de dados na imagem de entrada, A, *blockproc* passa o bloco em um *struct block* para a função do usuário, o bloco correspondente na imagem de saída. Se vazio, *blockproc* não gera qualquer saída e retorna vazio após o processamento de todos os blocos. Escolher um tamanho de bloco apropriado pode melhorar significativamente o desempenho. A Figura 18 mostra como é realizado a chamada das funções para binarizar a imagem.

Figura 18 – Conversão em imagem binária.

```
108 -   BW = @(I_cam) im2bw(I_cam.data,median(double(I_cam.data(:)))/1.2/600);
109 -   I_cam = ~blockproc(I_cam,[92 92],BW);
```

Fonte: Autoria própria

O comando *[B,L] = bwboundaries(BW,'noholes'),* na linha 133 da Figura 19, traça os limites exteriores de objetos, bem como limites de buracos no interior desses objetos, na imagem binária BW. BW deve ser uma imagem binária onde os pixels 1 pertencem a um objeto e 0 pixels constituem o fundo. No prótotipo desenvolvido a função bwboundaries foi utilizada para localizar os pontos que delimitam a área da folha de resposta, para a partir desses pontos poder ser localizadas as questões que foram marcadas.

A função imshow(I) é utilizada para mostrar a imagem binária na tela.

Figura 19 – Encontra os pontos e mostra a imagem na tela.

```
132
133 -   [B,L] = bwboundaries(I_cam,'noholes');
134 -   imshow(L,'Parent', handles.gabarito);
135
```

Fonte: Autoria própria

A função *[B,L] = bwboundaries(BW,'noholes')* retorna dois argumentos, a quantidade de objetos localizados na imagem (B) como também a marcação de cada um desses objetos (L), essa marcação consiste em todos os pontos (x,y) que estão no entorno de cada objeto presente na imagem.

As próximas linhas localizam o os quatro pontos iguais que delimitam o entorno da parte destinada as respostas, como mostra a Figura 20.

A função *plot(x,y)* é utilizada para marcar os pontos encontrados que serão tomados como base para o software localizadar a região desejada.

Figura 20 – Encontra os pontos e plota na tela.

```
136
137 -  pos_marc=1;
138 -  for j = 1:length(B)
139 -      if (length(B{j}) > (length(B{1})*0.90)) && (length(B{j}) < (length(B{1})*1.1))
140 -          marcacao(pos_marc) = B(j);
141 -          pos_marc = pos_marc+1;
142 -      end
143 -  end
144
145 -  hold on
146 -  for k = 1:4
147 -      boundary = marcacao{k};
148 -      plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 1);
149 -  end
150
```

Fonte: Autoria própria

Após os pontos serem encontrados, eles são tomados como base para se definir o tamanho de cada célula que representará as questões e alternativas.

A função *y = linspace(a,b,n)* (linha 210) gera vetores espaçados linearmente. A função *linspace* é utilizada para dividir a imagem em linhas e colunas e armazenar as posições (x,y) onde são começa e termina cada uma dessas células.

O modelo de gabarito utilizado no software, mostrado na Figura 21, possui 50 questões, onde, da questão 1 à questão 25 estão organizadas verticalmente, e da questão 26 à questão 50 estão dispostas ao lado das 25 primeiras. Com isso se faz necessária uma matriz, *g[25]10]*, de 25 linhas por 10 colunas para representar todo o gabarito.

Com a divisão das células, é necessário saber se cada célula está pintada ou não, para se obter as respostas para cada questão. Para tal é feita a soma (*sum*) de todos pixels que compõem cada célula, esses valores são armazenados na matriz que representa o gabarito.

A Figura 21, a seguir, mostra o processo de divisão do gabarito em células e soma da área pintada de cada célula.

Figura 21 – Define o tamanho de cada célula e soma o total pintado.

```
210 -  gabx=linspace(xa,xb,11);
211 -  gaby=linspace(ya,yd,26);
212
213 -  g=zeros(25,10);
214 -  k=1;
215 -  for i=26:-1:2
216 -      for j=1:10
217 -          g(k,j)=sum(sum(gab_log(gaby(i):gaby(i-1),gabx(j):gabx(j+1))));
218 -      end
219 -      k = k+1;
220 -  end
```

Fonte: Autoria própria

Depois de obter a matriz que representa o gabarito, é preciso saber se o valor pintado em cada célula representa uma questão marcada, ou se é apenas um risco não proposital, ou sombra. Para tal é analisado o total de pixels que foram pintados na célula. Se a quantidade de pixels pintado for grande, é atribuído o valor 1, que representa questão marcada, se não, é atribuído 0 na matriz que representa o gabarito, como é mostrado da Figura 22 abaixo.

Figura 22 – Transforma as células pintadas em 0 ou 1.

```
230 -  for i=1:25
231 -      for j=1:10
232 -          if (g(i,j))<150
233 -              g(i,j)=0;
234 -          else
235 -              g(i,j)=1;
236 -          end
237 -      end
238 -  end
```

Fonte: Autoria própria

O processo realizado até o aqui descreve o processo de captura e processamento do gabarito que será utilizado como base para correção das provas.

Todo esse processo é repetido quando clicado no segundo botão, “Corrigir Prova”, porém a matriz que representará a prova será a matriz *p[25]10]*.

A matrizes que representam o gabarito e a prova são matrizes de tamanho 25x10, para uma melhor exploração das mesmas, elas são transformadas em matrizes de tamanho 50x5, onde cada linha representa uma questão e as colunas representam as alternativas. Esse processo é mostrado na Figura 23, mostrada a seguir.

Figura 23 – Transformação da matriz

```
for i=1:25
    g_vet(i,1) = g(i,1);
    g_vet(i,2) = g(i,2);
    g_vet(i,3) = g(i,3);
    g_vet(i,4) = g(i,4);
    g_vet(i,5) = g(i,5);
end
for i=26:50
    g_vet(i,1) = g(i-25,6);
    g_vet(i,2) = g(i-25,7);
    g_vet(i,3) = g(i-25,8);
    g_vet(i,4) = g(i-25,9);
    g_vet(i,5) = g(i-25,10);
end

for i=1:25
    p_vet(i,1) = p(i,1);
    p_vet(i,2) = p(i,2);
    p_vet(i,3) = p(i,3);
    p_vet(i,4) = p(i,4);
    p_vet(i,5) = p(i,5);
end
for i=26:50
    p_vet(i,1) = p(i-25,6);
    p_vet(i,2) = p(i-25,7);
    p_vet(i,3) = p(i-25,8);
    p_vet(i,4) = p(i-25,9);
    p_vet(i,5) = p(i-25,10);
end
```

Fonte: Autoria própria

Após a transformação das matrizes, é feita a comparação lógica linha a linha de toda a matriz. A comparação é feita até o número de questões informadas pelo usuário. O Total de acertos é somado na variável *acertos* e as questões corretas são salvas no vetor certas. Essa comparação é mostrada na Figura 24.

Figura 24 – Comparação das matrizes

```
for i=1:numQuestoes
    if ((g_vet(i,1)== p_vet(i,1)) && (g_vet(i,2)== p_vet(i,2)) &&
        (g_vet(i,3)==p_vet(i,3)) && (g_vet(i,4)==p_vet(i,4)) && (g_vet(i,5)==p_vet(i,5)))
        acertos=acertos+1;
        questao=num2str(i);
        if(i==1)
            certas = strcat(certas,questao);
        else
            certas = strcat(certas,hifem,questao);
        end
    end
```

Fonte: Autoria própria

Após a comparação é os valores salvos em *acertos* e *certas* são mostrados na tela pela função *set*, como mostrado na Figura 25.

Figura 25 – Exibe os textos na tela

```
472 -   set(handles.text4,'String',acertos);
473 -   set(handles.text6,'String',certas);
```

Fonte: Autoria própria

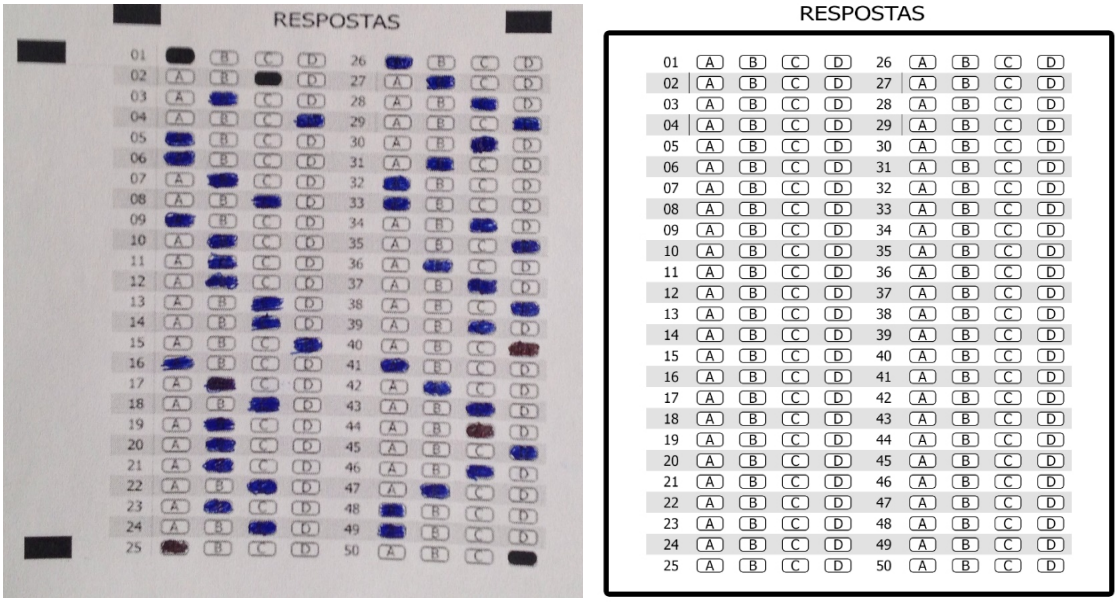
3.2 IMPLEMENTAÇÃO DO CORRETOR PARA DISPOSITIVOS MÓVEIS

O elemento diferencial da proposta de solução é o método de correção através de dispositivos móveis que rodem a plataforma Android.

A implementação apresentada faz uso da capacidade largamente utilizada nos dispositivos móveis de obter imagens digitais. O sistema se baseia na obtenção de uma imagem de um formulário de gabarito impresso, previamente definido e pelo consequente processamento de seu conteúdo.

Como o processo de obtenção e processamento de imagens na plataforma Android se dá de forma diferente do encontrado no MATLAB, foi preciso realizar uma alteração no modelo de gabarito proposto. O modelo de gabarito de utilizado para desenvolvimento no Android foi o mostrado na Figura 26 a seguir.

Figura 26 – Gabarito modelo do protótipo (esquerda) e do Android (direita)

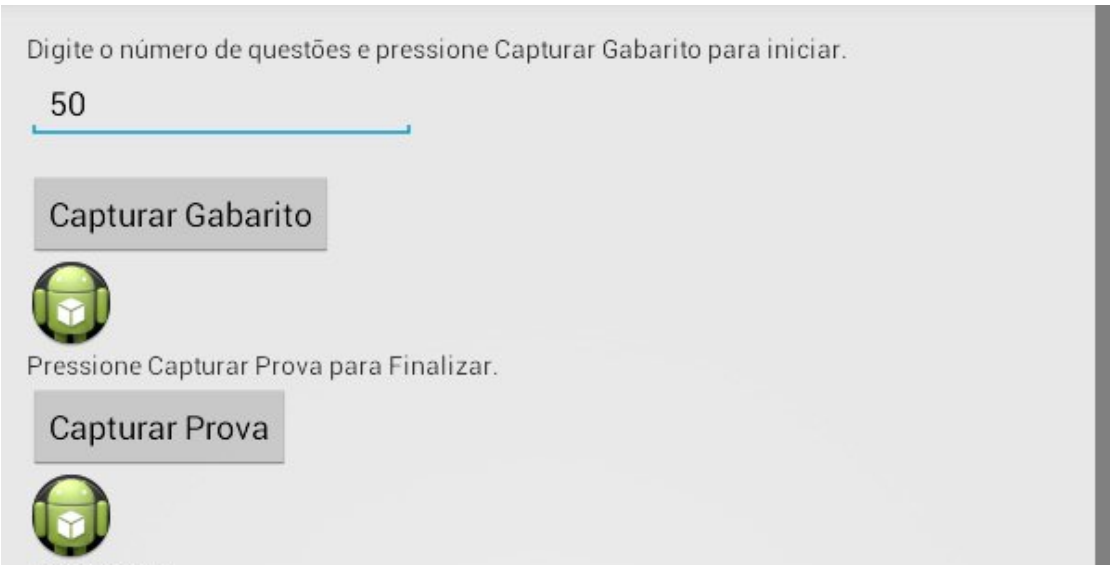


Fonte: Autoria própria

3.2.1 Tela de usuário

Assim como no protótipo desenvolvido no MATLAB, o aplicativo desenvolvido para dispositivos móveis conta somente com uma tela. Essa tela, mostrada na Figura 27, contém um campo onde o usuário informa o número de questão presentes no gabarito, os botões de *Capturar Gabarito* e *Capturar Prova*, e dois *imageView* onde serão exibidas as imagens dos gabaritos capturadas.

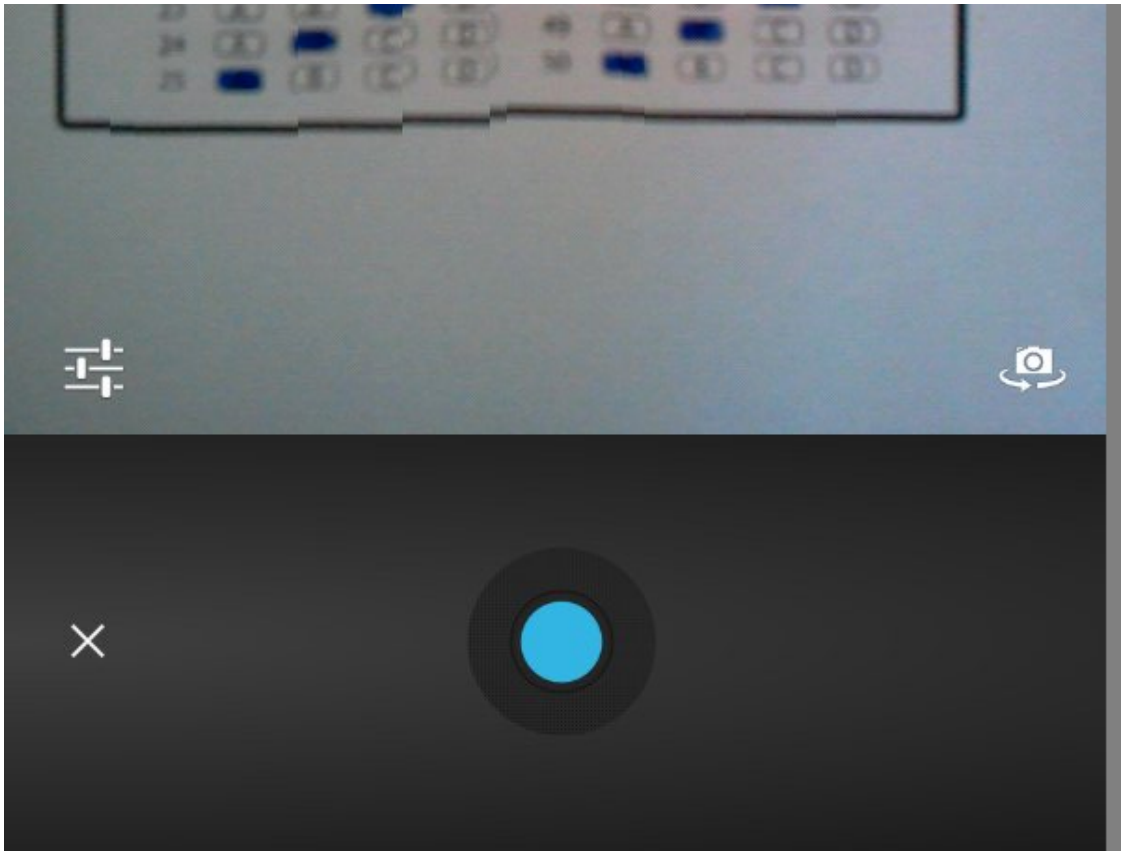
Figura 27 – Tela inicial do aplicativo Android



Fonte: Autoria própria

Ao clicar em um botão é acionada a câmera padrão do dispositivo, como mostra a Figura 28, esse recurso está disponível à todos os aplicativos por meio de *intents*, bastando apenas que seja chamado.

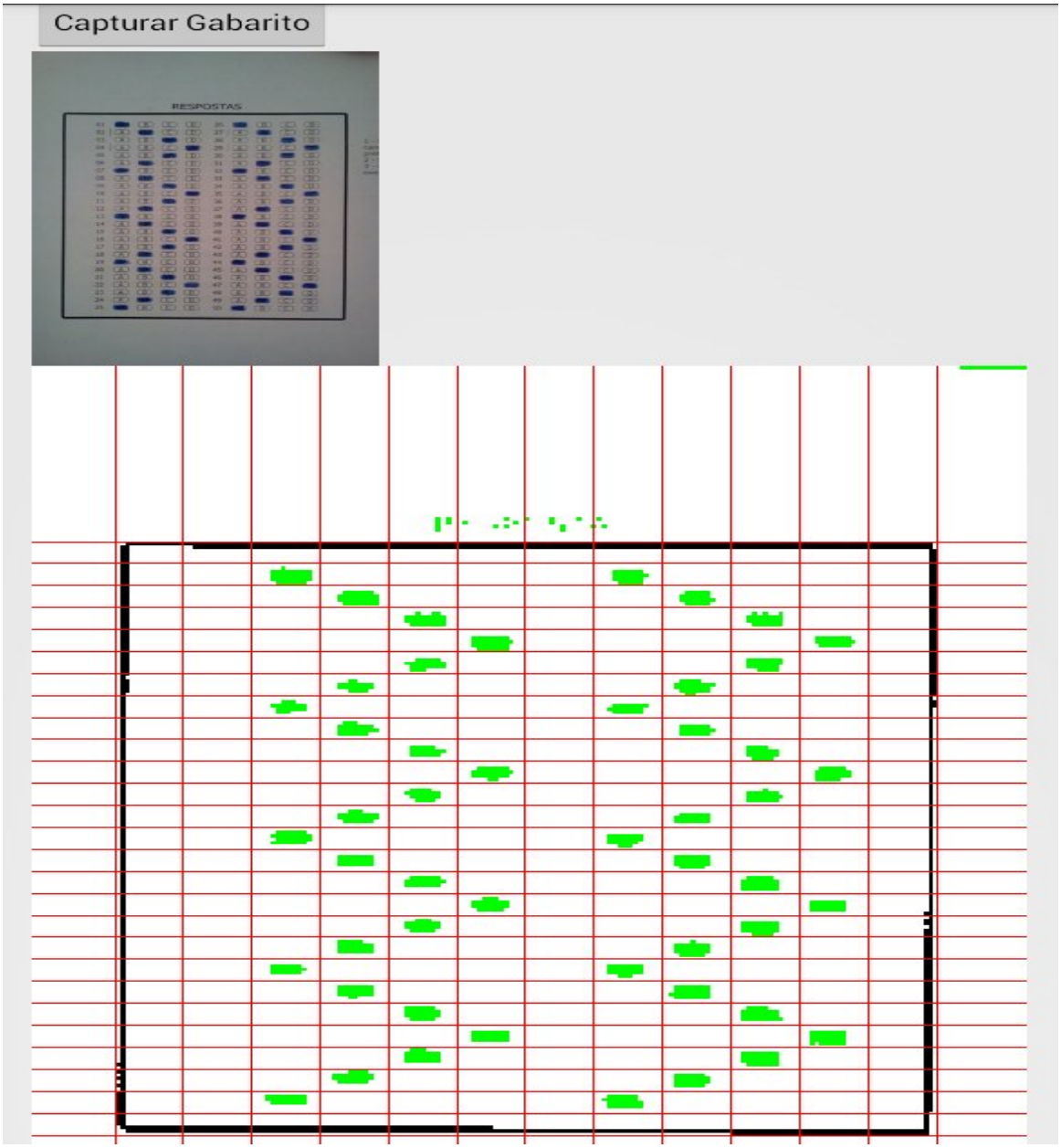
Figura 28 – Câmera do dispositivo Android



Fonte: Autoria própria

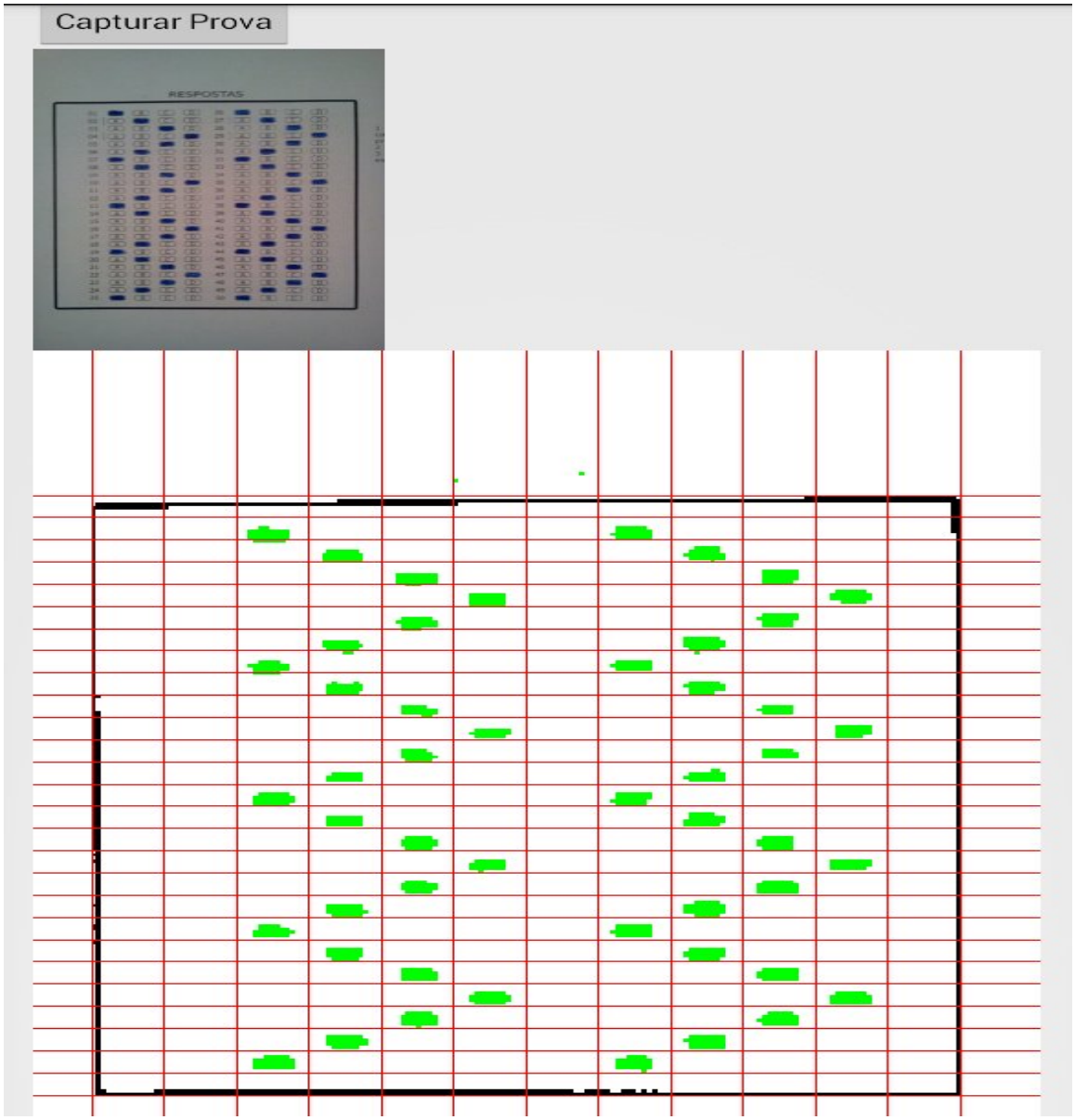
Ao capturar a imagem do gabarito, é exibido na tela tanto a imagem que foi capturada, como um *bitmap* onde são demarcadas as bordas e as células do gabarito e as marcações das questões. A Figura 29 e Figura 30, a seguir, mostram um exemplo de gabarito e de uma prova capturados, respectivamente.

Figura 29 – Gabarito exemplo capturado



Fonte: Autoria própria

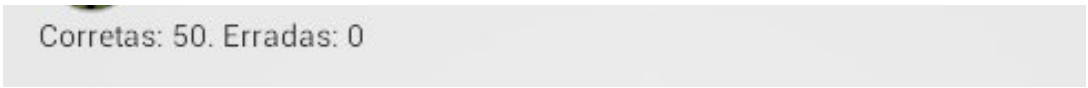
Figura 30 – Prova exemplo capturado



Fonte: Autoria própria

Logo após, como mostra a Figura 31, é exibido o número de acertos e erros da prova.

Figura 31 – Prova exemplo capturado



Fonte: Autoria própria

3.2.2 Trechos de código do aplicativo desenvolvido

Para o desenvolvimento de aplicativos na plataforma Android, é necessário gerar dois tipos de arquivos, os arquivos *.xml*, responsáveis por criar o *layout* do aplicativo, e os arquivos *.java*, que realizam a parte lógica do aplicativo.

O arquivo *activity_main.xml*, por padrão, é o arquivo responsável por gerar a tela inicial do programa, nele é descrito todos os componentes da tela, incluindo suas funções, ids, tipo, alinhamento. Um exemplo presente no aplicativo é mostrado na Figura 32:

Figura 32 – Exemplo de código .xml

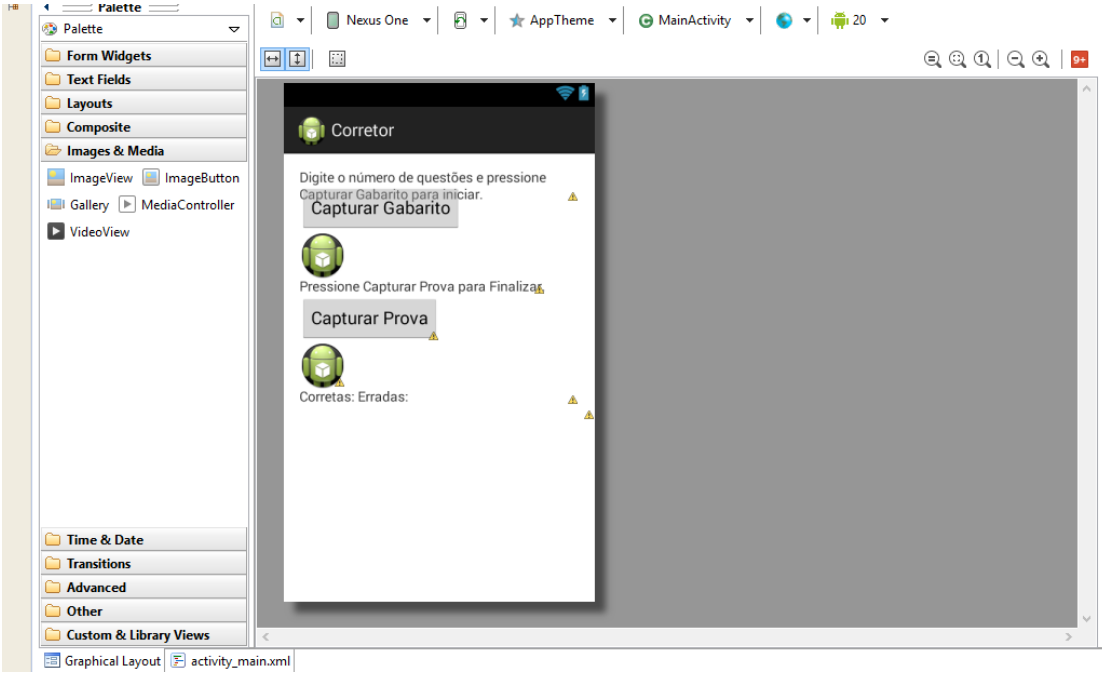
```
<EditText
    android:id="@+id/numQuestoes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/linearLayout"
    android:layout_below="@+id/textView1"
    android:ems="10"
    android:inputType="number" >

    <requestFocus />
</EditText>
```

Fonte: Autoria própria

O *layout* do aplicativo além de poder ser definido através de linhas de código, pode ser elaborada visualmente, através do *drag and drop* (arrastar e soltar), como visto na Figura 33.

Figura 33 – Exemplo de elaboração de layout com drag and drop



Fonte: Autoria própria

Os trechos de código a seguir mostram como a parte lógica do aplicativo foi implementada.

Ao clicar nos botões *Capturar Gabarito* e *Capturar Prova* são chamadas as funções *onTakePhotClicked()* e *onTakePhotoProva()*, respectivamente, como visto na Figura 34. Ambas as funções fazem uma chamada à câmera padrão do dispositivo, por meio de um *Intent*. Um *intent* é uma descrição abstrata de uma operação que deve ser realizada. *Intent* é usado para realizar uma ligação entre o código de diferentes aplicações. Esse recurso é disponibilizado pela plataforma Android, para não ser necessário a implementação de uma câmera própria. Ao chamar um *intent* para fazer uso da câmera padrão do dispositivo, todos os recursos da câmera podem ser utilizados, como flash, foco, timer, e é retornado a imagem capturada, para ser tratada ou salva.

Quando o resultado da câmera é obtido, é enviado pela função *startActivityForResult()* para ser tratado, juntamente com o identificador da função que está enviando.

Figura 34 – Ações dos botões do aplicativo

```
public void onTakePhotoClicked(View v){
    Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);

    startActivityForResult(cameraIntent, CAMERA_RESULT_GABARITO);
}
public void onTakePhotoProva(View v){
    Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);

    startActivityForResult(cameraIntent, CAMERA_RESULT_PROVA);
}
```

Fonte: Autoria própria

O resultado da câmera é recebido pela função *OnActivityResult()* e é verificado sua origem, se o remetente é o botão *Capturar Gabarito,* como visto na Figura 35.

Se confirmado o remetente, ele inicia o processo de tratamento do conteúdo recebido.

Inicialmente é verificado o número de questões que o usuário informou e guardado na variável global *numQuestoes*.

O conteúdo do *intent,* armazenado em *data* é salvo na variável gobImage, esse conteúdo é do tipo bitmap com a resolução 256 x 192. A Imagem original é mostrada na tela no *ImageView imgGabarito*. Em seguida é localizado o *id* do *imageView* onde será exibido o *bitmap* processado. E o *bitmap* é salvo na variável *bitmap* para ser processada.

Figura 35 – Recebe o resultado do gabarito e exibe

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    super.onActivityResult(requestCode, resultCode, data);

    if(requestCode == CAMERA_RESULT_GABARITO){

        numQuestoes =  R.id.numQuestoes;
        gabImage = (Bitmap) data.getExtras().get("data");
        imgGabarito.setImageBitmap(gabImage);
        ImageView imageViewShower = (ImageView) findViewById(R.id.imageView);
        Bitmap bitmap = gabImage;
```

Fonte: Autoria própria

Em seguida o *bitmap* é transformado em binário através da função *createBlackAndWhite(bitmap)*, o resultado da função é salvo na variável *imgBinary* do tipo *Bitmap.* A função extrai a largura (*width)* e altura (*height)* do *bitmap* que recebe como parâmetro. Em seguida cria um *bitmap* com as mesmas características do original (largura e altura), esse *bitmap* será usado para guardar o *bitmap* modificado para servir como retorno da

função. Então é iniciado o processo por todos os pixels, onde é obtido as componentes de cada cor de cada pixels, é feito a média entre as cores para gerar a imagem de tons de cinza.

Se o tom de cinza do pixel for maior que o limiar 70, ele será definido como 255, se menor, como 0. A Figura 36 mostra o processo de transformação da imagem em preto e branco.

Figura 36 – Transformação do bitmap em binária

```
public static Bitmap createBlackAndWhite(Bitmap src) {
    int width = src.getWidth();
    int height = src.getHeight();
    Bitmap bmOut = Bitmap.createBitmap(width, height, src.getConfig());
    int A, R, G, B;
    int pixel;

    for (int x = 0; x < width; ++x) {
        for (int y = 0; y < height; ++y) {
            pixel = src.getPixel(x, y);
            A = Color.alpha(pixel);
            R = Color.red(pixel);
            G = Color.green(pixel);
            B = Color.blue(pixel);
            int gray = (int) (0.2989 * R + 0.5870 * G + 0.1140 * B);

            if (gray > 70)
                gray = 255;
            else
                gray = 0;
            bmOut.setPixel(x, y, Color.argb(A, gray, gray, gray));
        }
    }
    return bmOut;
}
```

Fonte: Autoria própria

Após a transformação do *bitmap* em preto e branco, ele é redimensionado para melhorar a precisão e visualização dos pixels.

O processo de redimensionamento de um *bitmap* se dá por meio da criação de um novo *bitmap* que toma por base o *bitmap* original e escala entre o original e o tamanho desejado para o novo. Como visto na Figura 37.

Figura 37 – Redimensionamento do bitmap

```
public Bitmap getResizedBitmap(Bitmap bm, int newHeight, int newWidth) {

    int width = bm.getWidth();
    int height = bm.getHeight();

    float scaleWidth = ((float) newWidth) / width;
    float scaleHeight = ((float) newHeight) / height;

    Matrix matrix = new Matrix();
    matrix.postScale(scaleWidth, scaleHeight);
    Bitmap resizedBitmap = Bitmap.createBitmap(bm, 0, 0, width, height, matrix, false);

    return resizedBitmap;

    }
```

Fonte: Autoria própria

A localização da área destinada a respostas é feita com a ajuda da biblioteca *cvblob,* que é uma divisão da biblioteca *opencv.* A biblioteca *cvblob* realiza detecções de regiões conectadas em imagem binárias. Essa biblioteca é utilizada para localizar a posição, na imagem da borda exterior do gabarito, a partir desses pontos são calculados pontos interiores e a posição de célula do gabarito.

Ao chamar a biblioteca *cvblob,* como mostrado na Figura 38, é retornado um elemento da classe *blob* que contém uma lista de todas as regiões conectadas (*blobList*), as posições mínimas e máximas de cada região, no vetores *x* e *y (xMinTable, xMaxTable, yMinTable, yMaxTable)* , além da área de cada região (*massTable)*.

Com essas informações é possível saber a localização de cada marcação feita no gabarito, tanto da marcação externa (borda) como das marcações internas (questões marcadas).

Figura 38 – Chamada da biblioteca cvblob

```
BlobDetection blob = new BlobDetection(imgBinary);
Bitmap anotherBitmap = blob.getBlob(imgBinary);
BitmapDrawable bitmapDrawable = new BitmapDrawable(anotherBitmap);
imageViewShower.setBackgroundDrawable(bitmapDrawable);
```

Fonte: Autoria própria

A borda é o elemento de maior área quando é capturado o gabarito, pois ele engloba toda a região destinadas as respostas marcadas. Assim se pode saber a sua localização na imagem, como visto na Figura 39.

Figura 39 – Localizando a borda

```
int posicaoBorda = blob.retornaMaior();
int xminGab = blob.xMinTable[posicaoBorda];
int xmaxGab = blob.xMaxTable[posicaoBorda];
int yminGab = blob.yMinTable[posicaoBorda];
int ymaxGab = blob.yMaxTable[posicaoBorda];
```

Fonte: Autoria própria

A função *retornaMaior()* retorna o elemento de maior área da lista, essa função foi implementada com o objetivo de encontrar a posição da borda na lista de regiões. A implementação da função é mostrada na Figura 40.

Figura 40 – Função que encontra a posição da borda

```
    public int retornaMaior(){
        int maximoBlob = 0;
        int posicion = 0;
        for (int i = 0;i<massTable.length;i++) {
            if ( massTable[i] > maximoBlob ) {
                posicion = i;
                maximoBlob = massTable[i];
            }

        }
        return posicion;

    }
}
```

Fonte: Autoria própria

Em seguida é feita a divisão das células através das funções *divisaoX()* e *divisão()*. Essas funções dividem o vetor *x* em 12 células, 10 correspondente às alternativas das questões, e o vetor *y* em 27, 25 correspondente as questões, já o modelo de gabarito dispõe de 50 questões organizadas da 1 a 25 de forma sequencial em linhas e da 26 a 50 situadas ao lado das 25 primeiras.

A Figura 41 e a Figura 41 mostram a implementação das funções *divisaox()* e *divisaoY()*, respectivamente.

Figura 41 – divisaoX()

```
public float[] divisaoX(int xmin, int xmax){
    float[] linha = new float[13];

    float tamanho = xmax-xmin;
    float largura = tamanho/12;
    linha[0] = xmin;


    for(int i=1; i<13;i++){
        linha[i] = linha[i-1]+largura;
    }

    return linha;
}
```

Fonte: Autoria própria

Figura 42 – divisaoY()

```
public float[] divisaoY(int ymin, int ymax){

    float[] coluna = new float[28];
    float tamanho = ymax-ymin;
    float altura = tamanho/27;
    coluna[0] = ymin;

    for(int i=1; i<28;i++){
        coluna[i] = (float) (coluna[i-1]+altura);
    }

    return coluna;
}
```

Fonte: Autoria própria

Após a divisão da imagem em células, faz-se a verificação se existe uma região conectada dentro de cada célula, comparando-se os valores mínimos e máximos dos vetores *x* e *y* de cada região com as células, e por fim, comparando se a área da região é suficientemente grande para caracterizar uma marcação de questão, ou se é apenas risco não intencional. Se existir uma região conectada dentro de uma célula, essa célula recebe o valor 1, enquanto as outras recebem o valor 0. Esses valores serão utilizados para comparar o gabarito com a prova. Esse processo é visto na Figura 43, a seguir.

Figura 43 – Verifica se existe região conectada dentro da célula

```
for(int j=0;j<27;j++){
    for(int k=0;k<12;k++){
        for(int i=0;i<blob.blobList.size();i++){
            if(((xmaxGabVet[i]-xminGabVet[i])/2)>linhaGab[k]){
                if(((xmaxGabVet[i]-xminGabVet[i])/2)<linhaGab[k+1]){
                    if(((ymaxGabVet[i]-yminGabVet[i])/2)>colunaGab[j]){
                        if(((ymaxGabVet[i]-yminGabVet[i])/2)<colunaGab[j+1]){
                            if(massGabVet[i]>30)
                                matrizGab[j][k] = 1;
                        }
                    }
                }
            }
        }
    }
}
```

Fonte: Autoria própria

Todo o processo realizado até aqui descreve a captura e processamento do gabarito que será utilizado como base para correção das demais provas.

O processo de obtenção e processamento da prova a ser corrigida é similar ao utilizado no gabarito, porém no quando verificado se existe uma região conectada dentro de cada célula, o resultado será gravado na matriz *matrizPro[ ][ ]*.

As matrizes que armazenam se existem regiões conectadas dentro de cada célula, são matrizes de dimensões 27 x 12. A primeira e a última linha, armazenam uma linha, assim como a primeira e última coluna, armazenam valores 0, pois nela estão representados os espaços que existem da borda, até a região destinada a marcação das alternativas.

Assim como feito no protótipo implementado no MATLAB, as matrizes 27 x 12 são transformadas em matrizes 50 x 5 onde as linhas representam as questões e as colunas representam as alternativas, como mostra a Figura 44.

Figura 44 – Transforma as matrizes em 50 x 5

```
int[][] gabMarcados = new int[50][5];
int[][] proMarcados = new int[50][5];

for(int j=1;j<26;j++){
    for(int k=1;k<6;k++){
        gabMarcados[j-1][k-1] = matrizGab[j][k];
    }
}
for(int j=1;j<26;j++){
    for(int k=6;k<11;k++){
        gabMarcados[j+24][k-6] = matrizGab[j][k];
    }
}

for(int j=1;j<26;j++){
    for(int k=1;k<6;k++){
        proMarcados[j-1][k-1] = matrizPro[j][k];
    }
}
for(int j=1;j<26;j++){
    for(int k=6;k<11;k++){
        proMarcados[j+24][k-6] = matrizPro[j][k];
    }
}
```

Fonte: Autoria própria

Após a obtenção das duas matrizes em dimensões 50 x 5, é feita a comparação entre as matrizes *matrizGab* e *matrizPro*, como mostra a Figura 45. O número de questões corretas é armazenado na variável *corretas*. O número de questões certas e erradas são exibidas na tela pelo *textView* utilizando a função *setText()*.

Figura 45 – Comparação das matrizes

```
for(int j=0;j<numQuestoes;j++){
    teste =1;
    for(int k=0;k<5;k++){
        if(proMarcados[j][k]==gabMarcados[j][k]){
        }
        else
            teste=0;
    }
    if(teste == 1){
        proCertas[j] = 1;
        corretas++;
    }
}
textViewInfo.setText("Corretas: "+corretas+". Erradas: "+(50-corretas)+" blobs:\n");
```

Fonte: Autoria própria

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos com os testes realizados após a implementação do protótipo desenvolvido com o MATLAB e com a implementação do aplicativo Android.

O protótipo foi desenvolvido utilizando o MATLAB e as capturas de imagens foram realizadas utilizando a Microsoft Webcam LifeCam HD 3000, que utiliza a resolução 1280 x 720.

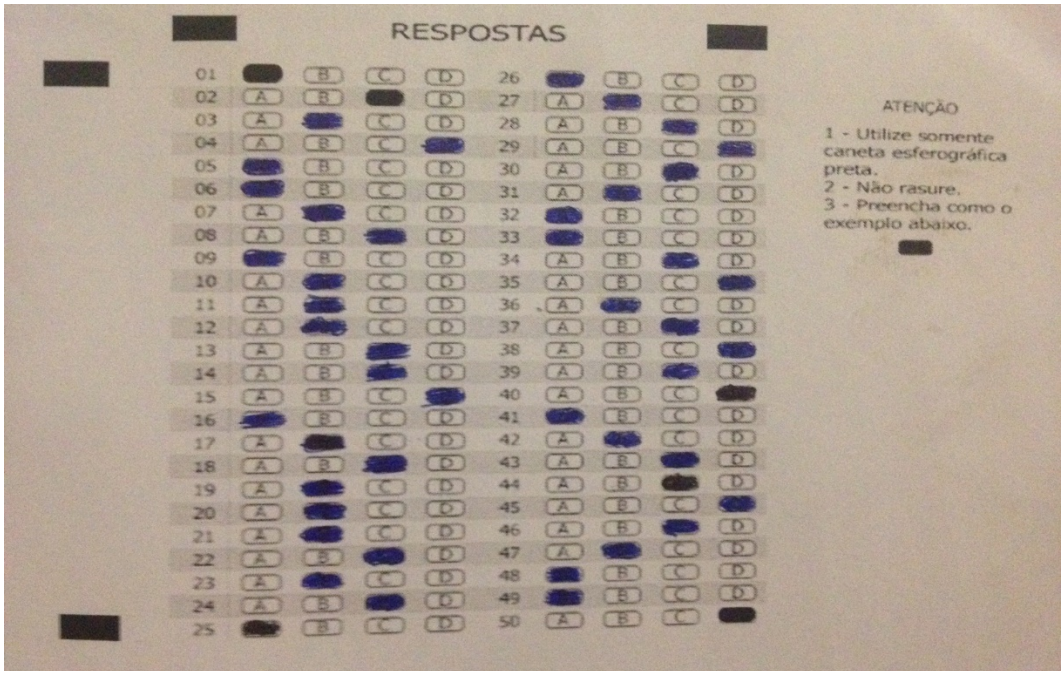
O aplicativo Android foi desenvolvido utilizando a IDE Eclipse Juno incorporando os plug-ins de desenvolvimento Android com o auxilio da biblioteca *cvblob* pertencente à biblioteca *opencv*. O aplicativo foi testado utilizando o Samsung Galaxy Tab 7" rodando o sistema operacional Android 4.1.

Tanto no protótipo como no Aplicativo Android desenvolvido, foram encontradas dificuldades de localizar a área destinada para as respostas, mas rapidamente foram encontradas soluções, no MATLAB, o uso da função *bwboundaries*, e no Android o uso da biblioteca *opencv*. Em ambos os casos também foi encontrado problema com relação a configuração de iluminação necessária para a captura sem falhas do gabarito e da prova. O problema foi sanado realizando-se testes em várias intensidades de luz e ajustando-se o *threshold* de luminância. Nenhuma das funções chegou ao final dos testes apresentando erros críticos.

4.1 EXEMPLO DE TESTE UTILIZANDO O PROTÓTIPO EM MATLAB

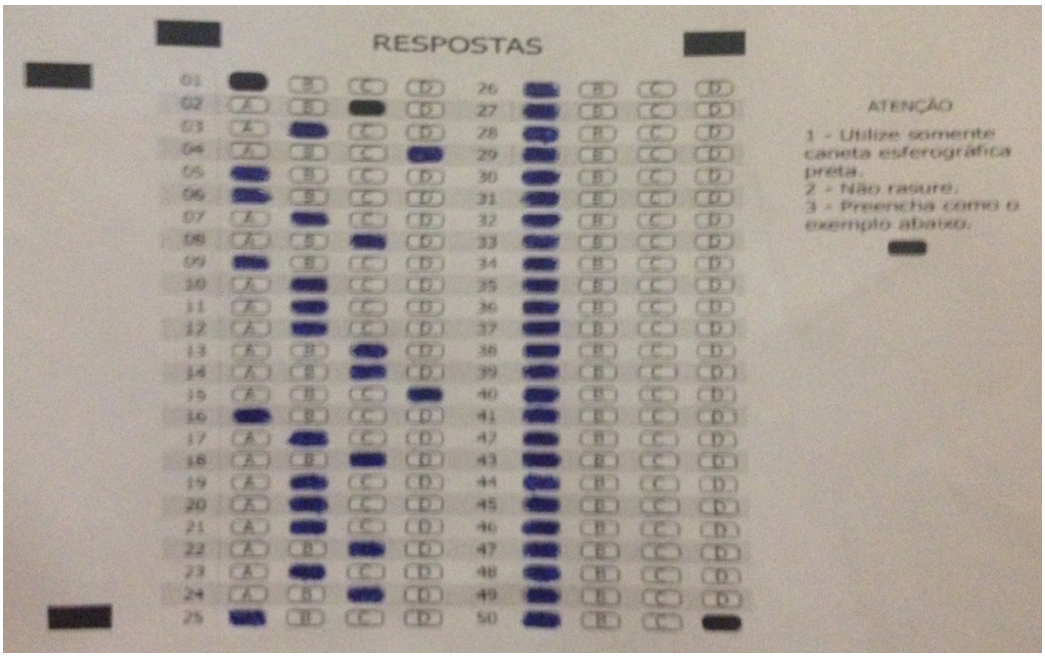
As figuras a seguir, mostram o exemplo que será testado com o protótipo desenvolvido em MATLAB. A Figura 46 e a Figura 47 mostram o gabarito que servirá de base e a prova à ser corrigida, respectivamente.

Figura 46 – Gabarito base para teste do protótipo



Fonte: Autoria própria

Figura 47 – Prova a ser corrigida no teste do protótipo

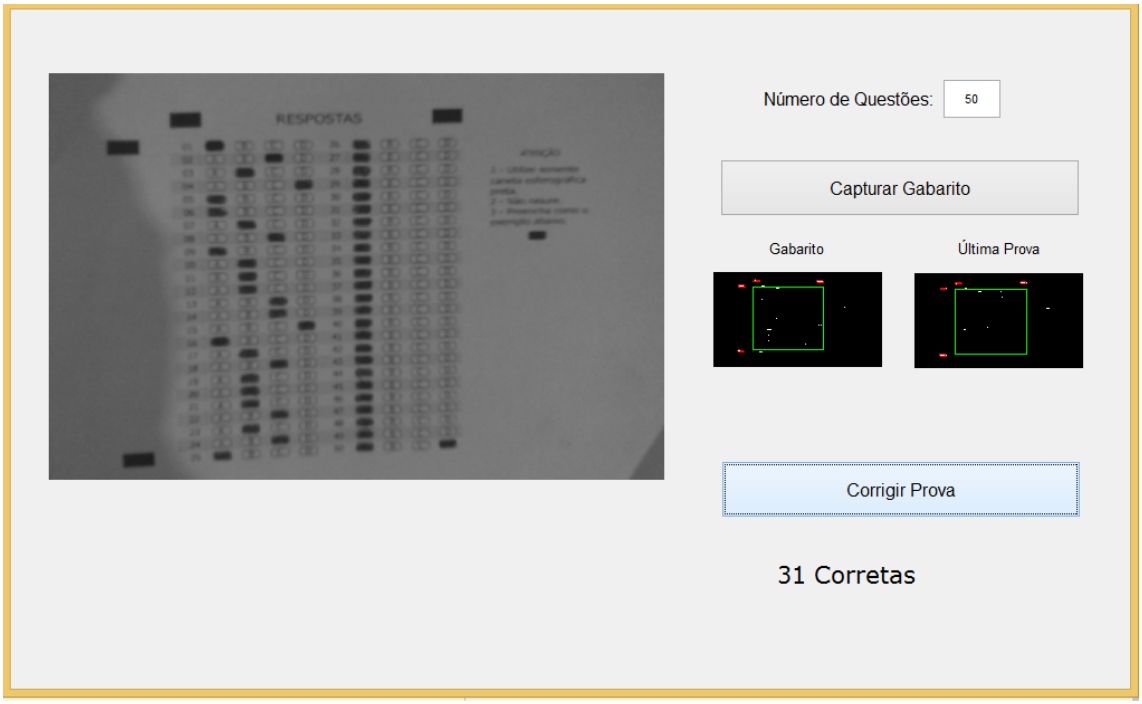


Fonte: Autoria própria

Vale observar que a primeira parte da prova (questões de 1 a 25) está completamente correta, enquanto a segunda parte (questões de 26 a 50) estão todas marcadas a alternativa ‘a’, e na questão 50 estão marcadas duas alternativas (‘a’ e ‘d’), o que faz com que a questão seja considerada errada. Com isso o número de questões certas é 31.

Com a realização do teste, o resultado obtido foi 31, o resultado esperado, como mostrado na Figura 48, o protótipo foi capaz de processar as questões corretas, as erradas e o caso de existir duas alternativas marcada em uma mesma questão.

Figura 48 – Teste do protótipo



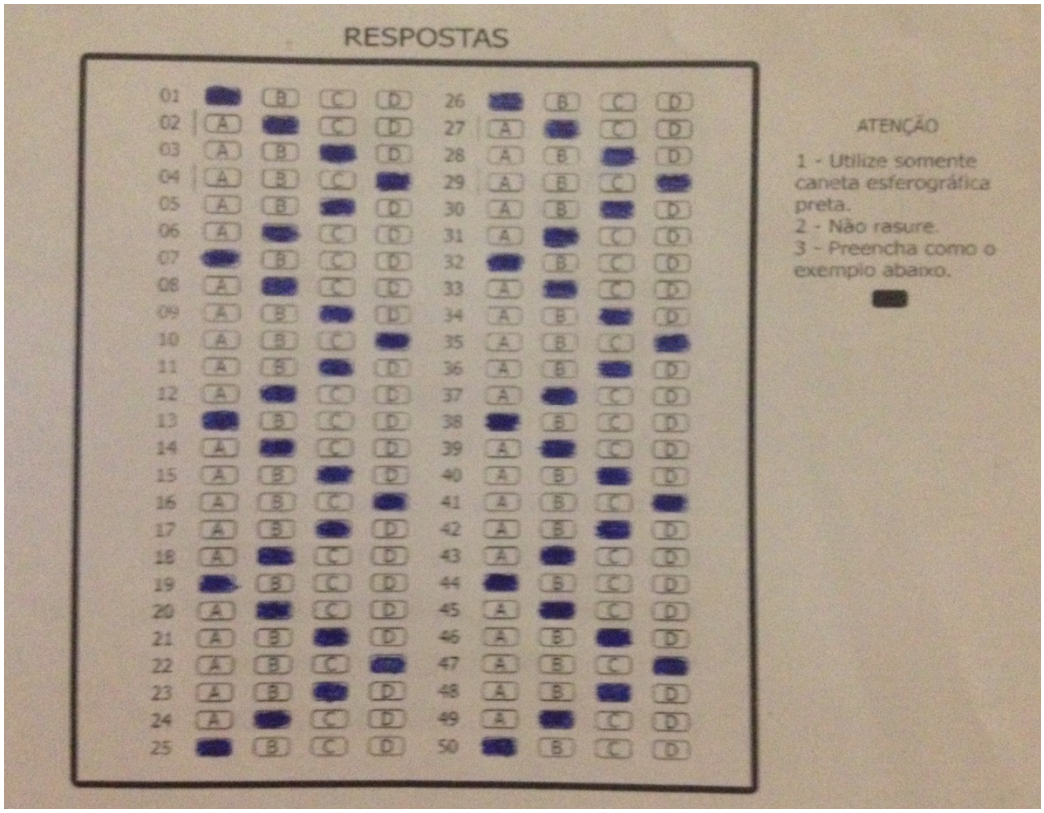
Fonte: Autoria própria

4.2 EXEMPLO DE TESTE UTILIZANDO APLICATIVO ANDROID

Para realização do teste do aplicativo Android, é necessário compilar o projeto e rodar em um dispositivo móvel.

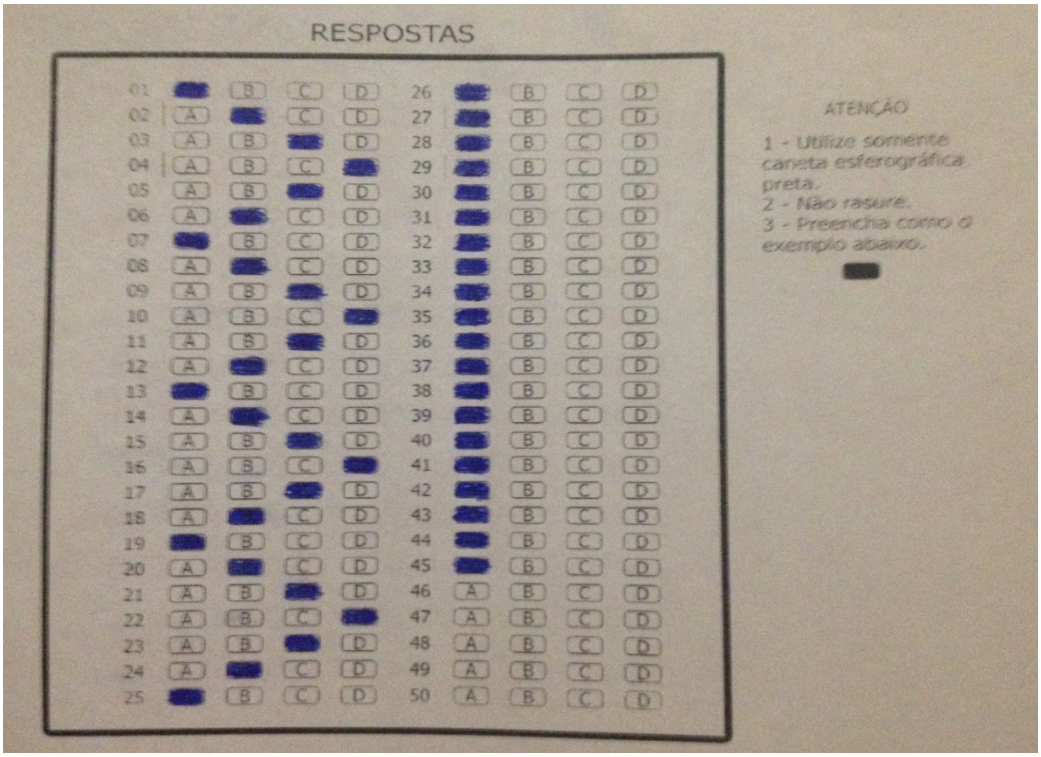
A Figura 49 e a Figura 50, a seguir, mostram o gabarito base e a prova a ser corrigida no teste efetuado no aplicativo.

Figura 49 – Gabarito base usado no teste do aplicativo Android



Fonte: Autoria própria

Figura 50 – Prova a ser corrigida pelo aplicativo Android

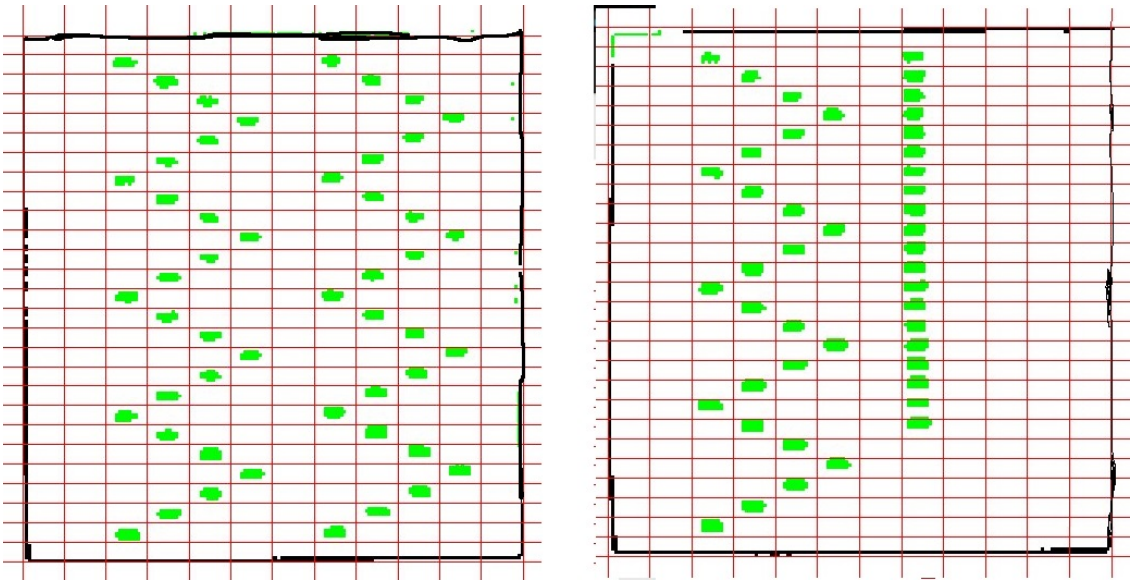


Fonte: Autoria própria

Observa-se que a primeira parte da prova (questões de 1 a 25) está completamente correta, enquanto a segunda parte (questões de 26 a 45) estão todas marcadas a alternativa ‘a’, e da questão 46 a questão 50 estão em branco, o que faz com que a questão seja considerada errada. Com isso o número de questões certas é 29.

A Figura 51 mostra o gabarito (direita) e a prova corrigida (esquerda) no teste do aplicativo Android. A Figura 52 mostra o resultado apresentado pelo aplicativo.

Figura 51 – Gabarito e prova corrigida pela aplicativo Android



Fonte: Autoria própria

Figura 52 – Resultado apresentado pelo aplicativo

Fonte: Autoria própria

Após a realização de testes, os resultados dos códigos fontes foram considerados satisfatórios.

5 CONCLUSÕES

Com a realização deste trabalho foi possível aplicar diversos conhecimentos adquiridos no decorrer da graduação, bem como a apreensão de muitos outros novos e o aprofundamento do conhecimento nas linguagens de programação utilizadas com todas as suas características e funcionalidades.

O software ainda não se encontra em uma versão final, sendo necessário que ocorra melhoramento de algumas funcionalidades como interface mais atraente e intuitiva, realização de mais testes, além da adição de funcionalidades que farão com que o software possa ser consolidado e que traga uma garantia de serviço a longo prazo ao usuário.

Com os resultados obtidos neste estudo é possível concluir que, no que diz respeito ao sistema, ele foi especificado, desenvolvido, testado em ambiente e comprovado sua eficiência, pois corrigiu provas de forma simples, ágil e cumprindo sua função de ajudar corretores de provas objetivas.

Portanto, pode-se concluir que os objetivos a que este trabalho se propôs foram atingidos, pois foi desenvolvido, um software que trabalhará com elementos de obtenção de imagens e processamento digital de imagens dos gabaritos impressos.

REFERÊNCIAS

ALBUQUERQUE, Márcio Portes de; ALBUQUERQUE, Marcelo Portes de. Processamento e Imagens: Métodos e Análises. Rio de Janeiro: Centro Brasileiro de Pesquisas Físicas – BPF/MCT, 2009. 12 p.

ANDROID | OpenCV. Disponível em:< http://opencv.org/platforms/android.html/>. Acesso em: 05 jul. 2014.

AQUINO, Juliana França Santos. Introdução à Computação Móvel. 2007. 14 f. Monografia – Programa Pós Graduação em Informática, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

BARROS, Thiago. Android Studio é o programa do Google para desenvolver apps para Android. Disponível em: <http://www.techtudo.com.br/tudo-sobre/s/android-studio.html>. Acesso em: 11 jun. 2014.

BLOOM, B. S., HASTINGS, J. T. & MADAUS, G. F. Manual de Avaliação Formativa e Somativa do Aprendizado Escolar. São Paulo, Livraria Pioneira Editora, 1983.

CVBLOB – Blob library for opencv. Disponível em:< https://code.google.com/p/cvblob/>. Acesso em: 06 jul. 2014.

CHUEIRI.M.S.F. Concepções sobre a Avaliação Escolar. v19.n39. PUC-MG. 2008.

FARTO, Guilherme de Cleva. Abordagem Orientada a Serviços para implementação de um aplicativo Google Android. Trabalho de Conclusão de Curso Assis: Fundação Educacional do Município de Assis - FEMA, 2010. 83 p.

LECHETA, Ricardo R. Google Android: Aprenda a criar aplicações para dispositivos móveis com Android SDK. 3. ed. São Paulo: Novatec, 2013. 576 p.

MATLAB GUI. Disponível em:< http://www.mathworks.com/discovery/matlab-gui.html>. Acesso em: 10 jun. 2014.

MEDNIEKS, Zigurd et al. Programando Android: Programando Java para a nova geração de dispositivos móveis. 2. ed. São Paulo: Novatec, 2012. 116 p.

NETO, Hugo Vieira; FILHO, Oge Marques. Processamento Digital de Imagens. Rio de Janeiro: Brasport, 1999. 330 p.

NOBRE, Marcelo. **O uso do software matlab para o estudo de alguns tópicos de álgebra linear**. Disponível em: <http://www.ucb.br/sites/100/103/TCC/22005/MarcelloNobreCardoso.pdf>. Acesso em: 16 jun. 2014.

OPENCV: OpenSource Computer Vision. Disponível em: <http://opencv.org/>. Acesso em: 14 jun 2014.

PEREIRA, Lucio Camilo de Oliveira; SILVA, Michel Lourenço da. Android para desenvolvedores. Rio de Janeiro: Brasport, 2009. 221 p.