GABRIEL AKIRA YANAGUYA

**DESENVOLVIMENTO DE APLICATIVO ANDROID PARA SÍNTESE DE VOZ DE TEXTOS EXTRAÍDOS DE IMAGENS POR MEIO DE OCR**

Projeto de pesquisa apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, *campus* São João da Boa Vista, como requerimento parcial para fins de aprovação da disciplina Metodologia de Pesquisa Científica. Orientador: Prof. Gustavo Aurélio Prieto. Coorientador: Prof. Ricardo Alexandre Neves

SÃO JOÃO DA BOA VISTA

2017

**RESUMO**

YANAGUYA, G. A. (2017). **Desenvolvimento de aplicativo Android para síntese de voz de textos extraídos de imagens por meio de OCR.** Artigo Cientifico - Instituto Federal de São Paulo, São João da Boa Vista, 2017.

A tecnologia OCR tem apresentado um avanço considerável nos últimos anos, estando presente em softwares de digitalização e catálogo de documentos, tradução automática e até no auxílio à leitura para portadores de deficiência visual.

Entretanto, a maioria dos softwares OCR possuem arquitetura projetada para processamento de imagens escaneadas, apresentando precisão e desempenho inferiores quando aplicados ao processamento de fotografias.

Este projeto tem como principal objetivo a proposta de um modelo de arquitetura de software para aplicação em desenvolvimento de aplicativos móveis para síntese de voz de textos extraídos de imagens obtidas a partir de câmera fotográfica de smartphones e tablets, determinando aspectos que influenciam negativamente na precisão do OCR, bem como aplicando intervenções que minimizam os efeitos desses aspectos.

Afim de avaliar a eficácia do modelo, também foram adotadas uma métrica para avaliação dos resultados e um teste estatístico que permite comparar resultados de extração de caracteres por dois aplicativos distintos.

De maneira geral, os resultados obtidos comprovam a eficácia do modelo em cenários onde a imagem não apresenta fundo complexo.

Palavras-chave: OCR. Tesseract OCR. Text-To-Speech. Android.

# 1 LINHA DE PESQUISA

Algoritmos e Processamento de Imagens;

Desenvolvimento para Dispositivos Móveis;

Acessibilidade e Tecnologias Assistivas;

Text Signage Recognition.

# 2 TEMA E SUA DELIMITAÇÃO

Este trabalho visa a análise e elaboração de um modelo de arquitetura para viabilizar a aplicação de síntese automática, por meio do uso dos recursos de voz e textos extraídos de fotografias fazendo uso de OCR[[1]](#footnote-1), bem como aplicar o referido modelo como ferramenta de auxílio à leitura para portadores de deficiência visual ou àqueles que possam apresentar dificuldades, tais como grau de analfabetismo na leitura de textos.

# 3 JUSTIFICATIVA

A tecnologia OCR é uma importante ferramenta para a construção de softwares de acessibilidade no que tange a construção de aplicações de auxílio à leitura para portadores de deficiência visual, justificada pela baixa precisão[[2]](#footnote-2) no reconhecimento óptico dos caracteres, em imagens capturadas por câmeras fotográficas de dispositivos móveis, destacada pelo obstáculo a ser transposto. Atualmente, a maioria dos softwares OCR possuem desempenho consideravelmente maior quando aplicados no processo de extração de caracteres em documentos de texto escaneados, apesar de apresentar queda de precisão considerável quando submetida, tal extração de caracteres, em fotografias em ambientess abertos (REDIG, 2012; RESNIKOFF, 2002; RUSIÑOL, 2014).

# 4 OBJETIVOS

Este trabalho apresenta como objetivo geral uma proposta de um modelo de arquitetura para aplicação em soluções de síntese de voz de textos, obtidos a partir de imagens, por meio de OCR. A solução aplicada neste trabalho, diante da proposta de arquitetura apresentada, funciona como ferramenta de acessibilidade para auxílio de leitura para indivíduos portadores de deficiência visual ou àqueles que apresentem algum grau de analfabetismo.

Os objetivos específicos desta pesquisa são:

* Aplicação de intervenções (pré-processamento de imagem) para melhoria da precisão do Tesseract OCR[[3]](#footnote-3);
* Desenvolvimento de um aplicativo baseado no modelo proposto, afim de coletar dados para validação do modelo;
* Adoção de uma métrica de mensuração de resultados de processamento OCR, bem como estruturação de testes, de acordo com a métrica;
* Proposta de uma metodologia comparativa, baseada em experimentos práticos, afim de mensurar, analisar e documentar a influência das intervenções nos resultados obtidos.

# 5 FUNDAMENTAÇÃO TEÓRICA

## 5.1 Tecnologia AAssistiva para auxílio à leitura

Tecnologia Assistiva (TA) é definida como o conjunto de recursos e serviços que contribuem para ampliar ou proporcionar habilidades funcionais de indivíduos portadores de deficiências, de modo a facilitar a inclusão e independência aos seus utilizadores. Dentre as diversas aplicações da TA este trabalho apresenta aspectos destas tecnologias de auxílio, para suporte da habilidade visual e à recursos que ampliam a capacidade de obtenção de informação destas pessoas com baixa visão ou cegas, utilizando-se de sintetização voz de textos impressos, por meio de TTS e reconhecimento óptico de caracteres (BERSCH, 2008).

## 5.2 TTS

TTS, sigla para *Text-To-Speech*, é uma tecnologia que permite converter texto digital em voz, sintetizando a fala correspondente ao texto de entrada (LECHETA, 2013).

O processo de síntese de voz via TTS é definido a partir dos seguintes passos, segundo ADDISON et. al., 2005 e TATHAM, 2005:

1. A entrada de um dado texto;
2. A normalização do texto recebido;
3. A detecção de derivações de pronúncia;
4. A conversão da representação ortográfica para a representação fonológica;
5. O *parsing* semântico e gramatical[[4]](#footnote-4);
6. A associação de uma sequência fonológica à cada unidade prosódica;
7. A geração do sinal de fala (saída de áudio).

A tecnologia TTS vem sendo amplamente empregada na construção de softwares de acessibilidade, permitindo a narração automática de páginas web, softwares e documentos digitais. Diversos sistemas operacionais modernos, como as versões atuais do Microsoft Windows, o Android e o IOs já possuem os recursos de TTS disponíveis como configuração de fábrica (WHITE, 2015).

## 5.3 OCR

*OCR* (*Optical Character Recognition*) é uma tecnologia empregada na detecção, identificação e extração de caracteres, a partir de um arquivo de imagem, ou seja, na obtenção de texto editável a partir de textos em formato de imagem, seja ela escaneada, escrita à mão, fotografada, datilografada ou impressa (VERMA, 2016).

Existem diversos softwares OCR disponíveis, como o *GOCR*, a *Tesseract* *Engine*, *ABBYY*, *Findreader*, entre outros. Entre as opções, a *Tesseract Engine* destaca-se por ser gratuita e *open-source* epor permitir o treinamento do software para reconhecimento de qualquer linguagem ou tipos de caracteres, bem como pela velocidade e precisão dos resultados. (BHASKAR, 2010; CHATTOPADHYAY, 2011)

### 5.3.1 Tesseract OCR

A *Tesseract Engine* é um *software* de reconhecimento óptico de caracteres *open-source* com distribuição gratuita desenvolvido pela Hewlett Packard (HP) entre 1984 e 1994. Em 1995, ainda de propriedade da HP, foi considerada um dos três *softwares OCR* com maior precisão e desempenho. Em 2006, o projeto recebeu patrocínio oficial da Google que, desde então, torna-se responsável pela manutenção e distribuição do *software*, sob licença Apache 2.0 (SMITH, 1995).

Inicialmente desenvolvido na linguagem C e posteriormente migrado para C++, a Tesseract está atualmente disponível para Windows, Linux e Mac OS X e pode reconhecer 38 idiomas, como inglês, alemão, chinês, italiano, japonês, russo, espanhol e outros, podendo ser treinado para reconhecer caracteres de qualquer idioma ou fonte, incluindo textos manuscritos (SMITH, 1995). Segundo Sonia Bhaskar (2010), a *Tesseract* é considerada o *software OCR* gratuito mais preciso em operação.

## 5.4 Precisão dos resultados

A *Tesseract Engine,* assim como a maioria dos softwares *OCR*, foi desenvolvida para extração de caracteres, a partir de análise de documentos escaneados. Quando utilizada em fotos capturadas por um dispositivo móvel, alguns fatores, grande parte relacionados à natureza do ambiente, influenciam na captura da imagem, causando distorções e ruídos que podem resultar em perda de precisão no processo de extração. Dentre tais fatores destacam-se a iluminação, as distorções de perspectiva, o contraste entre o texto e o resto do ambiente, o foco da imagem e a possibilidade da presença de objetos que podem ser confundidos com caracteres ou até mesmo linhas de texto (RUSIÑOL, 2014; YANAGUYA, 2015).

### 5.4.1 Processamento Básico

A *Tesseract Engine* executa diversas operações de processamento de imagem utilizando a biblioteca *open source* Leptonica, antes de realizar o processo de extração e reconhecimento de caracteres. Apesar deste processamento se apresentar eficiente em muitos casos, existem cenários onde tal processamento não atinge o resultado esperado. Então, nestes casos, há redução significativa na precisão**.** Assim, existem operações que, quando executadas antes da submissão da imagem ao *Tesseract*, auxiliam na redução dos problemas dos processos internos da API[[5]](#footnote-5), são eles:

* Redimensionamento da imagem: o *Tessetact* apresenta resultados melhores em imagens com densidade 300dpiou superior, sendo benéfico o redimensionamento da imagem, afim de atingir essa característica; e
* Conversão da imagem em *grayscale*: o processo de binarização de Otsu[[6]](#footnote-6), utilizado no Tesseract, funciona de forma mais eficiente em imagens em tons de cinza **(**GITHUB.COM, 2017).

### 5.4.2 Filtros de Luminosidade e Contraste

O *GPUImage* é um *framework* de processamento de imagens *open source* desenvolvido por Brad Larsonutilizando a biblioteca *OpenGL ES 2.0*, projetado inicialmente para *iOS*. Esse *framework* permite a aplicação de diversos filtros a imagens e vídeos. Os filtros de luminosidade e contraste presentes no *GPUImage* auxiliam na correção de distorções de luminosidade provenientes do processo de captura, bem como auxiliam na separação morfológica entre os objetos e o fundo comum da imagem.

### 5.4.3 Remoção de ruídos

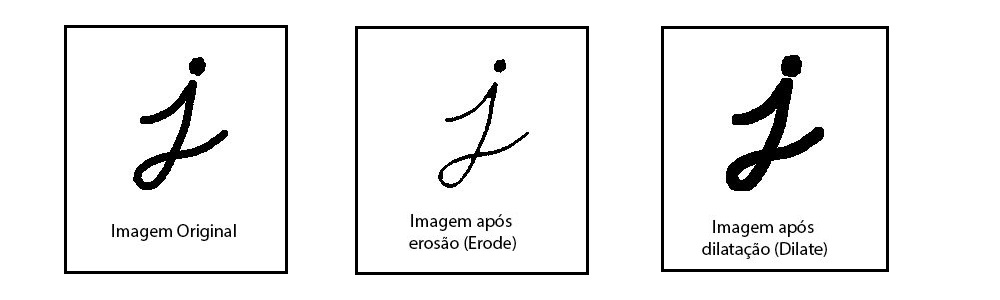
O ruído é definido como uma variação aleatória (adição, alteração ou remoção) de informações em uma dada imagem. Pequenos ruídos são detectados e removidos, automaticamente, pelo processo de binarização. Entretanto, ruídos de maior intensidade podem ser equivocadamente classificados como possíveis caracteres na etapa de binarização.

Portanto, afim de aprimorar a precisão de softwares *OCR*, utiliza-se operações de processamento de imagens com o objetivo de reduzir a ocorrência de ruídos em uma imagem, como as operações Erode e Dilate da *OpenCV**[[7]](#footnote-7)*.

Erode e Dilate são duas operações morfológicas, disponíveis na biblioteca *open source* OpenCV, que processam uma imagem com base em formas. Ambas podem ser aplicadas para remoção de ruído, isolamento de elementos individuais e junção de elementos diferentes em uma imagem. Essas operações permitem que componentes não-desconexos expandam sua área, ocupando espaços de componentes desconexos, minimizando-os ou até mesmo os eliminando por completo da imagem (OPENCV.ORG).

A Figura 1 ilustra a aplicação das operações Erode e Dilate por meio da OpenCV:

**Figura 1- Aplicação das operações Erode e Dilate da OpenCV**



*Fonte: Elaborada pelo autor*

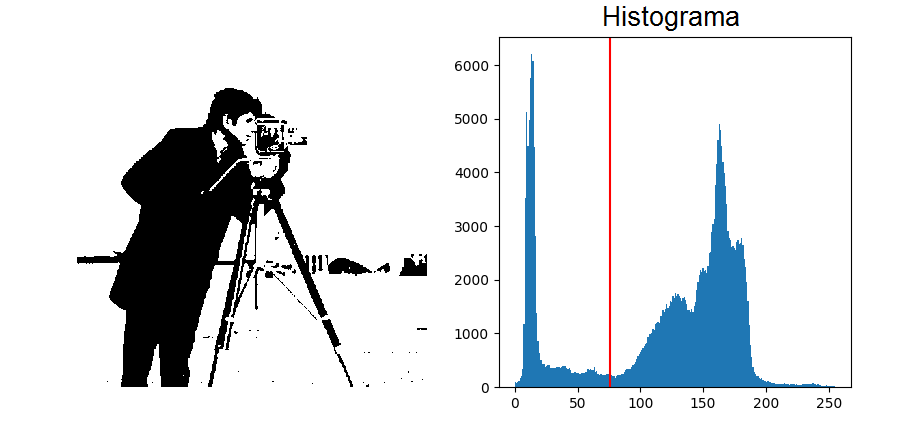
### 5.4.4 Binarização e Algoritmo de limiarização de OTSU

A binarização ou limiarização é o processo de segmentação de uma imagem, de acordo com a contextualização de seu conteúdo, no qual consiste na subdivisão da imagem em regiões ou objetos distintos. O processo de binarização permite a conversão de uma imagem *grayscale* em uma imagem binária (preto e branco), de forma a separar em duas classes diferentes objetos e regiões significativas (*foreground*), fundo ou cenário insignificativo (*background*). Este processo pode ser definido pela função s = T(r), onde há comparação do sinal de entrada com um determinado valor de *threshold* (T). O *threshold* pode ser pré-determinado ou calculado dinamicamente de acordo com aspectos da imagem, pois serve como referência para a separação dos níveis de cinza. O sinal de saída 0 (preto) ou 1 (branco) é obtido pela seguinte relação: 1, se r > T, ou 0, se r < T. A ocorrência de igualdade entre r e T é tratada de acordo com a ocasião. (VALA, 2013).

Existem diversos algoritmos de binarização distintos, tais como o método de Otsu (OTSU, 1979), o algoritmo de Klittler-Illingworth (KITTLER, 1985), a limiarização adaptativa de Bernsen (BERNSEN, 1986), a limiarização de Tsai (TSAI, 1985), a limiarização de Niblack (NIBLACK, 1986), entre outros. Em estudo comparativo, aplicado ao reconhecimento de caracteres em manuscritos antigos, Chamchong (2010) destaca o algoritmo de Otsu devido à precisão e velocidade de processamento, ao passo que Gupta (2007) obteve resultados semelhantes quanto a performance do algoritmo de Otsu, apesar de ambos os autores citarem problemas de excesso de ruído ou perda de informações na imagem aplicada a cenários em que a figura original apresentou baixa qualidade (CHAMCHONG, 2010; GUPTA, 2007).

A binarização de Otsu é um algoritmo de binarização desenvolvido por Otsu (1979) que, ao oposto de utilizar um valor arbitrário, calcula o valor de *threshold* automaticamente a partir de análise do histograma da imagem. Ao considerar uma imagem bimodal (imagem cujo histograma apresenta dois picos), o algoritmo de Otsu calcula um valor entre os dois picos do histograma e o utiliza como parâmetro *threshold*, conforme a Figura 2:

**Figura 2 – Imagem após binarização de Otsu e seu histograma, com o valor threshold**



Fonte: Elaborada pelo autor

A proposta deste algoritmo é realizar uma iteração por todos os valores possíveis para o *threshold*, com o objetivo de identificar o valor que minimiza a soma da variância intraclasses da imagem. A variância intraclasse, para um possível valor de *threshold* T, pode ser calculada de acordo com a Equação 1, sendo *W* o peso para cada classe:

 (1)

A variância intraclasses corresponde à probabilidade de um pixel pertencer a classe *background* ou f *foreground* (OTSU, 1979; TOROK, 2014).

## 5.5 Métrica de Alves para avaliação de OCR

Para que seja possível avaliar e classificar os resultados de uma ferramenta de reconhecimento óptico de caracteres, é necessário mensurar o quão distante um texto transcrito está em relação ao texto original. Nestes casos, são aplicadas métricas tais como a Distância de Levenshtein ou a F-score. Entretanto, essas métricas não são específicas para avaliação de resultados OCR, levando à supressão de informações devido à sua natureza genérica (ALVES, 2003; REYNAERT, 2008).

Alves (2003) desenvolveu uma métrica para avaliação de OCR baseando-se na Distância de Levenshtein. A Distância de Levenshtein é um algoritmo que permite comparar e determinar a distância entre duas sequências de caracteres, considerando apenas a soma da quantidade de caracteres inseridos, excluídos e/ou substituídos. Como exemplo da aplicação do algoritmo de Distância de Levenshtein são consideradas as distâncias entre as palavras “mestre” e “mastros”, da seguinte forma:

**mestre**

mastre – substituição de “e” por “a”;

mastro – substituição de “e” por “o”;

mastros – inserção do caractere “s”;

Desta forma, a distância entre “mestre” e “mastros” é equivalente a 3, considerando duas substituições e uma inserção.

Entretanto, esse algoritmo processa a quantidade de erros, sem considerar a localização onde esses erros são gerados. Como a maioria dos softwares de OCR comparam ou recomendam a comparação das palavras do texto extraído com uma base pré-existente de palavras ou um dicionário, a ocorrência de muitos erros em uma única palavra reduz a chance de correção automática nesse processo. A respeito, Alves (2003) declara:

A métrica original de Levenshtein não leva em consideração a onde os erros ocorrem. Por exemplo:

*‘To be or not to be, thaz’s the queztion.’*

*‘To be or not to be, that’s the querkion.’*

A primeira frase do famoso monólogo de Hamlet, de Shakespeare, tem a mesma distância de Levenshtein. Entretanto, o fato da segunda transcrição ter dois erros na mesma palavra (“querkion” em vez de “question”) reduz bastante a probabilidade de recuperação do erro. Então a distância de Levenshtein foi estendida com a taxonomia de classificar os erros e sua posição. (p. 52).

Na métrica proposta por Alves (2003), os erros foram classificados em erros de caractere, de palavra e de linha, de acordo com as definições a seguir:

* Erros de caractere: inclusão (caracteres não-existentes inseridos erroneamente no texto), exclusão (caracteres existentes no texto original excluídos do texto obtido), troca simples (um caractere é substituído por outro), troca múltipla incluinte (um caractere é substituído por dois caracteres, como ‘h’ por ‘l’ e ‘n’), junção de palavras com perda (o espaço separador de palavras mais o próximo caractere são excluídos, resultando em junção de duas palavras com a perda de um caractere) e junção de palavras sem perda (exclusão de um espaço separador de palavras, resultando na junção das mesmas).
* Erros de palavra: inclusão (caracteres não-existentes inseridos erroneamente em uma palavra), exclusão (caracteres existentes no texto original excluídos da palavra), troca simples (um caractere é substituído por outro).
* Erros de linha: inserção (uma linha de texto, inexistente no texto original, é inserida na transcrição) e exclusão (uma linha de texto existente no texto original é removida erroneamente na transcrição).

Assim, de acordo com o trabalho Alves (2003), são propostas duas métricas para avaliar, separadamente, a qualidade dos resultados na transcrição do texto: a primeira métrica considera os erros de caractere e a segunda métrica considera os erros de palavra, dadas de acordo com a Equação 2 e a Equação 3 abaixo:

MC = 100 – ( ( ERR\_CHAR x 100 ) / TOTAL\_CHAR) (2)

MP = 100 – ( (ERR\_PALAVRA x 100) / TOTAL\_PALAVRAS (3)

Onde:

ERR\_CHAR – a quantidade total de ocorrência de erros de caractere;

TOTAL\_CHAR - a quantidade total de caracteres do texto original;

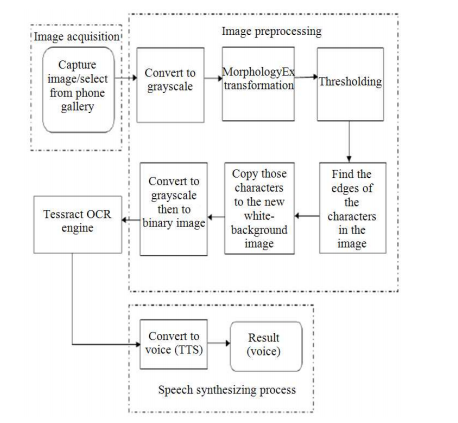
ERR\_PALAVRA - a quantidade total de ocorrência de erros de palavra;

TOTAL\_PALAVRA - a quantidade total de palavras do texto original;

## 5.6 Trabalhos Correlatos

Foong (2013) desenvolveu um modelo para arquitetura e desenvolvimento de aplicativos para dispositivos móveis que trabalha com síntese de voz, de textos de placas, sinalizações e avisos, obtidos por meio de OCR, fazendo uso da Tesseract Engine, afim de prover maior independência para indivíduos portadores de deficiência visual que, com a ajuda do aplicativo, podem identificar informações impressas em ambientes internos (interior de residências ou edifícios comerciais) como quadros de avisos, placas de alerta ou sinalização de rotas. O autor também tratou o aumento da precisão do processo de OCR. Então, propôs o processamento da imagem antes da submissão ao Tesseract, convertendo esta em *grayscale*. Após esta etapa aplica filtros de redução de ruído e efetua a binarização da imagem por meio do algoritmo de Otsu, e assim isola os caracteres do restante da imagem. Desta forma, a imagem analisada pelo Tesseract contém apenas caracteres, pois o fundo é removido pela etapa de pré-processamento. O texto extraído pelo Tesseract é então sintetizado a partir do recurso TTS, nativo do Sistema Operacional Android. O modelo proposto é descrito pela Figura 3:

**Figura 3 - Arquitetura proposta por Foong**

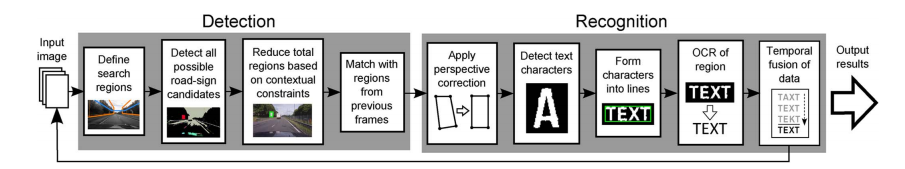


Fonte: Foong (2013)

Foong descreve ainda que a precisão dos resultados foi obstruída por fatores como iluminação e coloração do texto, embora de maneira geral tais fatores tenham se mostrado muito satisfatórios (FOONG, 2013).

Greenhalgh (2015) também apresentou uma proposta de modelo de arquitetura de aplicativos móveis para detecção de símbolos e textos presentes em sinais de trânsito. Greenhalgh propôs, assim como Foong, uma etapa de processamento da imagem prévia ao reconhecimento óptico de caracteres, afim de eliminar elementos não-textuais, aumentando a performance e a precisão do OCR. Greenhalgh propôs o isolamento das regiões de texto e a correção de distorções de perspectiva na imagem antes da etapa de OCR, conforme a Figura 4:

**Figura 4- Arquitetura proposta por Greenhalgh**



Fonte: Greenhalgh (2015)

Greenhalgh utilizou a F-measure, método estatístico de precisão e revocação para avaliar os resultados do modelo (GREENHALGH, 2015).

Alves (2003) desenvolveu uma pesquisa na área de reconhecimento óptico de caracteres, tendo como objetivo primário o desenvolvimento de uma métrica baseada no algoritmo da Distância de Levenshtein, de modo a mensurar o impacto da aplicação, frente a diferentes filtragens, na imagem, antes de submetê-la a ferramentas OCR comerciais. Assim, além de quantificar a interferência de cada filtro tais como brilho, contraste, resolução, rotação e etc, Alves descreve a métrica utilizada para avaliação dos resultados individuais de cada filtro (ALVES, 2003).

# 6 METODOLOGIA

Os trabalhos de Foong (2013), Greenhalgh (2015) e Rusiñol (2014) evidenciaram a necessidade de processamento da imagem antes da submissão ao Tesseract, afim de minimizar a influência de fatores externos, relacionados à natureza do ambiente onde a foto é capturada e do próprio processo de captura, que geram ruídos que acabam por afetar o processo de OCR. Assim, este trabalho apresenta uma proposta de modelo de arquitetura de um aplicativo móvel que viabilize a síntese de voz de imagens capturadas em ambiente livre, a partir da câmera de smartphones e outros dispositivos móveis.

Com o intuito de permitir a aferição e comparação dos resultados do modelo, este trabalho também apresenta uma proposta de modelo de testes e medição dos resultados por meio de um método estatístico que é explanado em detalhes no tópico 6.4.

A metodologia empregada na concepção de um modelo de arquitetura e no desenvolvimento geral deste projeto é composta de seis etapas distintas e ilustradas na Figura 5:

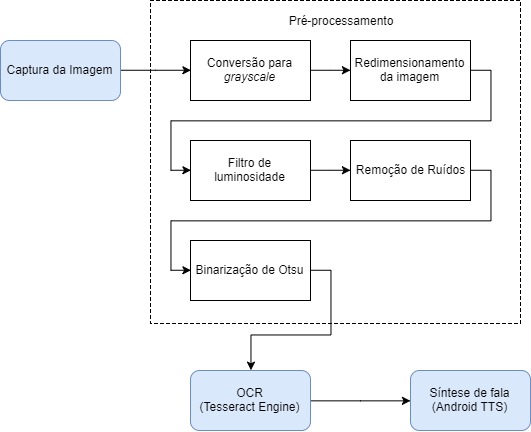
**Figura 5 - Passos da Metodologia**

Fonte: Elaborada pelo autor

## 6.1 Modelo de Arquitetura

O modelo de arquitetura para desenvolvimento de aplicativos, cujo processo compreenda o reconhecimento de caracteres a partir de fotografias, é composto por cinco etapas distintas de processamento, de acordo com a Figura 6:

**Figura 6: Fluxo de processamento do aplicativo**

****

Fonte: Elaborada pelo autor

A primeira etapa, identificada como Captura da Imagem, consiste na obtenção da imagem por meio de uma câmera fotográfica, cujo recurso é normalmente disponível na maioria dos smartphones e tablets. A captura pode utilizar as funções nativas da plataforma Android, não sendo necessário o uso de tecnologias adicionais.

A segunda etapa, identificada como Pré-Processamento, é a etapa do modelo na qual a imagem capturada é processada, de forma a eliminar ou minimizar ruídos e outras interferências, potencializando a precisão do OCR na etapa de detecção e reconhecimento dos caracteres. Os processos envolvidos no tratamento da imagem são:

* Redimensionamento: processo onde a imagem é redimensionada, afim de atingir a resolução de 300 dpi, densidade na qual a Tesseract apresenta melhor precisão (GITHUB.COM, 2017);
* Conversão para *grayscale*: processo no qual a imagem, até então colorida, é convertida em escala de cinza, formato no qual a Tessract apresenta melhor precisão;
* Aplicação de filtros de luminosidade e contraste: alteram automaticamente os aspectos de iluminação e contraste da imagem, de forma a destacar e definir bordas, evidenciando a separação de blocos de texto do resto da imagem. As ferramentas propostas para implementação são os filtros de luminosidade e contraste presentes na biblioteca GPUImage, descritos no capítulo 5.4.2;
* Binarização: processo por meio do qual a imagem é convertida em preto e branco, de forma a separar o que é classificado como ‘background’ (elementos secundários na imagem) e ‘foreground’ (elementos primários, incluindo os textos). A tecnologia proposta é o algoritmo de Otsu, descrito no capítulo 5.4.4;
* Remoção de ruídos: processo que permite a identificação pequenos componentes desconexos e ruídos na imagem, removendo-os. Propõe-se o emprego das funções Erode e Dilate da biblioteca *OpenCV*, descritos no capítulo 5.4.3;

A terceira etapa, identificada como OCR, é definida e entendida de modo que as linhas de texto são identificadas e os caracteres presentes são extraídos, analisados e identificados por meio de comparação da imagem com padrões pré-cadastrados. A tecnologia proposta é a Tesseract OCR que em 1995 foi considerada um dos três softwares OCR com maior precisão e desempenho (SMITH, 1995).

A quarta e última etapa, identificada como Síntese de voz, é responsável pela exibição, em áudio, do resultado do processamento OCR, fazendo uso da API TTS, compatível com o sistema operacional Android, onde é utilizada para conversão automática de texto digital editável em voz, ou seja, áudio. Nas versões mais recentes do Android~~,~~ já estão disponíveis as engines de voz nos idiomas italiano, francês, inglês, alemão e espanhol. Nem todas as versões suportam o idioma português do Brasil, conhecido tecnicamente como *pt-BR*. (LECHETA, 2013)

## Aplicação do Modelo

Uma vez proposto o modelo de arquitetura foi desenvolvido um aplicativo móvel (APP) para aplicação do modelo, a fim de cumprir o objetivo de prover uma ferramenta de acessibilidade para auxílio de leitura para indivíduos portadores de deficiência visual ou àqueles que apresentem algum grau de analfabetismo.

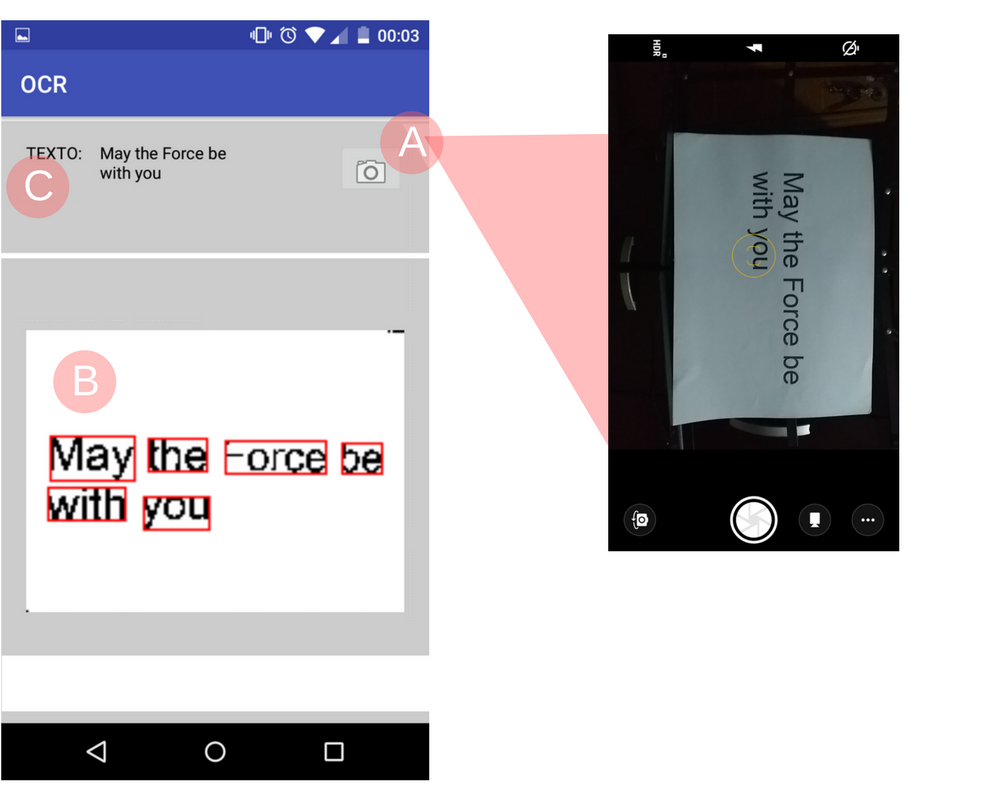
Para a construção de um aplicativo para dispositivos móveis, baseado no modelo de arquitetura proposto, é sugerido como requisitos técnicos o uso de tecnologias gratuitas, tais como: o pacote de ferramentas de desenvolvimento Android SDK, a Tesseract Engine OCR (encapsulada em um *wrapper* para a linguagem Java na biblioteca Tess4J) devido ao bom desempenho e precisão~~;~~ e as bibliotecas OpenCV, GPUImage e Leptonica para processamento das imagens, devido à compatibilidade com a arquitetura e a qualidade dos resultados. Os recursos de captura de imagem e Text-To-Speech podem ser encontrados nativamente na SDK do Android, de forma que não é necessário a instalação de recursoa adicionais.

Como requisitos funcionais da aplicação foram consideradas as capacidades de: a) adquirir uma imagem por meio da câmera de dispositivos móveis; b) de interpretar programaticamente a figura obtida de forma a identificar e extrair quaisquer textos nela presentes; c) de reproduzir em áudio os textos identificados.

Com base nos requisitos supracitados, foi desenvolvido um APP capaz de reproduzir, em formato de áudio, textos a partir de capturas fotográficas em um smartphone, aplicado como ferramenta de auxilio de leitura para deficientes visuais ou analfabetos.

A seguir é apresentado o APP desenvolvido, por meio do qual se aplicam as etapas previstas no modelo de arquitetura, conforme descrito no capítulo 6.1. Cada etapa do modelo contempla uma ou mais interfaces da aplicação, as quais serão manipuladas pelo usuário para que o texto impresso seja capturado, processado, reconhecido e reproduzido em áudio. A interface geral do aplicativo é descrita de acordo com a Figura 07:

**Figura 7: Interface geral do aplicativo**



Fonte: Elaborada pelo autor

Na interface do aplicativo, ilustrada na Figura 07, são representadas as etapas previstas no modelo, conforme:

1. Obtenção da imagem: Ao tocar o botão da captura, conforme ilustrado na Figura 7a, o usuário é direcionado à tela de captura de imagem, passo correspondente à etapa de “Captura da Imagem” prevista no modelo ilustrado na Figura 6. Todo o processo de captura da imagem é gerenciado pelo sistema operacional Android, podendo apresentar diferenças visuais e funcionais, de acordo com a versão utilizada.
2. Processamento da imagem: Assim que a imagem é concluída, o aplicativo processa de acordo com os passos previstos nas etapas “Pré-Processamento”, vide modelo da Figura 6, e então submete a imagem já processada ao Tesseract, conforme o modelo (etapa “OCR”). Então, o Tesseract identifica as regiões com potencial para abrigar linhas de texto e o aplicativo exibe a imagem resultante do pré-processamento já com as regiões de texto demarcadas pelo Tesseract, conforme ilustrado na Figura 7b, para apuração da manipulação da imagem para que o Tesseract, finalmente, possa extrair os caracteres identificados nessas regiões.
3. O texto extraído é enviado para síntese de voz, como última etapa do modelo proposto (etapa “Síntese de Voz”, também ilustrado na Figura 6). Todo o processo de text-to-speech é gerenciado pelo Android. O texto é também exibido ao usuário para apuração dos resultados e confirmação visual do produto do aplicativo, conforme ilustrado na Figura 6c.

A partir da aplicação prática, a partir do modelo de arquitetura proposto, foram elaborados experimentos, aferições e análise dos resultados, conforme descritos a seguir.

## 6.3 Experimentos práticos

A partir do protótipo da APP construído foram elaborados experimentos práticos, afim de analisar o comportamento da ferramenta em ambientes com características construtivas. Os mesmos testes foram aplicados a uma versão do APP desenvolvido, sem a aplicação do modelo, ou seja, onde foi removida a etapa de pré-processamento da imagem, de forma a possibilitar a análise e comparação de resultados de ambas as aplicações.

Os experimentos realizados consistiram na extração de caracteres de textos impressos por uma impressora Epson Ecotank L375, em papel sulfite A4.

A tipografia dos textos impressos foi padronizada nas fontes: Arial, Calibri e Adobe Gothic Std B, com tamanhos e cores variadas ao acaso. A iluminação do local de testes foi considerada adequada, ou seja, com a fonte de luz posicionada de forma estratégica para não provocar sombras sobre a superfície escrita, e as imagens capturadas de forma a não causar distorções de perspectiva.

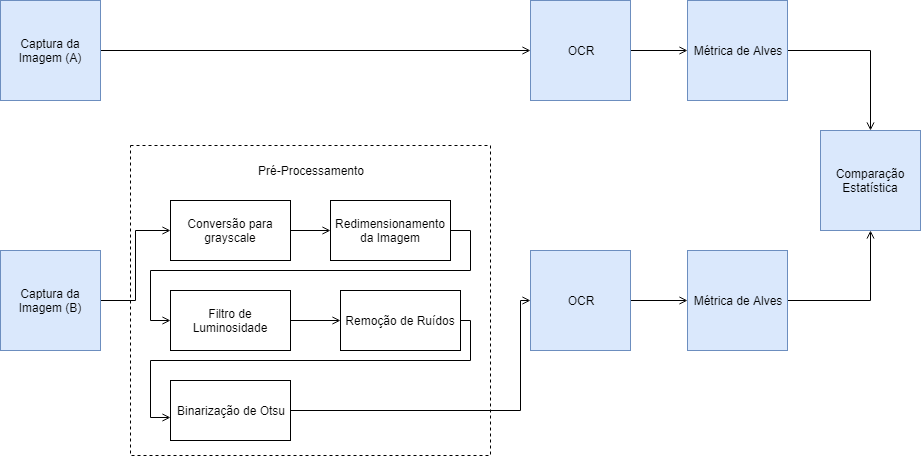
O aplicativo foi executado em um smartphone Motorola Moto G 1ª Geração, equipado com câmera traseira com resolução de 5 MegaPixels. As imagens foram capturadas em tamanho 1944 pixels de largura e 2592 pixels de altura, com 72 DPI. Os recursos de zoom e flash não foram acionados em nenhuma das capturas.

## 6.4 Avaliação dos resultados

Para mensurar e avaliar o impacto das intervenções propostas, este trabalho propôs um modelo comparativo, baseado no trabalho de Alves (2003), onde os resultados da Tesseract OCR, a partir de imagens originais, foram comparados com os resultados de imagens que sofreram alterações, por meio dos processos descritos anteriormente.

Assim, o processo de avaliação e comparação dos resultados é descrito pelo diagrama apresentado na Figura 8:

**Figura 8: Diagrama do processo de avaliação e comparação dos resultados de OCR**



Fonte: Elaborada pelo autor

Para avaliar se há diferença estatisticamente significante entre as duas versões do APPs, utilizou-se o teste de Wilcoxon (HOLLANDER, 2013). O teste permite, a partir de duas dadas populações, atestar a rejeição ou não-rejeição da hipótese nula, ou seja, a hipótese onde não há diferenças estatisticamente significantes entre as duas populações.

Entre as justificativas para o emprego do teste de Wilcoxon, destaca-se a aplicabilidade diante dos dados obtidos por meio dos testes - os experimentos práticos conceberam duas populações pareadas, sendo a primeira a representação da avaliação via métrica de Alves do aplicativo sem a aplicação do modelo e a segunda com a aplicação do modelo. Além disso, teste de Wilcoxon possui ampla aceitação e empregabilidade e possui diversas implementações em bibliotecas matemáticas como a ALGLIB, a GNU Octave e a SciPy, além da implementação no software estatístico R (ALGLIB.COM, 2018; GNU.ORG, 2018; SCIPY.ORG, 2018).

Assim, após a coleta dos resultados dos testes das duas versões do APPs, sendo o primeiro desenvolvido a partir das diretrizes do modelo e o segundo sem as diretrizes do modelo, esses valores foram analisados estatisticamente por meio do teste de Wilcoxon, afim de determinar se a aplicação do modelo proposto tende a gerar resultados mais precisos.

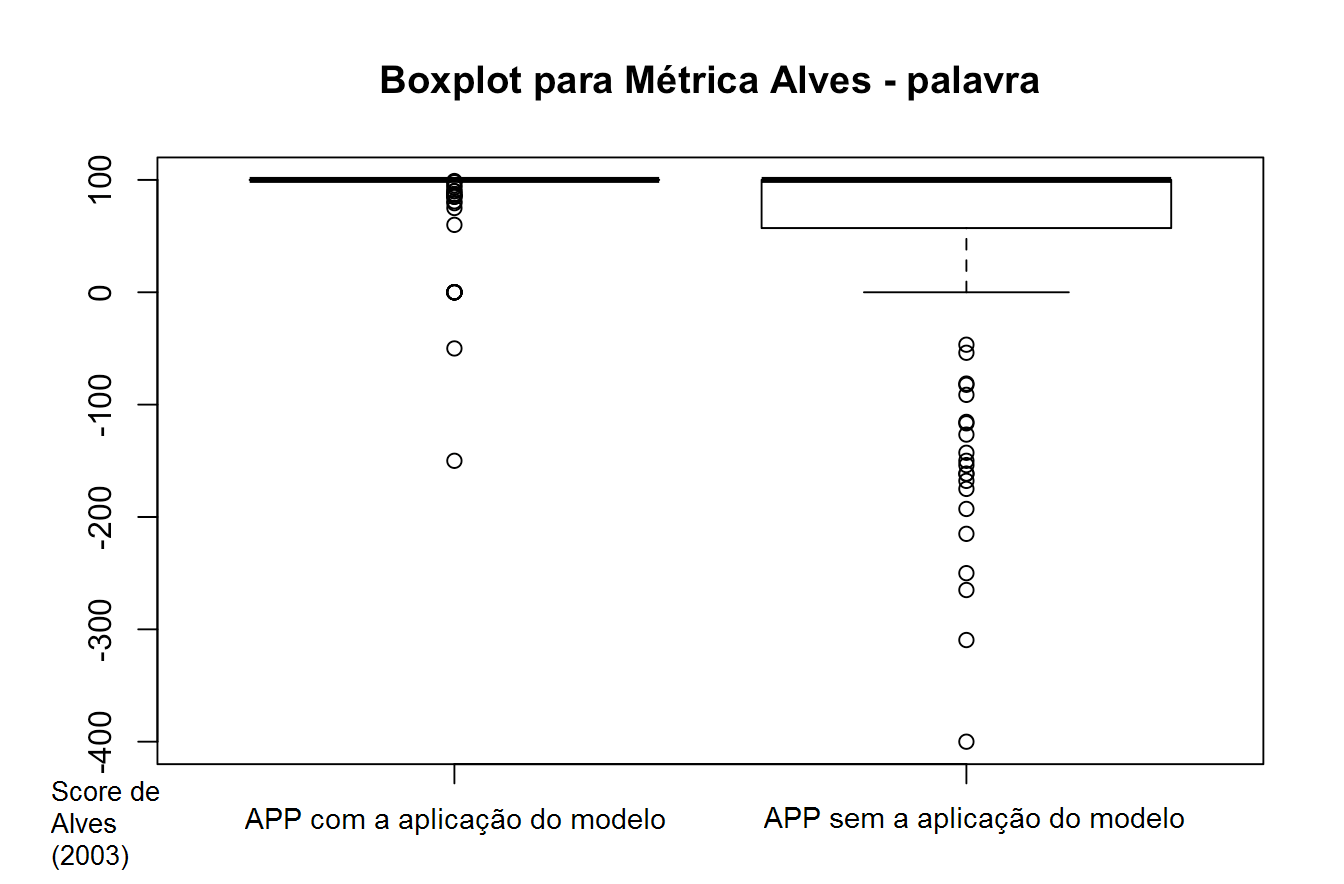
Na Tabela 01 abaixo é apresentada uma amostra de resultados de 10 dos 130 testes efetuados, utilizando a Métrica de Alves por Caractere:

**Tabela 1 Amostra dos resultados dos testes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Texto de Entrada** | **Métrica de Alves (APP com a aplicação do modelo)** | **Métrica de Alves (APP sem a aplicação do modelo)** | **Classificação do fundo** |
| WINTER IS COMING | 100 | 58,82 | Fundo simples |
| You Know Nothing, Jon Snow | 100 | 100 | Fundo simples |
| I am Daenerys Stormborn of House Targaryen, the Unburnt, Mother of Dragons, khaleesi to Drogo's riders, and queen of the Seven Kingdoms of Westeros. | 87,2 | 34,4 | Fundo simples |
| STAR WARS | 87,5 | 75 | Fundo complexo |
| I am your father | 100 | 84,61 | Fundo simples |
| May the Force be with you | 100 | 95 | Fundo simples |
| The Lion King | 100 | 45,45 | Fundo complexo |
| HAKUNA MATATA | 100 | 100 | Fundo simples |
| BATMAN | 100 | 100 | Fundo complexo |
| Gothan City | 90 | 100 | Fundo complexo |

Fonte: Elaborada pelo autor

A Figura 9 apresenta os resultados utilizando a métrica Alves por palavra, separadamente para as duas versões do APP. A diferença entre os APPs é estatisticamente significante (estatística do teste Wilcoxon, v= 1207.5, p-valor = 3.846).

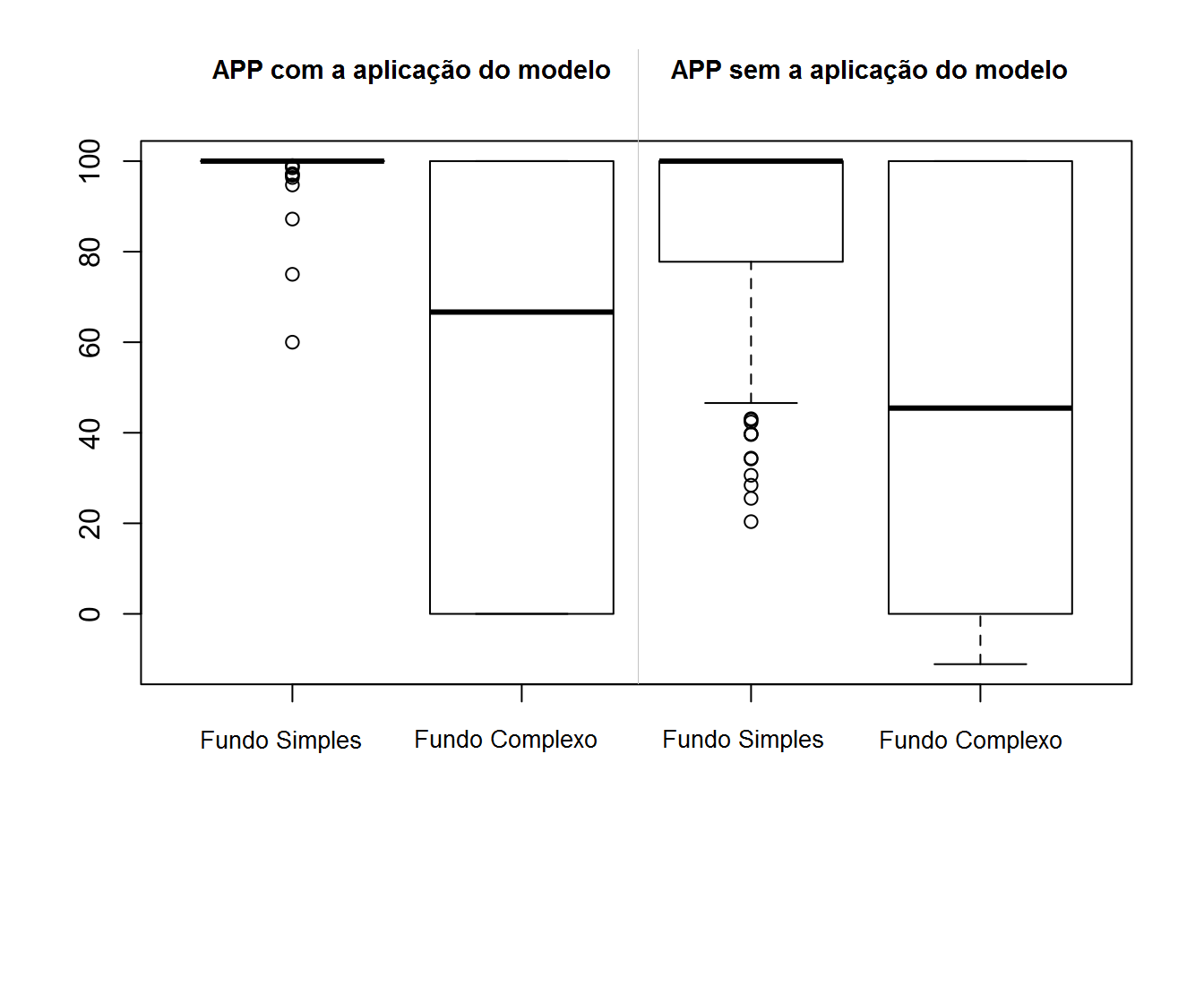
**Figura 9 - Boxplot para os resultados de acordo com a métrica de Alves, por palavra**

Fonte: Elaborada pelo autor

Apesar de a mediana de ambos os grupos ser a mesma, tanto o limite inferior quanto o valor do quartil inferior são menores para os resultados dos testes do APP desenvolvido sem a aplicação do modelo, ou seja, sem o pré-processamento da imagem. Assim, torna-se evidente que o pré-processamento influencia positivamente na precisão dos resultados da extração de caracteres feita pelo Tesseract.

Entretanto, ao analisar separadamente a população de resultados agrupados por existência e não-existência de fundo complexo na imagem processada, não há evidências estatísticas suficientes para rejeitar a hipótese nula para os dados de fundo complexo (estatística do teste de Wilcoxon v = 25.5, p-valor=0.3264 ) - ao contrário dos casos onde não há fundo complexo, onde a hipótese nula continua sendo rejeitada (estatística do teste de Wilcoxon v=896.5, p-valor=2.728). Esses resultados são apresentados na Imagem 10:

**Figura 10 - Boxplot para os resultados agrupados por tipo de fundo**



Fonte: Elaborada pelo autor

# 7 CONCLUSÃO

O modelo de arquitetura para desenvolvimento de aplicativos OCR proposto neste trabalho sugere a aplicação de uma etapa de tratamentos e intervenções na imagem a ser trabalhada antes do seu processamento pelo Tesseract OCR. O modelo foi aplicado por meio do desenvolvimento prático de um aplicativo assistivo para leitura automática de textos extraídos de imagens. Também foi desenvolvido uma versão deste aplicativo, subtraindo a etapa de intervenções proposta no modelo. Ambos os aplicativos foram submetidos a experimentos práticos estruturados e avaliados de acordo com a métrica de Alves, e posteriormente analisados por meio do teste estatístico de Wilcoxon.

Os testes apontam um aperfeiçoamento considerável na precisão do processo de OCR quando aplicadas as intervenções propostas. Contudo, esse aperfeiçoamento só foi comprovado nos cenários de teste que possuíam imagens com fundo simples. Os cenários que apresentavam fundo complexo não apresentaram resultados estatisticamente mais precisos.

Como trabalhos futuros, sugere-se adicionar um processamento de correção de perspectiva na etapa de pré-processamento, assim como descrito por Greenhalgh (2015) e Foong (2013), bem como isolar as áreas contendo texto do resto da imagem, de forma a submeter ao Tesseract apenas as áreas da imagem que contenham informação textual, descartando o restante, afim de aumentar a performance e reduzir a possibilidade de erros, assim como fez Foong.

Como a análise estatística constatou que a melhora de precisão ainda não atingiu o resultado esperado nos casos onde a imagem processada apresenta fundo complexo, também se propõe o estudo e aplicação de algoritmos alternativos de binarização, que são responsáveis por isolar os blocos de texto do resto da imagem.

**REFERÊNCIAS**

ADDISON, Edwin R. et al. **Text to speech**. U.S. Patent n. 6,865,533, 8 mar. 2005.

ALGLIB.COM. Wilcoxon signed-rank test. Disponível em: < http://www.alglib.net/hypothesistesting/wilcoxonsignedrank.php>. Acesso em: 08 junho de 2018.

ALVES, Neide F. Estratégias para melhoria do desempenho de ferramentas comerciais de reconhecimento óptico de caracteres. 2003.

BERNSEN, J. Dynamic Thresholding of Gray Level Image, ICPR`86: **Proceedings of International Conference on Pattern Recognition**, Berlin, 1986, pp. 1251-1255.

BERSCH, Rita. Introdução à tecnologia assistiva. **Porto Alegre: CEDI**, 2008.

BHASKAR S. et al., Implementing Optical Character Recognition on the Android Operating System for Business Cards, 2010.

CHAMCHONG R, Fung C. C, Wong K. W. Comparing Binarisation Techniques for the Processing of Ancient Manuscripts. Ryohei Nakatsu; Naoko Tosa; Fazel Naghdy; Kok Wai Wong; Philippe Codognet. Second IFIP TC 14 Entertainment Computing Symposium (ECS) / Held as Part of World Computer Congress (WCC), Sep 2010, Brisbane, Australia. Springer, IFIP Advances in Information and Communication Technology, AICT-333, pp.55-64, 2010, Cultural Computing.

CHATTOPADHYAY, Tanushyam; SINHA, Priyanka; BISWAS, Provat. Performance of document image OCR systems for recognizing video texts on embedded platform. In: **Computational Intelligence and Communication Networks (CICN), 2011 International Conference on**. IEEE, 2011. p. 606-610.

FOONG, Oi-Mean; SULAIMAN, Suziah; LING, Kiing Kiu. Text signage recognition in Android mobile devices. **Journal of Computer Science**, v. 9, n. 12, p. 1793, 2013.

GITHUB.COM. Tessetact-OCR - ImproveQuality. 2017. Disponível em: < https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality >. Acesso em: 13 junho de 2017.

GNU.ORG. GNU Octave. Disponível em: < https://www.gnu.org/software/octave/>. Acesso em: 08 de junho de 2018.

GREENHALGH, Jack; MIRMEHDI, Majid. Recognizing text-based traffic signs. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, n. 3, p. 1360-1369, 2015.

Gupta M.R., Jacobson N.P., Garcia E.K.: OCR binarization and image pre-processing for searching historical documents. Pattern Recognit **40**(2), 389–397 (2007)

HOLLANDER, Myles; WOLFE, Douglas A.; CHICKEN, Eric. **Nonparametric statistical methods**. John Wiley & Sons, 2013.

KITTLER, Josef; ILLINGWORTH, John; FÖGLEIN, J. Threshold selection based on a simple image statistic**. Computer vision, graphics, and image processing**, v. 30, n. 2, p. 125-147, 1985.

LECHETA, Ricardo R. Google Android-3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK. Novatec Editora, 2013.

NIBLACK, W. An Introduction to Digital Image Processing. **Prentice Hall**, 1986.

OPENCV.ORG. Eroding and Dilating. Disponível em: < http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion\_dilatation/erosion\_dilatation.html>. Acesso em: 14 de junho de 2017.

OTSU, Nobuyuki. A threshold selection method from gray-level histograms. **IEEE Transactions on systems, man, and cybernetics**, v. 9, n. 1, p. 62-66, 1979.

REDIG, Annie Gomes; DO COUTO JUNIOR, Dilton Ribeiro. A tecnologia assistiva nos processos de leitura e escrita na educação inclusiva. **Informática na educação: teoria & prática**, v. 15, n. 2, 2012.

RESNIKOFF, Serge et al. Global data on visual impairment in the year 2002. **Bulletin of the world health organization**, v. 82, n. 11, p. 844-851, 2004.

REYNAERT, Martin. All, and only, the Errors: more Complete and Consistent Spelling and OCR-Error Correction Evaluation. In: **LREC**. 2008.

RUSIÑOL, Marçal; CHAZALON, Joseph; OGIER, Jean-Marc. Combining focus measure operators to predict ocr accuracy in mobile-captured document images. In: **Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on**. IEEE, 2014. p. 181-185.

SCIPY.ORG. Wilcoxon. Disponível em: < https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html>. Acesso em: 08 de junho de 2018.

SMITH, Ray. A simple and efficient skew detection algorithm via text row accumulation. In**: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on**. IEEE, 1995. p. 1145-1148.

SMITH, Ray. An Overview of the Tesseract OCR Engine. ICDAR. Vol. 7. No. 1. 2007.

TATHAM, Mark, MORTON, Katherine: Developments in Speech Synthesis, John Wiley & Sons Ltd: England, (2005).

TOROK, Leonardo; DE IMAGENS, Análise; CONCI, Aura. Método de Otsu. **Instituto de Computação–Universidade Federal Fluminense, Niterói–RJ–Brasil, notas de aula**, 2014.

TSAI, Wen-Hsiang. Moment-preserving thresolding: A new approach. **Computer Vision, Graphics, and Image Processing**, v. 29, n. 3, p. 377-393, 1985.

VALA, Miss Hetal J.; BAXI, Astha. A review on Otsu image segmentation algorithm. **International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)**, v. 2, n. 2, p. pp: 387-389, 2013.

VERMA, Abhishek; ARORA, Suket; VERMA, Preeti. OCR-OPTICAL CHARACTER RECOGNITION. In: **7th International Conference on Recent Innovations in Science, Engineering and Management**. 2016.

WHITE, D. Heather; ROBERTSON, Lorayne. Implementing assistive technologies: A study on co-learning in the Canadian elementary school context. **Computers in Human Behavior**, v. 51, p. 1268-1275, 2015.

YANAGUYA, G. A. Aplicação de Tradução OCR Multiplataforma via Web Services, 2015.

1. OCR (Optical Character Recognition) é uma tecnologia utilizada para detecção e extração de caracteres a partir de um arquivo de imagem (VERMA, 2016) [↑](#footnote-ref-1)
2. A precisão dos resultados de reconhecimento de caracteres é abordada no capítulo 5.4. [↑](#footnote-ref-2)
3. Tesseract OCR ou Tesseract Engine é um *software* de reconhecimento óptico de caracteres *open-source* com distribuição gratuita desenvolvido pela Hewlett Packard e mantido pela Google. [↑](#footnote-ref-3)
4. *Parsing* semântico e *parsing* gramatical são processos nos quais uma sentença é analisada afim de determinar a sua estrutura no ponto de vista gramatical e semântico. [↑](#footnote-ref-4)
5. API (Application Programming Interface) é definido como o conjunto de padrões estabelecidos por um software afim de permitir sua utilização por outros softwares ou sistemas computacionais. [↑](#footnote-ref-5)
6. Binarização de Otsu ou simplesmente Método de Otsu, é um algoritmo que permite a limiarização automática de uma imagem, ou seja, a conversão de uma imagem qualquer em uma imagem em preto e branco, afim de separar objetos em evidência de um fundo em comum (OTSU, 1979). [↑](#footnote-ref-6)
7. OpenCV (Open Source Computer Vision Library) é uma biblioteca multiplataforma, gratuita para uso acadêmico e comercial, para o desenvolvimento de softwares na área de Visão Computacional. [↑](#footnote-ref-7)