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 expansão das tecnologias na área de informática e o crescimento dos recursos de hardware e software disponíveis em dispositivos móveis intensificam o potencial que essas tecnologias têm de assumirem o papel de principal ferramenta de auxílio a indivíduos portadores de deficiência, promovendo a sua autonomia, independência e qualidade de vida.

A síntese de fala, que permite a conversão automática de texto digital em áudio, é um recurso recorrente em softwares de tecnologia assistiva. Entretanto, apesar do crescimento considerável de pesquisas e produtos lançados nessa área, ainda é modesta a abrangência e eficiência dessa tecnologia, principalmente em casos onde o texto não se encontra em formato de texto digital editável, como textos impressos ou em formato de imagem ou vídeo.

Este projeto tem como principal objetivo a análise e desenvolvimento de um aplicativo Android para síntese de voz de textos extraídos de fotografias obtidas a partir de câmera fotográfica de smartphones e tablets. Os resultados obtidos após a submissão do aplicativo a experimentos práticos possibilitou, por meio da aplicação de uma métrica de avaliação de processamento de extração óptica de caracteres, determinar aspectos que influenciam negativamente na precisão do OCR, bem como identificar intervenções que minimizam os efeitos desses aspectos.

Palavras-chave: OCR. Tecnologias assistivas. 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

Análise e definição de um modelo de arquitetura para viabilização de aplicativo móvel para síntese automática de voz de textos extraídos de fotografias por meio de OCR[[1]](#footnote-1), bem como sua aplicação como ferramenta de auxílio de leitura para portadores de deficiência visual ou àqueles que apresentarem grau de analfabetismo que impeça ou dificulte a leitura de texto, afim de validar o modelo e coletar dados de sua aplicação prática.

# 3 JUSTIFICATIVA

Sendo a tecnologia OCR uma importante ferramenta para a construção de softwares de acessibilidade, principalmente para construção de aplicações de auxílio de leitura para portadores de deficiência visual, a baixa precisão no reconhecimento óptico dos caracteres em imagens capturadas por câmeras fotográficas de dispositivos móveis destaca-se como um obstáculo a ser transposto. Atualmente, a maioria dos softwares OCR possuem desempenho consideravelmente maior quando aplicados na extração de caracteres de documentos de texto escaneados, apresentando queda de precisão considerável quando aplicados na extração de caracteres em fotografias de ambiente aberto (REDIG, 2012; RESNIKOFF, 2002; RUSIÑOL, 2014).

# 4 OBJETIVOS

Este trabalho apresenta como objetivo geral a análise e proposta de um modelo de arquitetura para desenvolvimento de aplicativos de síntese de voz de textos obtidos a partir de imagens por meio de OCR, aplicado neste trabalho 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 favorecer a precisão do Tesseract OCR[[2]](#footnote-2);
* Desenvolvimento de um aplicativo baseado no modelo proposto, afim de coletar dados para validação e avaliação do modelo;
* Adoção de uma métrica de mensuração de resultados de processamento OCR, bem como estruturação de testes condizentes com essa 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 assistiva para auxílio à leitura

Tecnologia assistiva (TA) é definida como o conjunto de recursos e serviços que contribuem para ampliar ou proporcionar as habilidades funcionais de indivíduos portadores de deficiências, proporcionando inclusão e independência aos seus utilizadores. Dentre as diversas aplicações da TA, este trabalho apresenta aspectos das tecnologias de auxílio para a suporte da habilidade visual e recursos que ampliam a informação a pessoas com baixa visão ou cegas, sintetizando 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:

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;
6. A associação de uma sequência fonológica à cada unidade prosódica;
7. A geração do sinal de fala (saída de áudio);

(ADDISON et. al., 2005; TATHAM, 2005)

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.

## 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 (OCH, 2015).

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,* por permitir o treinamento do software para reconhecimento de qualquer linguagem ou tipo de caracteres e 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 em 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 é responsável pela manutenção e distribuição do *software*, sob licença Apache 2.0 (SMITH, 1995).

Inicialmente desenvolvido na linguagem C e posteriormente migrado para C++, o 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), o Tesseract é considerado 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 utilizado em fotos capturadas através de dispositivos móveis, alguns fatores, grande parte relacionados a natureza do ambiente, influenciam na captura da imagem, causando distorções e ruídos que podem culminar em perda de precisão no processo de extração. Entre esses 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 de presença de objetos que podem ser confundidos com caracteres ou até linhas de texto (RUSIÑOL, 2014; YANAGUYA, 2015).

### 5.4.1 Processamento Básico

Internamente, 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 de esse processamento se apresentar eficiente em muitos casos, existem cenários onde esse processamento não atinge os resultados esperados, resultando em uma 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[[3]](#footnote-3). São eles:

* Redimensionamento da imagem – O Tessetact apresenta resultados melhores em imagens com densidade 300dpiou superior, sendo benéfico redimensionar a imagem afim de atingir essa característica; e
* Conversão da imagem em *grayscale* – O processo de binarização de Otsu[[4]](#footnote-4), utilizado internamente 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 de brilho ou cor 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[[5]](#footnote-5).

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, subdividindo a 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*) e fundo ou cenário insignificativo (*background*). O processo pode ser definido pela função s = T(r) : compara-se o sinal de entrada com um determinado valor de *threshold* (T). Esse valor pode ser pré-determinado ou calculado dinamicamente de acordo com aspectos da imagem, e 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, como o método de Otsu, o algoritmo de Klittler & Illingworth, a limiarização adaptativa de Bernsen, a limiarização de Tsai, a limiarização de Niblack, 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. 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 em cenários onde a figura original apresenta 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. Considerando 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*. A proposta do 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:



Sendo *W* o peso para cada classe. Esse valor corresponde à probabilidade de um pixel pertencer a classe b (*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. Para isso, empregam-se métricas 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 dois textos, considerando apenas a soma da quantidade de caracteres inseridos, excluídos e/ou substituídos.

Como exemplo, a Distância de Levenshtein entre a palavra “mestre” e a palavra “mastros” é dada da seguinte forma:

**mestre**

mastre – substituição de “e” por “a”;

mastro – substituição de “e” por “o”;

mastros – inserção do caractere “s”;

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

Entretanto, esse algoritmo computa 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, os erros foram classificados em erros de caractere, de palavra e de linha, sendo:

* 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 delas).
* 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, são propostas duas métricas para avaliar separadamente a qualidade dos resultados da transcrição do texto: uma considerando os erros de caractere e uma considerando os erros de palavra, dadas da seguinte forma:

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

e

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

Sendo:

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;

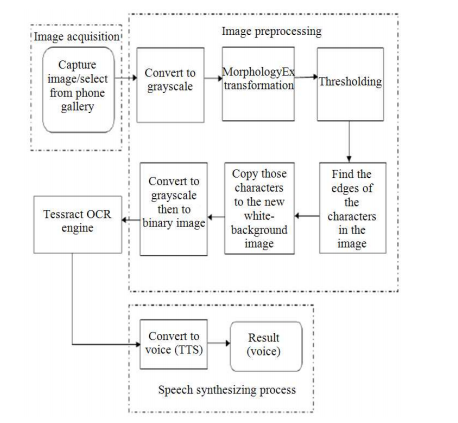
TOTAL\_PALAVRA - a quantidade total de palavras do texto original;

## 5.6 Trabalhos Correlatos

FOONG (2013) desenvolveu um modelo para arquitetura e desenvolvimento de aplicativos para dispositivos móveis para síntese de voz de textos de placas, sinalizações e avisos, obtidos por meio de OCR, empregando a 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.

Foong, afim de aumentar a precisão do processo de OCR, também propôs o processamento da imagem antes da submissão ao Tesseract, convertendo-a em *grayscale*, aplicando filtros de redução de ruído, efetuando a binarização da imagem pelo algoritmo de Otsu, e isolando os caracteres do resto da imagem. Assim, a imagem analisada pelo Tesseract contém apenas caracteres, pois o fundo é removido pelo pré-processamento. O texto extraído pelo Tesseract é então sintetizado a partir do recurso TTS nativo do Android. O modelo proposto é descrito pela Figura 2:

**Figura 2 - Arquitetura proposta por Foong**

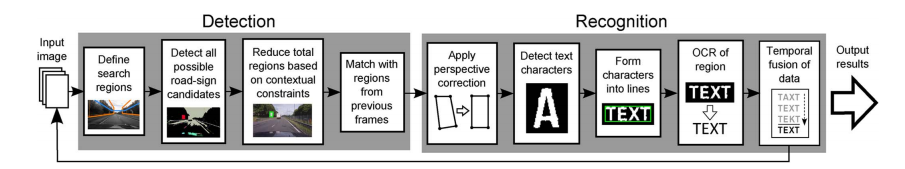


Fonte: Foong (2013)

Foong descreve ainda que a precisão dos resultados foi obstruída por fatores como iluminação e coloração do texto, embora de maneira geral 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 3:

**Figura 3- 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) também 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, afim de mensurar o impacto da aplicação de diferentes filtragens na imagem antes de submetê-la a ferramentas OCR comerciais. Assim, além de quantificar a interferência de cada filtro (brilho, contraste, resolução, rotação, etc), Alves também descreve a métrica utilizada para avaliar os resultados individuais deles (ALVES, 2003).

# 6 METODOLOGIA

* Melhorar a apresentação do início do capítulo, levando em consideração um txto “novo” seu uso de fragmentos do objetivo, justificativa e deixando claro qual a sua proposta frente à teoria que você estudou.
* Colocar a representação para o leitor dos passos que você usou para formular a 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.

Afim de permitir a aferição e comparação dos resultados do modelo, este trabalho também apresenta uma proposta de modelo de testes, medição e comparação de resultados por meio de um método estatístico.

A metodologia empregada na concepção de um modelo de arquitetura e no desenvolvimento geral deste projeto é composta de seis etapas distintas, ilustradas pela Figura 4:

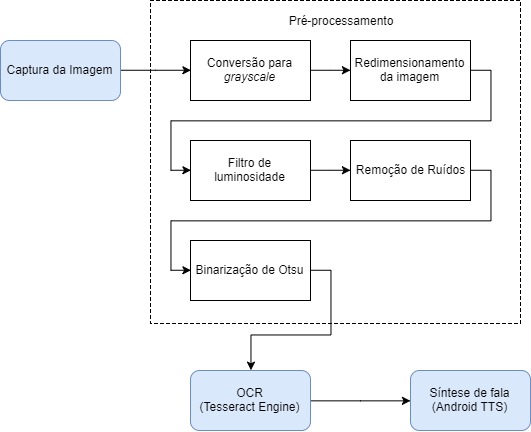
**Figura 4 - Passos da Metodologia**

Fonte: Elaborada pelo autor

## 6.1 Modelo de Arquitetura

O modelo de arquitetura para desenvolvimento de aplicativos cujo processo compreenda reconhecimento de caracteres a partir de fotografias é composto de cinco etapas distintas de processamento, de acordo com a Figura 5:

**Figura 5: Fluxo de processamento do aplicativo**

****

Fonte: Elaborada pelo autor

A primeira etapa (Captura da Imagem) consiste na obtenção da imagem por meio da câmera fotográfica, recurso normalmente disponível em smartphones e tablets. A captura pode utilizar as funções de captura nativas da plataforma Android, não sendo necessário o emprego de tecnologias adicionais.

A segunda etapa (Pré-Processamento) é a etapa da aplicação 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 no qual a imagem é redimensionada afim de atingir 300dpi, densidade na qual o Tesseract apresenta melhor precisão;
* Conversão para *grayscale*, processo no qual a imagem, até então colorida, é convertida em escala de cinza, formato no qual o Tessract apresenta melhor precisão;
* Aplicação de filtros de luminosidade e contraste, que 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;
* 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);
* Remoção de ruídos, processo que permite identificar pequenos componentes desconexos e ruídos na imagem, removendo-os;

A terceira etapa (OCR), é a etapa na qual as linhas de texto são identificadas, e os caracteres presentes são extraídos, analisados e identificados através de comparação da imagem com padrões pré-cadastrados. A tecnologia proposta é o Tesseract OCR, software de reconhecimento óptico de caracteres desenvolvido pela Hewlett Packard (HP) entre 1984 e 1994 e patrocinado oficialmente pela Google, que desde 2006 é responsável pela manutenção e distribuição do software, sob licença Apache 2.0. Em 1995, ainda em propriedade da HP, a Tesseract Engine foi considerada um dos três softwares OCR com maior precisão e desempenho (SMITH, 1995).

A quarta e última etapa (Síntese de voz), é responsável pela exibição, em áudio, do resultado do processamento OCR. TTS é a API do Android responsável pela conversão automática de texto digital editável em voz (áudio). Nas versões mais atuais do Android, já estão disponíveis as engines de voz em 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 como materialização da construção teórica, com o objetivo de validar uma aplicação prática do modelo.

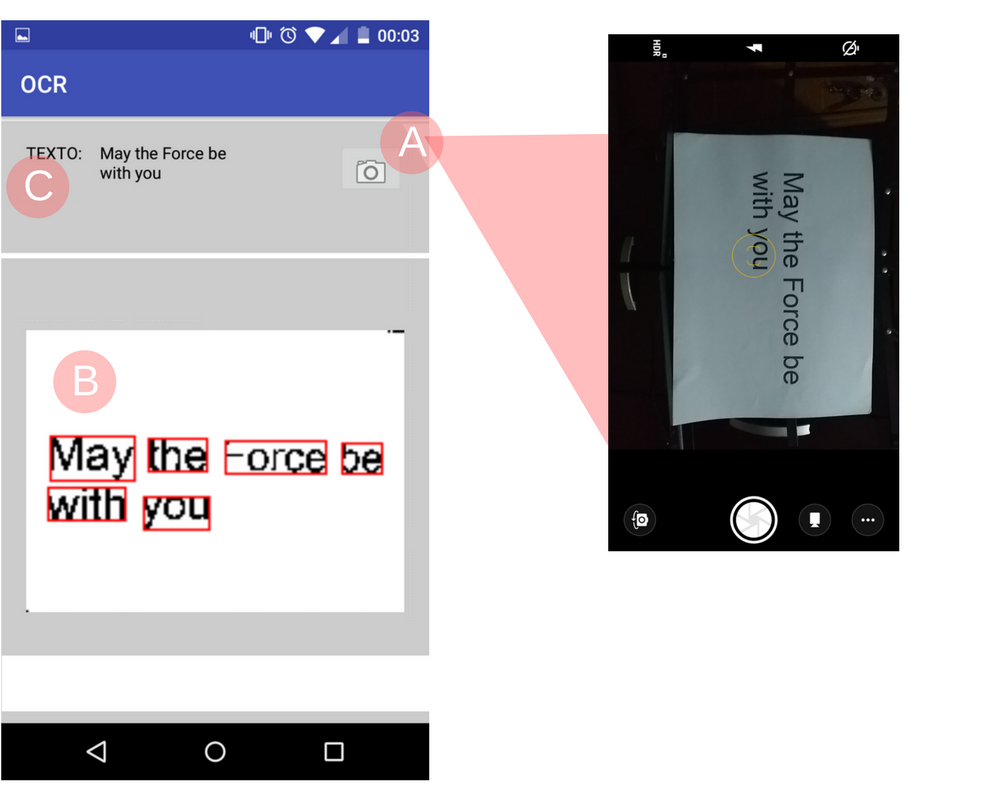
Para a construção de um aplicativo para dispositivos móveis baseado no modelo de arquitetura proposto, sugere-se, como requisitos técnicos e baseando-se apenas em tecnologias gratuitas, o emprego da Android SDK como pacote de ferramentas de desenvolvimento; a Tesseract Engine OCR (encapsulada em um *wrapper* 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, não sendo necessário o emprego de softwares adicionais.

Como requisitos funcionais da aplicação do modelo desenvolvida neste trabalho, foram consideradas a capacidade de adquirir uma imagem por meio da câmera de dispositivos móveis, a capacidade de interpretar programaticamente a figura obtida de forma a identificar e extrair quaisquer textos nela presentes, e a capacidade de reproduzir em áudio os textos identificados.

Com base nesses requisitos, foi desenvolvido um aplicativo 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 apresentada a aplicação desenvolvida, por meio da qual se aplicam as etapas previstas no modelo de arquitetura 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 06:

**Figura 6: Interface geral do aplicativo**



Fonte: Elaborada pelo autor

Na interface do aplicativo, ilustrada na Figura 06, são representadas as etapas previstas no modelo, conforme:

1. Obtenção da fotografia – Ao tocar o botão da captura, conforme ilustrado na Figura 6a, o usuário é direcionado à tela de captura de imagem, passo correspondente à etapa de “Captura da Imagem” prevista no modelo ilustrado na Figura 5. Todo o processo de captura da fotografia 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 fotografia é concluída, o aplicativo processa a imagem de acordo com os passos previstos nas etapas “Pré-Processamento” vide modelo da Figura 5, e então submete a imagem, já processada, ao Tesseract, também de acordo com o modelo (etapa “OCR”). O Tesseract então 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 6b, para apuração da manipulação da figura. O Tesseract, finalmente, extrai os caracteres identificados nessas regiões.
3. O texto extraído é enviado para síntese de voz, materializando a última etapa do modelo proposto (etapa “Síntese de Voz”, também ilustrado na Figura 5). 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 dessa aplicação prática do modelo de arquitetura proposto, foram conduzidos experimentos, aferições e análise dos resultados, conforme capítulo 6.3 a seguir.

## 6.3 Experimentos práticos

A aplicação concebida a partir do modelo proposto foi exposta a experimentos práticos afim de analisar o comportamento da ferramenta em ambientes com características favoráveis. Os mesmos testes foram aplicados a um aplicativo desenvolvido sem a aplicação do modelo, ou seja, sem 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. A iluminação do local de testes foi abundante, com a fonte de luz posicionada de forma estratégica para não provocar sombras sobre a superfície escrita, e as imagens foram capturadas de forma a não causar distorções de perspectiva.

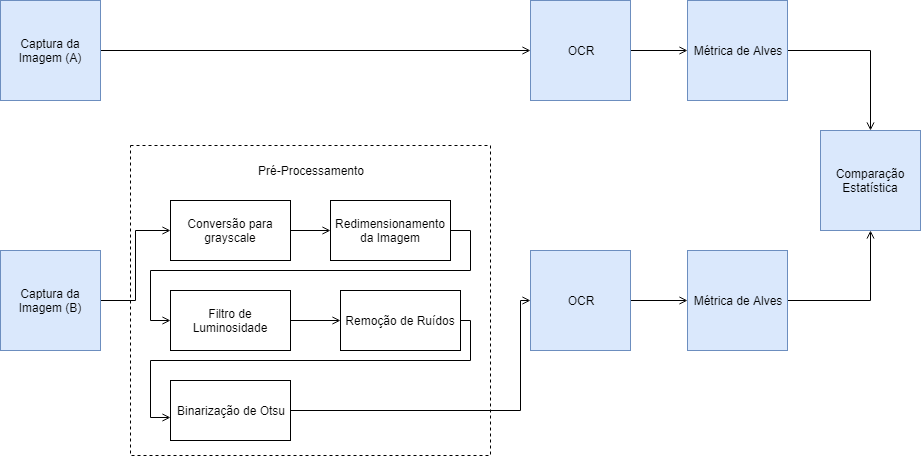
O aplicativo foi executado em um smartphone Motorola Moto G 1ª Geração, equipado com câmera traseira de 5 MegaPixels. As imagens foram capturadas em tamanho 1944 pixels de largura e 2592 pixels de altura, com 72 DPI. Os recursos de zoom e flash não foram acionados em nenhuma das capturas.

## 6.4 Avaliação dos resultados

Para mensurar e avaliar o impacto das intervenções propostas, este trabalho propõe um modelo de um estudo comparativo, baseado no trabalho de Alves (2003), onde os resultados do Tesseract OCR a partir de imagens originais são 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 6:

**Figura 7: Diagrama do processo de avaliação e comparação dos resultados de OCR**



Fonte: Elaborada pelo autor

Espera-se que o método estatístico possa comprovar que a população B (testes onde foram executadas operações de processamento de imagem antes da extração de caracteres) tende a apresentar resultados melhores que a população A. Assim, torna-se evidente que o processamento da imagem influencia positivamente na precisão dos resultados do Tesseract. INSERIR PARTE DA DRA SAMARA

## 6.5 Trabalhos futuros

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.

* Melhorar processamento de imagem com fundo complexo. Método estatístico demonstrou que a melhora de precisão ainda não atingiu o resultado esperado nesses casos.

# 7 CONCLUSÃO

Os experimentos práticos foram conduzidos em ambiente controlado com uma pluralidade de cenários. De maneira geral, os testes apontam um aperfeiçoamento considerável na precisão do processo de OCR quando aplicadas as intervenções propostas.

Explicar cenários onde a precisao aumentou muito ( textos longos ) e onde piorou (fundo complexo) E mostrar tabela de resultados.

**REFERÊNCIAS**

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

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

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.

CHAKRABORTY, P.; MALLIK A. An Open Source Tesseract based Tool for Extracting Text from Images with Application in Braille Translation for the Visually Impaired.  
**International Journal of Computer Applications**. (p. 26-32), April 2013.

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.

FIALHO, J. Informação e conhecimento acessíveis aos deficientes visuais nas bibliotecas universitárias. **Perspectivas em Ciência da Informação** 17.1 (2012): 153-168.

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.

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)

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

**International Conference on Document Analysis and Recognition**, 1995.

OCH F. J. et al. **Optical character recognition**. Estados Unidos, 2015. Acesso em: 29 abr. 2015

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.

SMITH, R. A simple and efficient skew detection algorithm via text row accumulation. **Proceedings of 3rd**

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.

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.

WOLF, Christian; JOLION, J.-M. Extraction and recognition of artificial text in multimedia documents. Pattern Analysis & Applications, v. 6, n. 4, p. 309-326, 2004.

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 (OCH, 2015) [↑](#footnote-ref-1)
2. 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-2)
3. 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-3)
4. 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-4)
5. 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-5)