



UNIVERSIDADE FEDERAL DO CEARÁ
CURSO DE ENGENHARIA DA COMPUTAÇÃO

RAPHAEL CARNEIRO DE FREITAS

**DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID QUE UTILIZA
RECONHECIMENTO ÓTICO DE CARACTERES PARA A AQUISIÇÃO DE TEXTOS
CONTIDOS EM ÁREAS DE INTERESSE DE UMA IMAGEM**

SOBRAL

2019

RAPHAEL CARNEIRO DE FREITAS

DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID QUE UTILIZA
RECONHECIMENTO ÓTICO DE CARACTERES PARA A AQUISIÇÃO DE TEXTOS
CONTIDOS EM ÁREAS DE INTERESSE DE UMA IMAGEM

Trabalho de Conclusão de Curso apresentado
ao Curso de Engenharia da Computação da
Universidade Federal do Ceará , como requisito
parcial à obtenção do grau de bacharel em
Engenharia da Computação.

Orientador: Prof. Dr. Iális Cavalcante de
Paula Júnior

SOBRAL

2019

RAPHAEL CARNEIRO DE FREITAS

DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID QUE UTILIZA
RECONHECIMENTO ÓTICO DE CARACTERES PARA A AQUISIÇÃO DE TEXTOS
CONTIDOS EM ÁREAS DE INTERESSE DE UMA IMAGEM

Trabalho de Conclusão de Curso apresentado
ao Curso de Engenharia da Computação da
Universidade Federal do Ceará , como requisito
parcial à obtenção do grau de bacharel em
Engenharia da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula
Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Rodrigues de Almeida Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. Erick Aguiar Donato
Universidade Federal do Ceará (UFC)

Aos meus pais Durval e Conceição, pelo incentivo, amor e apoio que sempre foi dado durante toda a vida. A minha Namorada Ana Clara pelo apoio emocional e companheirismo durante os momentos bons e ruins.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Iális Cavalcante, pelas orientações e pela paciência durante a elaboração deste trabalho.

Aos meus pais, Durval e Conceição por apoiarem as minhas decisões, acreditarem no meu potencial e pela motivação para os estudos.

A minha namorada Ana Clara por sempre me apoiar, me aconselhar e me motivar nos momentos difíceis.

Aos Amigos que tive o prazer de conhecer no decorrer do curso, pelos conhecimentos e momentos compartilhados.

A todos os meus familiares, que de alguma forma contribuíram para a realização deste trabalho.

"A maior recompensa para o trabalho do homem
não é o que ele ganha com isso, mas o que ele
se torna."

(John Ruskin)

RESUMO

Diversas tarefas do cotidiano, sejam elas pessoais ou profissionais podem ser automatizadas com auxílio de computadores. Uma destas é a digitação de dados para diversos fins como alimentação de sistemas, preenchimento de formulários, criação de textos, dentre outros. Para este caso, existe o reconhecimento ótico de caracteres que é capaz de extrair textos de imagens digitais e utilizá-los, ao invés de serem digitados por um usuário. Visto isso, este trabalho apresenta o desenvolvimento de uma aplicação que utiliza o reconhecimento ótico de caracteres para obter campos de texto presentes em imagens padronizadas como documentos e formulários. Em paralelo ao desenvolvimento da aplicação é feita uma análise comparativa entre as ferramentas Firebase ML kit e Google Cloud Vision, onde ambas utilizam aprendizado de máquina para realizar o reconhecimento de caracteres.

Palavras-chave: Reconhecimento Ótico de caracteres. Formulários. Aprendizado de Máquina. ML kit. Cloud Vision.

ABSTRACT

Several daily tasks, whether personal or professional, can be automated with the help of computers. One of these is data typing, used for various purposes such as feeding systems, filling in forms, creating texts among others. For this case, the optical character recognition is able to extract texts from digital images and use them, rather than being typed by a user. This paper presents the development of an application that uses optical character recognition to obtain text fields present in standardized images such as documents and forms. Parallel to the development of the application is a comparative analysis between the tools Firebase ML kit and Google Cloud Vision, where both use machine learning to perform character recognition.

Keywords: Optical Character Recognition. Forms. Machine Learning. ML kit. Cloud Vision.

LISTA DE FIGURAS

Figura 1 – Exemplo de Digitalização do Caractere A	15
Figura 2 – Processo de OCR	19
Figura 3 – Imagem Digitalizada	20
Figura 4 – Imagem Digitalizada com Limiar Fixo	20
Figura 5 – Imagem Digitalizada com Limiar Dinâmico	20
Figura 6 – Processo de segmentação	21
Figura 7 – Imagem Original (A) Imagem digitalizada com baixo valor de limiar (B) e alto valor de limiar (C)	22
Figura 8 – Suavização e normalização de caracteres	22
Figura 9 – Traços extraídos das letras F, H e N	24
Figura 10 – Imagem a ser tratada (A), Blocos de texto reconhecidos (B), Linhas de texto reconhecidas (C) e Elementos de texto reconhecidos (D)	27
Figura 11 – Diagrama de casos de uso	29
Figura 12 – Formulário utilizado e texto reconhecido	36
Figura 13 – Modelo de CNH	37
Figura 14 – Medidas da área de interesse.	38
Figura 15 – Resultados obtidos no reconhecimento de caracteres do primeiro teste . . .	40
Figura 16 – Resultados obtidos no reconhecimento de caracteres do segundo teste . . .	41
Figura 17 – Resultados obtidos no reconhecimento de caracteres do terceiro teste . . .	42
Figura 18 – Modelo de CNH com campos de interesse para realização de testes.	43
Figura 19 – Apresentação dos resultados obtidos a partir de imagem obtida da galeria. .	43
Figura 20 – Apresentação dos resultados obtidos a partir de imagem obtida da câmera do dispositivo.	44

LISTA DE TABELAS

Tabela 1	–	Requisitos funcionais	28
Tabela 2	–	Requisitos não funcionais	28
Tabela 3	–	UC01- Selecionar a imagem	29
Tabela 4	–	UC02- Selecionar área de interesse	30
Tabela 5	–	UC03 - Executar o reconhecimento OCR	30
Tabela 6	–	UC04 - Visualizar os caracteres reconhecidos	30
Tabela 7	–	Acurácia obtida no primeiro teste	41
Tabela 8	–	Acurácia obtida no segundo teste	41
Tabela 9	–	Acurácia obtida no terceiro teste	42
Tabela 10	–	Acurácia obtida no primeiro e segundo testes de reconhecimento em um formulário complexo	44
Tabela 11	–	Quantidade de erros obtidos nos testes para cada amostra.	45
Tabela 12	–	Acurácia obtida nos testes com CNHs reais com base no melhor resultado	45
Tabela 13	–	Acurácia obtida nos testes com CNHs reais com base na média de erros	45

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicativos
CNH	Carteira Nacional de Habilitação
EM	Eletromagnético
OCR	Reconhecimento Ótico de Caracteres
PDI	Processamento Digital de Imagens
REST	Transferência de Estado Representacional
SDK	Kit de Desenvolvimento de Software

LISTA DE SÍMBOLOS

γ_{acc}	Acurácia obtida
ω	Quantidade de caracteres que devem ser reconhecidos
ω_{err}	Quantidade de caracteres reconhecidos de forma incorreta

LISTA DE CÓDIGOS-FONTE

Código-fonte 1	– Função para obter imagem da galeria.	31
Código-fonte 2	– Função para obter imagem a partir da câmera.	32
Código-fonte 3	– Função de reconhecimento de caracteres utilizando o ML kit.	33
Código-fonte 4	– Função de reconhecimento de caracteres utilizando o Cloud Vision. .	35
Código-fonte 5	– Exemplo de utilização do método split.	36
Código-fonte 6	– Exemplo de utilização do método createBitmap.	38

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Trabalhos Relacionados	16
1.2	Motivação	16
1.3	Objetivos	17
1.3.1	<i>Objetivos Específicos</i>	17
1.4	Organização	17
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	Metodologia de OCR	19
2.2	Escaneamento Ótico	19
2.3	Localização e Segmentação	21
2.4	Pré-Processamento	22
2.5	Extração de Características	22
2.5.1	<i>Distribuição dos Pontos</i>	23
2.5.2	<i>Transformadas e Expansão de séries</i>	23
2.5.3	<i>Análises Estruturais</i>	24
2.6	Classificação	24
2.6.1	<i>Métodos de Teoria da Decisão</i>	24
2.6.2	<i>Métodos Estruturais</i>	25
2.7	Pós-Processamento	25
2.7.1	<i>Agrupamento</i>	25
2.7.2	<i>Detecção e Correção de Erros</i>	26
2.8	Firestore MLkit	26
2.9	Google Cloud Vision	27
3	METODOLOGIA	28
3.1	Principais Requisitos do problema a ser trabalhado	28
3.2	Casos de uso	28
3.2.1	<i>Selecionar a imagem</i>	29
3.2.2	<i>Selecionar área de interesse</i>	29
3.2.3	<i>Executar o reconhecimento OCR</i>	30
3.2.4	<i>Visualizar os caracteres reconhecidos</i>	30

3.3	Planejamento de integração	31
3.4	Desenvolvimento do Software	31
3.4.1	<i>Desenvolvimento da primeira parte da aplicação</i>	31
3.4.1.1	<i>Implementação do Firebase ML Kit</i>	33
3.4.1.2	<i>Implementação do Google Cloud Vision</i>	35
3.4.2	<i>Desenvolvimento da segunda parte da aplicação</i>	36
3.4.2.1	<i>Adaptação para imagens padronizadas que apresentam campos complexos</i>	37
4	RESULTADOS	40
4.1	Testes das ferramentas de OCR ML kit e Cloud Vision	40
4.1.1	<i>Primeiro Teste</i>	40
4.1.2	<i>Segundo Teste</i>	41
4.1.3	<i>Terceiro Teste</i>	42
4.2	Testes de reconhecimento de texto em imagens padronizadas que apresentam campos complexos	42
4.2.1	<i>Primeiro teste da Aplicação</i>	43
4.2.2	<i>Segundo teste da Aplicação</i>	44
4.2.3	<i>Terceiro teste da Aplicação</i>	44
5	CONCLUSÕES	46
5.0.1	<i>Trabalhos Futuros</i>	46
	REFERÊNCIAS	48

1 INTRODUÇÃO

Uma imagem digital é composta de um número finito de elementos conhecidos como elementos pictóricos, elementos de imagem, pels ou pixels e cada um possui localização e valores específicos. Isso possibilita o processamento da mesma por um computador digital, o que é chamado de Processamento Digital de Imagens (PDI). Diferente da visão humana, que é limitada à banda visual do Espectro Eletromagnético (EM), os aparelhos de PDI cobrem quase todo o espectro EM, variando de ondas gama a ondas de rádio. Dessa forma PDI inclui um amplo e variado campo de aplicações (GONZALEZ; WOODS, 2010).

Uma das aplicações para PDI é o Reconhecimento Ótico de Caracteres (OCR) que consiste em transformar uma imagem digitalizada em texto. Quando uma página é digitalizada, ela é armazenada como um arquivo mapeado por bit no formato TIF no momento em que a mesma é exibida na tela, pode ser lida por um ser humano, mas para o computador é apenas uma série de pontos pretos e brancos (zeros e uns) como observado na Figura 1, ou seja, ele não reconhece palavras na imagem. Para que ocorra tal reconhecimento pela máquina, o OCR analisa cada linha da imagem e tenta determinar se os pontos pretos e brancos representam uma determinada letra ou número (LEGAL SCANS, 2018).

Figura 1 – Exemplo de Digitalização do Caractere A



Fonte: (LEGAL SCANS, 2018)

Atualmente, existem muitos sistemas comerciais e gratuitos disponíveis para uma variedade de aplicações. Destaca-se desde a transformação de textos originalmente em papel para um formato digital sem a necessidade de digitação, até a utilização do OCR por radares de trânsito para identificação de placas de automóveis e sua verificação junto às bases de dados municipais, estaduais ou federais, referentes a situação do veículo, ou até mesmo para identificação de veículos roubados ou com pendências judiciais (G1, 2010).

1.1 Trabalhos Relacionados

O Trabalho de (AZEVEDO, 2018), aborda o funcionamento de duas ferramentas de OCR, uma livre (gImageReader) e uma comercial (ABBYY FlexiCapture), fazendo uma análise comparativa e avaliando a eficiência de ambas no reconhecimento de caracteres de 15 prontuários médicos.

Em (ARAUJO, 2017), é feita uma análise do impacto de um modelo de classificação, seguido de técnicas de PDI e OCR para extração de texto de cupons fiscais, classificando-os em subgrupos.

Em (MEDEIROS, 2009), são descritas técnicas, bem como a especificação e implementação de uma ferramenta que utiliza OCR para extrair os caracteres da placa de um veículo através de uma imagem digital.

Em (HERNANDEZ, 2018), é projetado e implementado um sistema, que através da tecnologia OCR é capaz de detectar e interpretar informações textuais presentes em imagens salvas em um dispositivo Android.

Em (FAHLEN, 2019), é realizado um estudo sobre OCR e implementado um aplicativo Android utilizando o reconhecimento de caracteres através do Google Cloud Vision para a leitura de informações de formulários como gabaritos ou questionários.

1.2 Motivação

A tecnologia OCR é uma ferramenta útil em que todo documento digitalizado é convertido e tratado como um documento de texto. Isso permite a utilização de algumas funções no arquivo, como por exemplo pesquisar por um determinado conteúdo utilizando palavras-chaves ou trechos através do mecanismo de busca do software definido para leitura do texto. Ou no caso de um editor de texto, se torna possível formatar e editar esse arquivo, achar rapidamente as partes desejadas para coletar os dados necessários, agilizando desta forma a busca e entrada de dados (NEOMIND, 2018).

Observa-se a funcionalidade do OCR, quando um usuário necessita procurar uma certa informação em uma imagem digital que não tenha sido convertida em texto. Na mesma será necessário percorrer o arquivo inteiro sem ter uma maneira de buscar rapidamente o dado desejado, isso porque o documento será interpretado pelo computador apenas como uma imagem e não um texto, fato este que acaba demandando um maior tempo e esforço do usuário.

Quanto a aplicação de tal tecnologia, existe uma ampla utilização na área comercial como em máquinas industriais, escritórios que necessitam obter o texto de documentos impressos, empresas de correspondências que reconhecem os endereços de pacotes através de OCR e criam rotas de entrega, dentre outras. Além das aplicações comerciais, há o uso da tecnologia na área social ou educacional. As aplicações de OCR com cegos e deficientes visuais permitem a digitalização de texto impresso e o seu reconhecimento como é habitual, mas, em seguida, adiciona a capacidade de ter um sintetizador, de modo que o texto reconhecido possa ser falado numa voz sintética (KNOOW, 2016).

Outra grande utilidade da tecnologia OCR é que a mesma tem as suas aplicações para que um usuário qualquer de uma máquina, ou um colaborador em uma empresa possa aumentar sua produtividade, visto que documentos como cartão de visita, artigos de revistas e documentos formais, podem ser digitalizados e reconhecidos para eliminar assim a necessidade de escrever esta informação manualmente, permitindo também a busca, modificação e reutilização da informação existente.

1.3 Objetivos

Utilizar a tecnologia de reconhecimento de caracteres no desenvolvimento de um aplicativo mobile capaz de reconhecer e extrair textos contidos em campos de interesse de uma imagem digital.

1.3.1 *Objetivos Específicos*

- Aplicar as técnicas e ferramentas de processamento de imagem ao desenvolvimento da aplicação.
- Avaliar as ferramentas Firebase ML kit e Google Cloud Vision e realizar uma análise comparativa entre elas.
- Utilizar e avaliar o desempenho da ferramenta Cloud Vision no processo de reconhecimento de caracteres implementado na aplicação.

1.4 Organização

Este trabalho está disposto da seguinte forma:

- Capítulo 2: Neste capítulo serão apresentados os conceitos e finalidades dos procedimentos

e técnicas abordados neste trabalho.

- Capítulo 3: Neste capítulo serão apresentados a metodologia abordada no desenvolvimento da aplicação proposta de forma detalhada, assim como os experimentos a serem realizados.
- Capítulo 4: São apresentados os resultados alcançados dos experimentos realizados.
- Capítulo 5: Trata-se das conclusões deste estudo e os trabalhos futuros a serem realizados em decorrência deste.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Metodologia de OCR

De acordo com (EIKVIL, 1993), um sistema OCR é composto por: escaneamento ótico, localização e segmentação, pré-processamento, extração de características, classificação e pós-processamento.

Resumidamente, o processo de reconhecimento de caracteres, como o observado na Figura 2, acontece da seguinte forma: um scanner ótico ou câmera digitaliza um determinado documento. Ao obter a imagem, ocorre a busca por partes do documento que contenham textos, extraíndo assim os símbolos, cada um destes passa por um processamento que elimina todo o ruído possível com o objetivo de facilitar a visualização das características. O símbolo obtido é comparado com as amostragens onde é encontrada a melhor correspondência que por sua vez é convertida no seu texto correspondente (ANDRADE, 2016).

Figura 2 – Processo de OCR



fonte: (EIKVIL, 1993)

2.2 Escaneamento Ótico

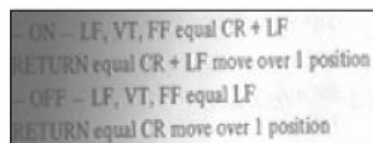
Em um sistema de OCR são utilizados câmeras ou scanners óticos, que consistem em dispositivos que, através de um sensor, obtém uma imagem digital do documento original. Documentos impressos geralmente consistem em impressão em preto sobre fundo branco, portanto, ao executar o OCR, é comum a prática de converter uma imagem colorida em uma imagem de dois níveis (preto e branco). Esse processo é conhecido como limiar, ele muitas vezes é executado no scanner para economizar espaço de memória e esforço computacional (EIKVIL, 1993).

O processo de limiarização ou thresholding é importante, pois os resultados do

reconhecimento do texto são totalmente dependentes da qualidade da imagem de dois níveis. O limiar realizado no scanner é geralmente muito simples. Um limite fixo é usado, onde os níveis de cinza abaixo deste limiar são considerados pretos e os níveis acima são considerados brancos, para um documento de alto contraste com fundo uniforme, um limiar fixo pré-definido pode ser suficiente.

No entanto, muitos casos encontrados na prática, como imagens de objetos variados que contém texto, obtidas através de câmeras por exemplo, têm um intervalo bastante grande em contraste, sendo necessário nesses casos, métodos mais sofisticados com a utilização de limiar variável ou dinâmico. Nestas situações, o mesmo é alterado dinamicamente conforme a necessidade, tornando o processo mais sofisticado e exigindo maiores recursos computacionais (EIKVIL, 1993). Na figura 3, pode-se observar a imagem digitalizada em tons de cinza com alguns ruídos de iluminação.

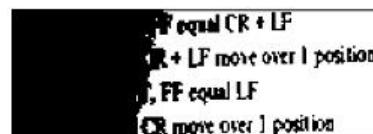
Figura 3 – Imagem Digitalizada



fonte: (EIKVIL, 1993)

Na Figura 4, pode-se observar a imagem obtida através do processo de digitalização com limiar fixo, onde parte do texto foi comprometido.

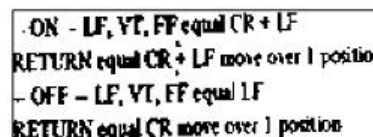
Figura 4 – Imagem Digitalizada com Limiar Fixo



fonte: (EIKVIL, 1993)

Na Figura 5, pode-se observar a imagem obtida através do processo de limiar dinâmico, onde é reconhecido o nível de limiar mais eficiente para o caso, e possibilita a maior eficiência no reconhecimento do texto presente na imagem.

Figura 5 – Imagem Digitalizada com Limiar Dinâmico



fonte: (EIKVIL, 1993)

2.3 Localização e Segmentação

Em uma aplicação OCR, é necessário localizar as regiões da imagem digital onde estão presentes os dados textuais (área de interesse) e distingui-los de figuras e gráficos, como por exemplo, ao realizar a digitalização de um cartão de visitas, o nome da empresa, endereço e telefone devem ser localizados e separados de outros gráficos como o logotipo da empresa antes do reconhecimento.

A segmentação subdivide uma imagem em regiões ou objetos que a compõem (conjunto de pixels). O nível de detalhe em que a subdivisão é realizada depende do problema a ser resolvido, ou seja, a segmentação deve parar quando os objetos ou as regiões de interesse de uma aplicação forem detectados (GONZALEZ; WOODS, 2010).

Aplicada ao reconhecimento de texto, a segmentação é o isolamento de caracteres ou palavras, logo, a maioria dos algoritmos de reconhecimento de caracteres óticos segmentam as palavras em caracteres isolados que são reconhecidos individualmente (EIKVIL, 1993). A Figura 6 ilustra os processos de localização e segmentação de imagens.

Figura 6 – Processo de segmentação



Fonte: Elaborada pelo Autor.

Um problema que ocorre no processo de segmentação é que muitas vezes os caracteres não são reconhecidos por conta de textos e gráficos com caracteres muito unidos. Se o documento foi digitalizado com o limiar muito baixo, ocorre muitas vezes a perda de caracteres e espaços. Se o limiar for muito alto, o sistema de OCR também pode ignorar espaços ou unir caracteres aos gráficos (MITH, 2013). O problema com segmentação de imagens relacionado ao valor do limiar pode ser observado na Figura 7, onde pode-se perceber a perda de dados textuais.

Figura 7 – Imagem Original (A) Imagem digitalizada com baixo valor de limiar (B) e alto valor de limiar (C)



Fonte: Elaborada pelo Autor.

2.4 Pré-Processamento

A imagem resultante do processo de digitalização pode conter ruído, dependendo da resolução da imagem digital obtida e do sucesso da técnica aplicada para o limiar, os caracteres podem ficar deformados ou quebrados. Alguns desses defeitos podem posteriormente causar baixas taxas de reconhecimento. Os mesmos podem ser eliminados usando um pré-processamento para suavizar as características dos caracteres digitalizados.

A remoção de ruídos ou suavização, envolve, entre outras técnicas de PDI, o preenchimento e afinamento: no preenchimento, pequenos espaços, lacunas e buracos nos caracteres digitalizados são removidos; já no afinamento, a largura da linha é reduzida. Além de suavizar, o pré-processamento geralmente inclui a normalização, que é aplicada para obter caracteres de tamanho uniforme, corrigindo também a inclinação e rotação. Para poder corrigir a rotação, o ângulo de rotação deve ser encontrado (EIKVIL, 1993). Na Figura 8 pode-se observar a suavização e normalização de um caractere.

Figura 8 – Suavização e normalização de caracteres



Fonte: (EIKVIL, 1993).

2.5 Extração de Características

Para (EIKVIL, 1993), o objetivo da extração de características é capturar as características essenciais dos símbolos, que é geralmente aceito como um dos problemas mais difíceis do reconhecimento de padrões. A maneira mais direta de descrever um caractere é pela imagem raster real, que é formada por pixels ou pontos. Outra abordagem é extrair certos recursos que ainda caracterizam os símbolos, mas deixam de fora os atributos sem importância. As técnicas

para extração de tais características são frequentemente divididas em três grupos principais, onde os recursos são encontrados a partir de: distribuição dos pontos, transformadas e expansão de séries ou análises estruturais.

2.5.1 Distribuição dos Pontos

Esta categoria abrange técnicas que extraem recursos com base na distribuição estatística de pontos, esses recursos geralmente são tolerantes a distorções e variações de estilo. Algumas das técnicas típicas nessa área são:

- **Zoneamento** – Onde o retângulo que circunscreve o caractere é dividido em várias regiões sobrepostas, ou não sobrepostas, e as densidades de pontos pretos nessas regiões são calculadas e usadas como recursos.
- **Ocasão** – A ocasião em que se encontram pontos pretos sobre um centro escolhido, por exemplo, o centro de gravidade, ou um sistema de coordenadas escolhido, são usados como recursos.
- **Cruzamentos e distâncias** - Na técnica de cruzamento, características são encontradas a partir do número de vezes que a forma do caractere é cruzada por vetores ao longo de certas direções. Ao usar a técnica de distância, certos comprimentos ao longo dos vetores que cruzam a forma do caractere são medidos. Por exemplo, o comprimento dos vetores dentro do limite do caractere.
- **N-tuplas** - A ocorrência conjunta relativa de pontos pretos e brancos (primeiro plano e plano de fundo) em determinadas ordenações especificadas é usada como recurso.

2.5.2 Transformadas e Expansão de séries

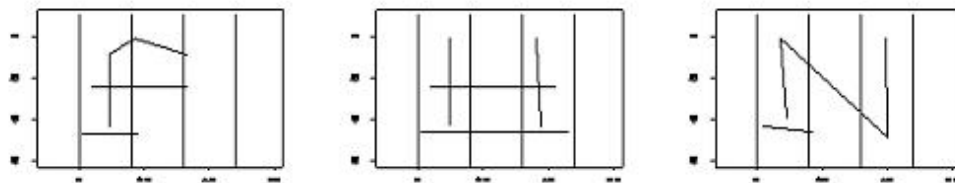
Tais técnicas ajudam a reduzir a dimensionalidade do vetor de características e as características extraídas podem se tornar invariantes a deformações globais como translação e rotação. Neste processo, podem ser usadas as transformadas de Fourier, Walsh, Haar, Hadamard, Karhunen-Loeve, Hough, transformada do eixo principal etc (EIKVIL, 1993).

Muitas dessas transformações são baseadas na curva que descreve o contorno dos caracteres. Isso significa que esses recursos são muito sensíveis ao ruído que afeta o mesmo como falhas não intencionais no contorno, por exemplo.

2.5.3 Análises Estruturais

Na análise estrutural, os recursos que descrevem as estruturas geométricas e topológicas de um símbolo são extraídos, e através destes, tenta-se descrever a composição física do caractere, e alguns dos recursos comumente usados são traços, pontos finais, interseções entre linhas e loops. Em comparação com outras técnicas, a análise estrutural fornece características com alta tolerância a variações de ruído e estilo. No entanto, os recursos são apenas moderadamente tolerantes à rotação e à tradução (EIKVIL, 1993). Um exemplo de análise estrutural pode ser observado na figura 9.

Figura 9 – Traços extraídos das letras F, H e N



Fonte: (EIKVIL, 1993).

2.6 Classificação

A classificação é o processo de identificar cada caractere e atribuir a ele a classe de caracteres correta. São utilizadas duas abordagens diferentes para classificação no reconhecimento de caracteres; O reconhecimento por teoria da decisão e o reconhecimento por métodos estruturais.

2.6.1 Métodos de Teoria da Decisão

As principais abordagens para o reconhecimento de caracteres utilizando teoria da decisão, segundo (EIKVIL, 1993), são:

- Classificação por Correspondência – Abrange os grupos de técnicas com base em medidas de similaridade, onde a distância entre o caractere e a sua classe correspondente é calculada. Diferentes medidas podem ser usadas, mas o mais comum é utilizar a distância euclidiana.
- Classificação por Estatística - Segundo (CHERIET, 2007), os métodos estatísticos são baseados na teoria de decisão de Bayes, que procura minimizar a perda de classificação quando dada uma matriz de perdas e probabilidades estimadas. De acordo com a aborda-

gem de estimação de densidade probabilística, os métodos de classificação estatística são divididos em parametrizados e não parametrizados.

- **Classificação por Redes Neurais** - Redes neurais podem ser utilizadas no reconhecimento de caracteres. Considerando-se uma rede do tipo back-propagation, composta de várias camadas de elementos interligados, um vetor é inserido na camada de entrada, cada elemento da camada calcula uma soma ponderada das suas entradas e a transforma em uma saída por uma função não linear. Durante o processo, os pesos em cada ligação são ajustados até se obter um resultado desejado.

2.6.2 Métodos Estruturais

Dentro da área de reconhecimento estrutural, os métodos sintáticos estão entre os mais abordados. Neles, as medidas de similaridade baseadas em relações entre componentes estruturais podem ser formuladas usando conceitos gramaticais. A ideia é que cada classe tenha sua própria gramática definindo a composição do caractere. Uma gramática pode ser representada como strings ou árvores, e os componentes estruturais extraídos de um caractere desconhecido são comparados com as gramáticas de cada classe. Supondo que existam duas classes de caracteres diferentes que podem ser geradas pelas duas gramáticas G1 e G2, respectivamente. Dado um caráter desconhecido, diz-se que é mais semelhante com a primeira classe, se pode ser gerado por G1, mas não por G2.

2.7 Pós-Processamento

2.7.1 Agrupamento

O resultado do reconhecimento de caracteres é um conjunto de símbolos individuais convertidos em texto. No entanto, esses símbolos em si geralmente não contêm informações suficientes, mas de associados, acabam compondo palavras e números. O processo de executar essa associação de símbolos em cadeias de caracteres é chamado de agrupamento, sendo este baseado na localização dos símbolos no documento, ou seja, os símbolos que são suficientemente próximos serão agrupados (LINS, 2012).

Para fontes com espaçamento fixo, o processo de agrupamento é bastante fácil, pois a posição de cada caractere é conhecida. Para caracteres digitalizados, a distância entre os caracteres é variável. No entanto, a distância entre as palavras geralmente é significativamente

maior do que a distância entre os caracteres, e o agrupamento ainda é possível. Os problemas reais ocorrem para caracteres manuscritos ou quando o texto está distorcido (EIKVIL, 1993).

2.7.2 Detecção e Correção de Erros

Em problemas avançados de reconhecimento ótico de texto, um sistema que consiste apenas em reconhecimento de caractere único não será suficiente, visto que mesmo os melhores sistemas de reconhecimento não fornecem a identificação correta de todos os caracteres analisados. Para contornar tal problema, alguns desses erros podem ser detectados ou mesmo corrigidos pelo uso do contexto (LINS, 2012).

Existem duas abordagens principais, onde a primeira utiliza a possibilidade de sequências de caracteres que aparecem juntos. Isso pode ser feito pelo uso de regras que definem a sintaxe da palavra, dizendo, por exemplo, que após um período geralmente deve haver uma letra maiúscula. Além disso, para diferentes idiomas, as probabilidades de dois ou mais caracteres aparecerem juntos em uma sequência podem ser calculados e podem ser utilizados para detectar erros. Por exemplo, no idioma português, a probabilidade de um “n” aparecer antes um “p” em uma palavra é zero, e se tal combinação for detectada, um erro é assumido .

2.8 Firebase MLkit

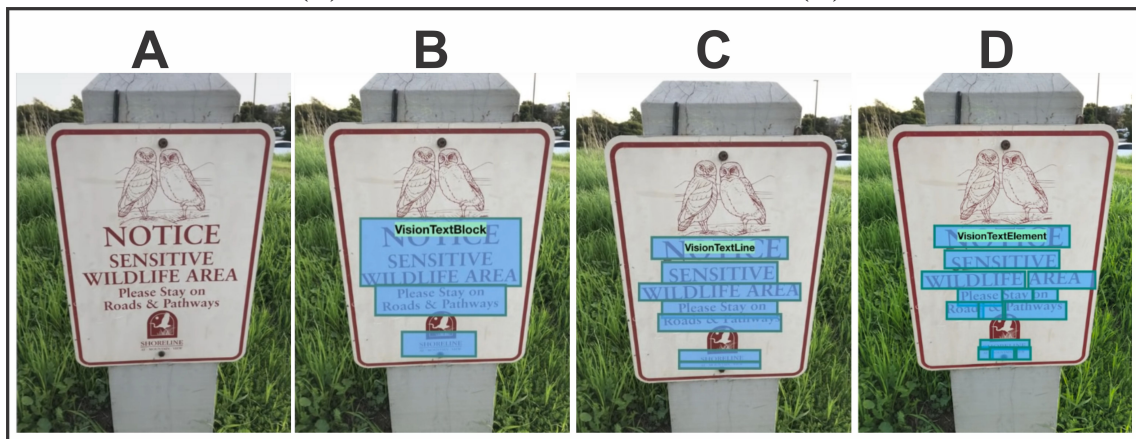
A Firebase consiste em uma plataforma de desenvolvimento mobile e web de propriedade da Google. Seu foco é ser uma ferramenta de back-end completa e de fácil usabilidade, disponibilizando diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicativos (MICREIROS, 2016).

Dentre as ferramentas disponíveis na plataforma, existe o ML kit, ou Machine Learning Kit, um Kit de Desenvolvimento de Software (SDK) que tem o objetivo de levar a experiência do aprendizado de máquina do Google para aplicativos Android e iOS. Tal ferramenta possibilita a utilização de reconhecimento de texto, detecção facial, leitura de código de barras, categorização de imagens, reconhecimento de pontos de referência e a utilização dos próprios modelos de aprendizado de máquina criados pelo usuário (CONCRETE, 2018).

Para executar o reconhecimento de texto no ML kit, é necessário criar um objeto a partir de uma matriz de bytes, bitmap ou arquivo salvo no dispositivo e em seguida transferir tal objeto para o método de processamento de imagem, retornando assim blocos de texto presente

na mesma. Cada bloco de texto (text block) representa um bloco retangular de texto que contém zero ou mais objetos linha (line), que contém zero ou mais objetos elementos (elements), que representam as palavras. Para cada objeto citado anteriormente, é possível acessar o texto reconhecido na região e suas coordenadas delimitadoras (FIREBASE, 2019). Um exemplo de reconhecimento de blocos, linhas e elementos de texto pode ser observado na figura 10.

Figura 10 – Imagem a ser tratada (A), Blocos de texto reconhecidos (B), Linhas de texto reconhecidas (C) e Elementos de texto reconhecidos (D)



Fonte: Elaborada pelo Autor.

2.9 Google Cloud Vision

O Cloud Vision encapsula modelos avançados de aprendizado de máquina em uma Interface de Programação de Aplicativos (API) com arquitetura de Transferência de Estado Representacional (REST) que permite aos desenvolvedores entender o conteúdo de imagens. Essa API é capaz de classificar imagens em categorias, detectar objetos e rostos individuais e extrair palavras contidas nas imagens. Com isso é possível criar metadados em catálogos de imagens, moderar conteúdos ofensivos e possibilitar novos cenários de marketing utilizando a análise de sentimento das imagens (GOOGLE CLOUD, 2019).

Nesta API, existe o recurso de detecção de caracteres onde um arquivo é obtido contendo uma string com o texto extraído da imagem, bem como cada palavra e suas caixas delimitadoras. Existe também um recurso de detecção de textos otimizado para documentos densos, onde o arquivo obtido inclui informações da imagem, blocos de texto, parágrafos, palavras e quebras de linha (GOOGLE CLOUD, 2019).

3 METODOLOGIA

Neste capítulo, serão discutidos os requisitos do sistema, seus casos de uso e as ferramentas e métodos utilizados para a implementação das funcionalidades da aplicação de que se trata este trabalho.

3.1 Principais Requisitos do problema a ser trabalhado

Nesta seção são levantados os requisitos da aplicação. Os requisitos Funcionais, descrevem explicitamente as funcionalidades e serviços do sistema, eles podem ser observados na Tabela 1.

Tabela 1 – Requisitos funcionais

REQUISITOS FUNCIONAIS
RF01: Recuperar uma imagem que contenha um texto
RF02: Identificar as regiões de interesse
RF03: Realizar a identificação dos caracteres
RF04: Apresentar os caracteres reconhecidos

Fonte: elaborado pelo autor.

Os requisitos não funcionais definem as propriedades e restrições da aplicação, eles podem ser observados na tabela 1.

Tabela 2 – Requisitos não funcionais

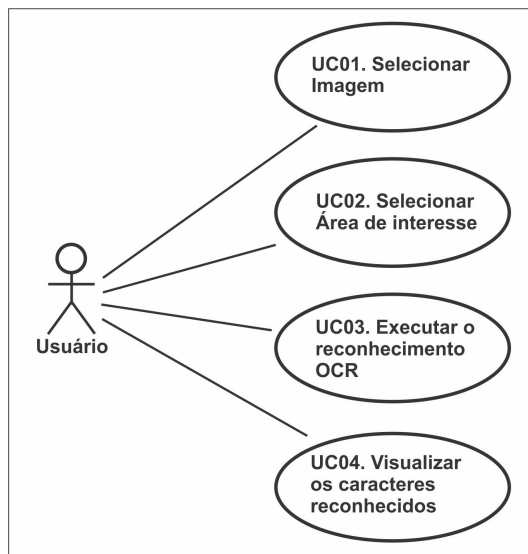
REQUISITOS NÃO FUNCIONAIS
RNF01: Implementar a ferramenta para a plataforma Android
RNF02: Utilizar a biblioteca Crop Image para cortar as imagens obtidas.

Fonte: elaborado pelo autor.

3.2 Casos de uso

A aplicação de reconhecimento de texto possui quatro casos de uso conforme a Figura 11, sendo que seguindo uma seqüência lógica, o primeiro, o segundo e o terceiro casos de uso devem ser obrigatoriamente executados pelo usuário. No quarto caso de uso, o usuário obtém o resultado do processamento.

Figura 11 – Diagrama de casos de uso



Fonte: Elaborada pelo Autor.

3.2.1 Selecionar a imagem

O primeiro caso de uso descreve como o usuário seleciona a imagem que contém o texto a ser reconhecido. Seus requisitos, pré-condições, cenário principal e pós-condições podem ser observados na Tabela 3.

Tabela 3 – UC01- Selecionar a imagem

Requisitos Atendidos	RF01
Pré-condições	O usuário possuir uma imagem em formato jpg ou bmp
Cenário principal	1. Usuário solicita a tela de seleção de imagens. 2. A Aplicação apresenta uma tela para que o usuário selecione a imagem a ser reconhecida da galeria ou use a câmera para obter a mesma. 3. O usuário escolhe a imagem. 4. O sistema valida o formato do arquivo selecionado.
Pós-condições	A aplicação exibe a imagem selecionada.

Fonte: elaborado pelo autor.

3.2.2 Selecionar área de interesse

O segundo caso de uso descreve como o usuário seleciona a área de interesse da imagem, diminuindo as dimensões da imagem e otimizando o processamento. Seus requisitos, pré-condições, cenário principal e pós-condições podem ser observados na Tabela 4.

Tabela 4 – UC02- Selecionar área de interesse

Requisitos Atendidos	RF02
Pré-condições	O usuário ter selecionado uma imagem.
Cenário principal	1. Usuário seleciona a região na qual deseja realizar o reconhecimento de texto através de toques na tela. 2. O usuário tem a opção de rotacionar a imagem.
Pós-condições	A Aplicação exibe a região selecionada da imagem.

Fonte: elaborado pelo autor.

3.2.3 Executar o reconhecimento OCR

O terceiro caso de uso descreve a ação que o usuário realiza para que o reconhecimento seja realizado. Seus requisitos, pré-condições, cenário principal, exceções e pós-condições podem ser observados na Tabela 5.

Tabela 5 – UC03 - Executar o reconhecimento OCR

Requisitos Atendidos	RF03
Pré-condições	O usuário ter selecionado a área de interesse da imagem.
Cenário principal	1. O usuário confirma a seleção da área de interesse na imagem. 2. Usuário espera o reconhecimento ser executado.
Exceção	Caso não sejam detectados caracteres, é informada uma mensagem de erro ao usuário.
Pós-condições	A aplicação exibe uma tela com os caracteres reconhecidos.

Fonte: elaborado pelo autor.

3.2.4 Visualizar os caracteres reconhecidos

No terceiro caso de uso, o usuário visualiza o reconhecimento podendo assim confirmar os dados obtidos ou alterar os mesmos. Seus requisitos, pré-condições e cenário principal podem ser observados na Tabela 6.

Tabela 6 – UC04 - Visualizar os caracteres reconhecidos

Requisitos Atendidos	RF04
Pré-condições	O usuário ter executado o reconhecimento OCR.
Cenário principal	1. O usuário visualiza os caracteres reconhecidos.

Fonte: elaborado pelo autor.

3.3 Planejamento de integração

O planejamento de integração foi desenvolvido respeitando os requisitos do sistema e baseados na ideia de que o reconhecimento de caracteres aplicado em uma imagem é um algoritmo de uso geral, ou seja, ele pode ser utilizado tanto para extrair os dados contidos em um formulário qualquer quanto para extrair as informações de uma placa de veículo, por exemplo. Contudo, considerando que as imagens a serem analisadas são padronizadas, é possível modificar as características da aplicação para cada caso distinto, visando obter melhora nos resultados do reconhecimento do texto.

3.4 Desenvolvimento do Software

Para o desenvolvimento da aplicação foi utilizado o ambiente de programação Android Studio com linguagem de programação JAVA. O desenvolvimento da aplicação ocorreu em duas etapas; Na primeira, foram implementadas as funções básicas para que a partir de uma imagem fosse retornado o texto contido nela, e assim possibilitar a realização de testes nas ferramentas ML Kit e Cloud Vision. Na segunda etapa, foram implementadas funções para extrair partes da imagem e assim aplicar as funções de OCR e obter o texto presente nelas, o que possibilita o reconhecimento de textos contidos em imagens padronizadas.

3.4.1 Desenvolvimento da primeira parte da aplicação

Na primeira parte da aplicação foi realizada a criação da tela principal, implementada a função de obter a imagem da galeria ou da câmera do dispositivo e implementada a função de reconhecimento de caracteres para que assim as ferramentas ML kit e Cloud Vision pudessem ser comparadas.

Para obter a imagem da galeria foi utilizada uma *intent*, que segundo (ANDROID-PRO, 2019) são mensagens assíncronas que permitem que os componentes de um aplicativo solicitem a funcionalidade de outros componentes do Android, sendo neste caso solicitada uma funcionalidade de acesso à galeria e recuperação de uma imagem. A função utilizada para obter uma imagem da galeria está disponível no código-fonte 1.

Código-fonte 1 – Função para obter imagem da galeria.

```
1 private void obterDaGaleria() {
```



```

2      //intent para obter imagem da galeria
3      Intent intent = new Intent(Intent.ACTION_PICK);
4      intent.setType("image/*");
5      startActivityForResult(intent,
6          IMAGE_PICK_GALLERY_CODE);
    }

```

Para obter a imagem a partir da câmera, foi utilizada uma intent que solicita a funcionalidade da câmera do dispositivo, permitindo que o usuário capture uma imagem, salve a mesma no dispositivo e a utilize na aplicação. A função utilizada para obter uma imagem a partir da câmera está disponível no código-fonte 2.

Código-fonte 2 – Função para obter imagem a partir da câmera.

```

1  private void obterDaCamera() {
2      ContentValues values = new ContentValues();
3      values.put(MediaStore.Images.Media.TITLE, "NovaImg")
4          ; //Titulo da nova imagem
5      values.put(MediaStore.Images.Media.DESCRPTION, "
6          Imagem para texto"); //descricao
7      image_uri=getContentResolver().insert(MediaStore.
8          Images.Media.EXTERNAL_CONTENT_URI, values);
9      //Intent da camera
10     Intent cameraIntent = new Intent(MediaStore.
11         ACTION_IMAGE_CAPTURE);
12     cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT,
13         image_uri);
14     startActivityForResult(cameraIntent,
15         IMAGE_PICK_CAMERA_CODE);
16 }

```

Após implementar a função de obter a imagem, foi implementada a função de cortar a mesma para que o usuário selecione apenas a parte que lhe interessa. Para isso, foi utilizada a

biblioteca Crop Image que envia a imagem para um ambiente que permite ao usuário rotacionar a imagem se for necessário e redimensionar a imagem através de linhas de guia horizontais e verticais, dimensionando as mesmas através de toques na tela do dispositivo.

Para realizar o procedimento de reconhecimento de caracteres nas imagens obtidas, foram utilizados dois métodos distintos para que após a implementação dos mesmos, fossem testados e assim comprovado qual se apresentava mais eficiente. No primeiro método, foi utilizando o SDK Firebase ML kit e no segundo foi utilizada a API do Google Cloud Vision, onde em ambos os casos foram feitas no código fonte as devidas importações, implementações e procedimentos para que a aplicação tivesse suporte às funções de PDI e OCR.

3.4.1.1 Implementação do Firebase ML Kit

Para utilizar o Firebase ML Kit na aplicação, é necessário primeiramente adicionar as dependências das bibliotecas do ML Kit ao arquivo do projeto, assim, as funções de reconhecimento de caracteres estarão disponíveis para serem utilizadas na aplicação.

O primeiro passo para criar a função de reconhecimento de texto utilizando o ML kit na aplicação é criar um objeto `FirebaseVisionImage` a partir de um objeto `Bitmap`. Após essa etapa, é necessário criar uma instância do `FirebaseVisionTextRecognizer`, transmitir a imagem para o método `processImage` e por fim, extrair o texto dos blocos de texto reconhecidos. A função utilizada para reconhecimento de caracteres utilizando o ML Kit pode ser observada no Código-fonte 3, ela tem como parâmetros de entrada um arquivo bitmap e um objeto `EditText`. A função retornará uma string com os caracteres reconhecidos.

Código-fonte 3 – Função de reconhecimento de caracteres utilizando o ML kit.

```

1 private void reconhecer(Bitmap bitmap, EditText texto ){
2     if(mPreviewIv==null){
3         Toast.makeText(getApplicationContext(),"Nao existe Imagem"
4             ,Toast.LENGTH_LONG).show();
5     }else{
6         //criacao do objeto firebaseVisionImage
7         final FirebaseVisionImage firebaseVisionImage =
            FirebaseVisionImage.fromBitmap(bitmap);
8         //Chamando Instancia do FirebaseVisionTextRecognizer

```

```
8  FirebaseVisionTextRecognizer firebaseVisionTextRecognizer
   = FirebaseVision.getInstance().
     getOnDeviceTextRecognizer();
9  //Transmitindo a imagem para o metodo ProcessImage
10 firebaseVisionTextRecognizer.processImage(
    firebaseVisionImage).addOnSuccessListener(new
    OnSuccessListener<FirebaseVisionText>() {
11  @Override
12  public void onSuccess(FirebaseVisionText
    firebaseVisionText){
13  //Extrair texto dos blocos de texto
14  List<FirebaseVisionText.TextBlock> blocks =
    firebaseVisionText.getTextBlocks();
15  if(blocks.size()==0){
16  Toast.makeText(getApplicationContext(),"Nenhum texto
    detectado", Toast.LENGTH_LONG).show();
17  }else{
18  for(FirebaseVisionText.TextBlock block:
    firebaseVisionText.getTextBlocks()){
19  String text = block.getText();
20  texto.setText(text);
21  } } } })
22 .addOnFailureListener( new OnFailureListener() {
23  @Override
24  public void onFailure(@NonNull Exception e) {
25  }
26  });
27 }
28 }
```

3.4.1.2 Implementação do Google Cloud Vision

Para utilizar o Google Cloud Vision na aplicação é necessário adicionar as dependências de sua bibliotecas ao arquivo Gradle do projeto. Assim, as funções de reconhecimento de caracteres estarão disponíveis para serem utilizadas na aplicação. Para criar a função de reconhecimento de texto utilizando o Google Cloud Vision na aplicação, deve ser criado um objeto Frame que recebe a imagem a ser tratada. Em seguida, um objeto TextRecognizer é criado a partir do objeto Frame, onde serão retornados os blocos de texto reconhecidos. Logo após, é criada uma String com o texto dos blocos de texto reconhecidos.

A função utilizada para reconhecimento de caracteres utilizando o Cloud Vision pode ser observada no Código-fonte 4, ela tem como parâmetros de entrada um arquivo bitmap e um objeto EditText. A função retornará uma string com os caracteres reconhecidos.

Código-fonte 4 – Função de reconhecimento de caracteres utilizando o Cloud Vision.

```
1 private void reconhecer(Bitmap bitmap, EditText texto ){
2     TextRecognizer recognizer = new TextRecognizer.Builder(
3         getApplicationContext()).build();
4     if(!recognizer.isOperational()){
5         Toast.makeText(this, "Erro", Toast.LENGTH_SHORT).
6             show();
7     }else {
8         Frame frame = new Frame.Builder().setBitmap(bitmap
9             ).build();
10        SparseArray<TextBlock> items = recognizer.detect(
11            frame);
12        StringBuilder sb = new StringBuilder();
13        for (int i=0; i<items.size(); i++){
14            TextBlock myItem = items.valueAt(i);
15            sb.append(myItem.getValue());
16            sb.append("\n");
17        } texto.setText(sb.toString());
18    } }
```

3.4.2 Desenvolvimento da segunda parte da aplicação

Nesta parte do desenvolvimento, a aplicação foi preparada para reconhecer apenas determinados campos de uma imagem padronizada. Para realizar o reconhecimento de caracteres, a função recebe uma imagem e retorna uma string que contém todo o texto reconhecido, logo, pode ser realizado um tratamento no mesmo para obter apenas as informações desejadas.

As ferramentas de OCR utilizadas no desenvolvimento da aplicação fazem o reconhecimento de blocos de texto presentes em imagens e ao final do processo, cada bloco é transformado em uma linha da string como pode ser observado no texto reconhecido da Figura 12.

Figura 12 – Formulário utilizado e texto reconhecido



Fonte: Elaborada pelo Autor.

Na linguagem JAVA existe o método *split* que é capaz de dividir uma string em várias substrings a partir de um caractere definido na função, podendo este, ser uma quebra de linha. Utilizando tal método apenas as substrings com as linhas que contém informações de interesse podem ser acessadas e enviadas para o campo de texto correspondente na tela principal da aplicação.

O Código-fonte 5 exemplifica a utilização do método *split*, onde a string `textoCompleto` contém todo o texto reconhecido. Se este código for aplicado à string reconhecida na Figura 12, ao acessar a posição 0 da string `texto`, será retornado o texto "FORMULÁRIO DE CADASTRO", se acessar a posição 1 será retornado o texto "Nome:" e assim por diante, assim cada substring pode ser enviada a seu campo correspondente na tela principal, onde o texto pode ser editado caso haja alguma falha no reconhecimento dos caracteres.

Código-fonte 5 – Exemplo de utilização do método *split*.

```

2 String[] texto = textoCompleto.toString().split("\n");
3     campoTitulo.setText(texto[0]);
4     campoEndereco.setText(texto[3]);
5     campoTelefone.setText(texto[5]);
6     ...

```

3.4.2.1 Adaptação para imagens padronizadas que apresentam campos complexos

O formulário mostrado na Figura 12, se trata de um formulário simples onde os dados estão dispostos de forma paralela e com poucas informações, mas existem casos em que a imagem apresenta muita informações e campos de texto com rotação como é o caso do modelo de Carteira Nacional de Habilitação (CNH) vigente no Brasil que pode ser observado na Figura 13.

Figura 13 – Modelo de CNH

O diagrama mostra a estrutura de uma CNH com as seguintes seções e campos:

- Topo:** Logotipo do Brasil e o texto "REPÚBLICA FEDERATIVA DO BRASIL", "MINISTÉRIO DAS CIDADES", "DEPARTAMENTO NACIONAL DE TRÂNSITO", "CARTEIRA NACIONAL DE HABILITAÇÃO".
- Seção Pessoal:**
 - NOME: NOME COMPLETO DO CONDUTOR
 - DOC. IDENTIFICAÇÃO EMISSOR: 0000000000 XXX XX
 - DATA DE NASCIMENTO: 000.000.000-00 / 00/00/0000
 - RELACÃO: NOME COMPLETO DO PAI, NOME COMPLETO DA MÃE
 - PERMISSÃO: [campo], ACC: [campo], COT. NAL: [campo]
 - Nº REGISTRO: 000000000000, VALOR: 00/00/0000, 1ª HABILITAÇÃO: 00/00/0000
- Seção Observações:** OBSERVAÇÕES
- Seção Assinatura:** ASSINATURA DO PORTADOR, LOCAL: [campo], DATA EMISSÃO: [campo]
- Seção Assinatura Digital:** ASSINADO DIGITALMENTE, DEPARTAMENTO ESTADUAL DE TRÂNSITO
- Seção Estado:** CEARÁ
- Seção Denatran/Contran:** DENATRAN, CONTRAN

Dimensões indicadas: 8,5cm de largura e 12cm de altura. Retângulos vermelhos destacam o "Nº habilitação" e o estado "CEARÁ".

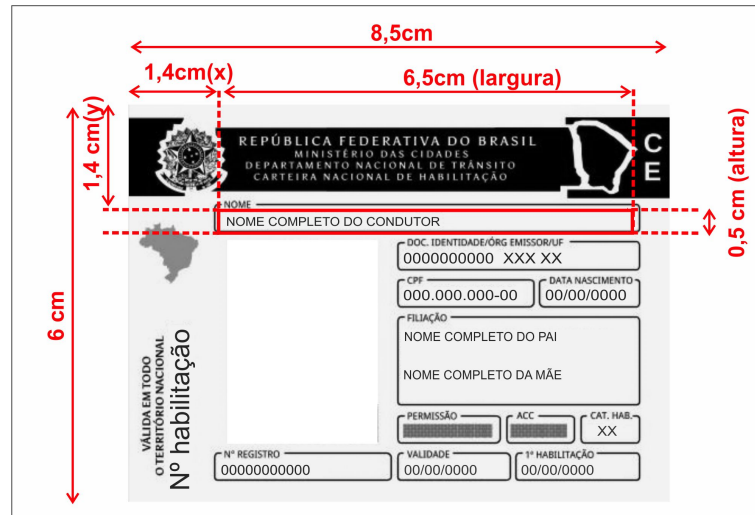
Fonte: Elaborada pelo Autor.

Neste modelo de CNH, existem campos com informações com rotação de 90 graus como é o caso do número da habilitação, além de existirem vários campos de texto espalhados e que não interessam. A solução para tal problema é selecionar apenas as partes da imagem que interessam como podem ser observados na Figura 13, onde os retângulos vermelhos são as áreas de interesse.

Para seleccionar as áreas de interesse primeiramente é necessário conhecer as dimensões da imagem em questão e ter conhecimento da:

- Distância entre o topo da imagem e o topo da área de interesse (y).
- Distância entre a lateral esquerda da imagem e a lateral esquerda da área de interesse (x).
- Altura e largura da área de interesse.

Figura 14 – Medidas da área de interesse.



Fonte: Elaborada pelo Autor.

Com essas medidas, é possível criar novas imagens a partir da imagem original, fazer o tratamento necessário nelas, como corrigir a rotação, por exemplo, e aplicar a função de reconhecimento de caracteres nas mesmas. Para isso existem métodos em JAVA para tratamento de imagem que permitem além de outras coisas, obter novos bitmaps a partir das medidas citadas anteriormente.

Um exemplo de uso de tal método pode ser observado no código-fonte 6, onde `imgOriginal` é o bitmap original, `novaImg` é o novo bitmap obtido a partir da coordenada (x,y), de largura `l` e altura `a`.

Código-fonte 6 – Exemplo de utilização do método `createBitmap`.

```

1  ...
2  Bitmap novaImg=Bitmap.createBitmap(imgOriginal,x,y,l,a);
3  ...

```

Para evitar mau funcionamento da aplicação é importante definir as dimensões exatas da imagem a ser tratada, o que também é importante para definir as áreas de interesse. Sabendo

que o formulário em questão (CNH) tem 8,5cm de largura por 12cm de altura, é necessário converter esses valores para a escala utilizada em JAVA. Utilizando uma razão igual a 100, as dimensões da imagem digital podem ser convertidas para 850dp de largura por 1200dp de altura.

Após o recebimento de uma imagem, a mesma será redimensionada para as dimensões 850dp x 1200dp para que possam ser obtidas as áreas de interesse com maior precisão. Para selecionar a área de interesse que contém o campo nome, por exemplo, será criado um novo bitmap conforme o Código-fonte 6 e a Figura 14, respeitando a razão do redimensionamento, com coordenadas $x=140dp$, $y=140dp$ e de largura $l=650dp$ e altura $a=50dp$.

Para os outros campos de interesse o processo será o mesmo. Após encontradas as áreas de interesse, será aplicada a função de OCR nas imagens e o texto reconhecido será enviado para seu respectivo campo na tela principal.

4 RESULTADOS

Para a realização dos testes nas ferramentas de OCR e na aplicação desenvolvida, foram utilizadas imagens contendo textos. Após o reconhecimento de caracteres ter sido realizado, foi aplicada a técnica de acurácia, que segundo (CPE TECNOLOGIA, 2019), é definido como exatidão de um valor obtido com relação a um valor tomado como referência e é dada pela equação 4.1.

$$\gamma_{acc} = \frac{\omega - \omega_{err}}{\omega} 100[\%] \quad (4.1)$$

Onde γ_{acc} é a acurácia, ω é a quantidade de símbolos que devem ser reconhecidos, ω_{err} é a quantidade de caracteres reconhecidos de forma incorreta.

4.1 Testes das ferramentas de OCR ML kit e Cloud Vision

Visando obter o desempenho das ferramentas Cloud Vision e ML kit no reconhecimento de caracteres, e assim comprovar qual se mostra mais eficiente, foram utilizadas três imagens digitais com características diferentes, todas contendo o mesmo texto composto por 159 caracteres. Cada caso e seus resultados estão descritos nos próximos tópicos.

4.1.1 Primeiro Teste

No primeiro teste realizado, foi utilizada uma imagem digital gerada a partir de captura de tela e recuperada do armazenamento interno do dispositivo. O texto contido em tal imagem apresenta a cor preta nos caracteres e o fundo branco. O Resultado do reconhecimento obtido pela aplicação android utilizando as ferramentas Cloud Vision e ML kit podem ser observados na Figura 15.

Figura 15 – Resultados obtidos no reconhecimento de caracteres do primeiro teste

Imagem Utilizada	Resultado Cloud Vision	Resultado ML Kit
OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.	OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.	OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.

Fonte: Elaborada pelo Autor.

A acurácia calculada em ambos os casos pode ser observada na Tabela 7. Analisando

o resultado, o Cloud Vision não apresentou nenhum erro no reconhecimento dos caracteres, já o ML Kit apresentou erro no reconhecimento de um caractere.

Tabela 7 – Acurácia obtida no primeiro teste

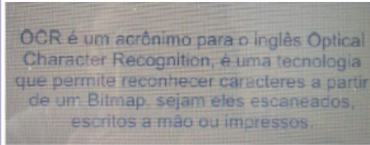
	Cloud Vision	ML kit
Quantidade de caracteres	159	159
Quantidade de erros	0	1
Acurácia Obtida	100%	99,37%

Fonte: elaborado pelo autor.

4.1.2 Segundo Teste

No segundo teste foi utilizada uma imagem de baixa qualidade obtida a partir da câmera de um dispositivo móvel. O Resultado do reconhecimento obtido pela aplicação Android utilizando as ferramentas Cloud Vision e ML kit podem ser observados na Figura 16.

Figura 16 – Resultados obtidos no reconhecimento de caracteres do segundo teste

Imagem Utilizada	Resultado Cloud Vision	Resultado ML Kit
	OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.	OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.

Fonte: Elaborada pelo Autor.

A acurácia calculada em ambos os casos pode ser observada na Tabela 8. Analisando o resultado, o Cloud Vision apresentou três erros no reconhecimento dos caracteres, sendo estes relacionados a acentuação, falta de caractere e falta de pontuação. Já o ML Kit, apresentou cinco erros, estes relacionados a acentuação, pontuação, falta e troca de caracteres.

Tabela 8 – Acurácia obtida no segundo teste

	Cloud Vision	ML kit
Quantidade de caracteres	159	159
Quantidade de erros	3	5
Acurácia Obtida	98,11%	96,85%

Fonte: elaborado pelo autor.

4.1.3 Terceiro Teste

No terceiro teste, foi utilizada uma imagem obtida a partir de captura de tela, esta com uma inclinação de dez graus em relação a linha da base do texto. O Resultado do reconhecimento obtido pela aplicação Android utilizando as ferramentas Cloud Vision e ML kit podem ser observados na Figura 17.

Figura 17 – Resultados obtidos no reconhecimento de caracteres do terceiro teste

Imagem Utilizada	Resultado Cloud Vision	Resultado ML Kit
OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir de um Bitmap, sejam eles escaneados, escritos a mão ou impressos.	OCR é um acrônimo para o inglês Optical Character Recognition, é uma tecnologia que permite reconhecer caracteres a partir <u>ap, sejam eles escaneados,</u>	ap, sejam eles escaneados,

Fonte: Elaborada pelo Autor.

A acurácia calculada em ambos os casos pode ser observada na Tabela 9. Analisando o resultado, o Cloud Vision apresentou oito vezes a falta de caracteres e uma sequência incorreta de 23 caracteres. Já o ML Kit reconheceu apenas 23 de um total de 159 caracteres.

Tabela 9 – Acurácia obtida no terceiro teste

	Cloud Vision	ML kit
Quantidade de caracteres	159	159
Quantidade de erros	8	137
Acurácia Obtida	94,97%	13,84%

Fonte: elaborado pelo autor.

4.2 Testes de reconhecimento de texto em imagens padronizadas que apresentam campos complexos

Para o reconhecimento de caracteres em campos de texto complexos, a aplicação foi preparada para reconhecer algumas áreas de interesse do modelo de CNH vigente, conforme as demarcações em vermelho na Figura 18, onde são selecionadas seis áreas de interesse sendo elas os campos de: foto, nome, CPF, categoria da habilitação, número da habilitação (campo com rotação) e Estado. Com excessão do campo de foto, os demais foram processados pela função de OCR.

Figura 18 – Modelo de CNH com campos de interesse para realização de testes.

REPUBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DAS CIDADES
DEPARTAMENTO NACIONAL DE TRÂNSITO
CARTEIRA NACIONAL DE HABILITAÇÃO

Nome: **ANTÔNIO ALVES MARTINS**

DOC. IDENTIDADE/ORG. EMISSOR: **0000000000 XXX XX**

CPF: **000.000.000-00**

DATA NASCIMENTO: **00/00/0000**

RELACÃO: **AB**

NOME COMPLETO DO PAI: **JOSE CARLOS MARTINS**

NOME COMPLETO DA MÃE: **ANA MARIA ALVES MARTINS**

PERMISSÃO: **AB**

Nº REGISTRO: **00000000000**

VALIDADE: **00/00/0000**

Nº HABILITAÇÃO: **00/00/0000**

OBSERVAÇÕES:

ASSINATURA DO PORTADOR: **CEARÁ**

LOCAL: **CEARÁ**

DATA EMISSÃO: **00/00/0000**

ASSINADO DIGITALMENTE
DEPARTAMENTO NACIONAL DE TRÂNSITO

Nº habilitação: **1233215696**

DENATRAN CONTRAN

Fonte: Elaborada pelo Autor.

4.2.1 Primeiro teste da Aplicação

Na realização do teste da aplicação, foi utilizada a imagem de um modelo de CNH com dados fictícios gerada a partir de um programa de edição de imagens, sendo esta salva no armazenamento do dispositivo e posteriormente recuperada a partir da função de obter imagem da galeria. O resultado obtido está presente na Figura 19, onde é mostrado em A a imagem da qual as informações foram retiradas, em B as áreas de interesse extraídas e em C os campos da aplicação com os caracteres reconhecidos a partir das áreas de interesse.

Figura 19 – Apresentação dos resultados obtidos a partir de imagem obtida da galeria.

OCRapp
clique em + para adicionar uma I...

Imagem Selecionada

REPUBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DAS CIDADES
DEPARTAMENTO NACIONAL DE TRÂNSITO
CARTEIRA NACIONAL DE HABILITAÇÃO

Nome: **ANTÔNIO ALVES MARTINS**

DOC. IDENTIDADE/ORG. EMISSOR: **1234321453 - SSP-CE**

CPF: **632.342.342-98**

DATA NASCIMENTO: **05/05/1990**

RELACÃO: **AB**

NOME COMPLETO DO PAI: **JOSE CARLOS MARTINS**

NOME COMPLETO DA MÃE: **ANA MARIA ALVES MARTINS**

PERMISSÃO: **AB**

Nº REGISTRO: **1233215696**

VALIDADE: **00/00/0000**

Nº HABILITAÇÃO: **00/00/0000**

OBSERVAÇÕES:

ASSINATURA DO PORTADOR: **CEARÁ**

LOCAL: **CEARÁ**

DATA EMISSÃO: **00/00/0000**

ASSINADO DIGITALMENTE
DEPARTAMENTO NACIONAL DE TRÂNSITO

Nº habilitação: **1233215696**

DENATRAN CONTRAN

OCRapp
clique em + para adicionar uma I...

Nome: **ANTÔNIO ALVES MARTINS**

CPF: **632.342.342-98**

Categoria: **AB**

Numero da Habilitação: **1233215696**

Estado: **CEARÁ**

OCRapp
clique em + para adicionar uma I...

Nome: **ANTONIO ALVES MARTINS**

Numero CNH: **1233215696**

Categoria: **AB**

CPF: **632.342.342-98**

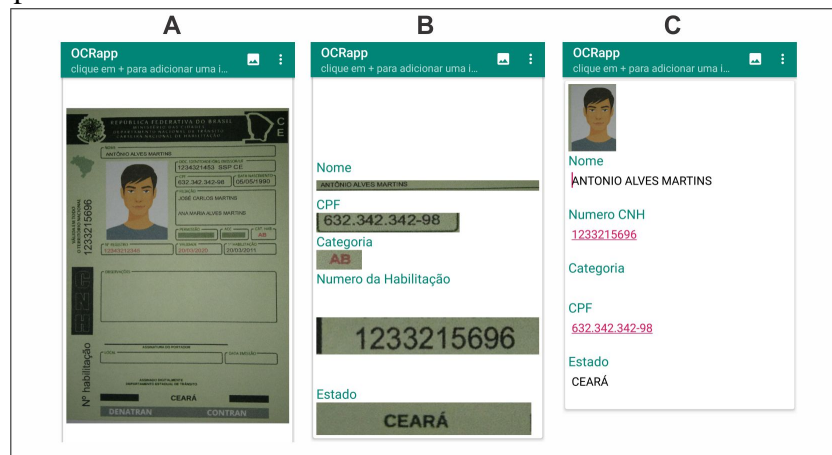
Estado: **CEARA**

Fonte: Elaborada pelo Autor.

4.2.2 Segundo teste da Aplicação

Outro teste foi realizado a fim de obter o desempenho da aplicação no reconhecimento OCR em imagens obtidas a partir da câmera do dispositivo, visto que estas apresentam grande quantidade de ruído, rotação e outros fatores que prejudicam o reconhecimento de caracteres. O resultado do processamento pode ser observado na Figura 20, onde é mostrado em A a imagem da qual as informações foram retiradas, em B as áreas de interesse extraídas e em C os campos da aplicação com os caracteres reconhecidos a partir das áreas de interesse.

Figura 20 – Apresentação dos resultados obtidos a partir de imagem obtida da câmera do dispositivo.



Fonte: Elaborada pelo Autor.

Para obter o desempenho da aplicação nos testes realizados foram calculadas as acurácias do primeiro e segundo testes, como pode ser observada na Tabela 10.

Tabela 10 – Acurácia obtida no primeiro e segundo testes de reconhecimento em um formulário complexo

	Primeiro Teste	Segundo Teste
Quantidade de caracteres	50	50
Quantidade de erros	2	3
Acurácia Obtida	96%	94%

Fonte: elaborado pelo autor.

4.2.3 Terceiro teste da Aplicação

A imagem de CNH fictícia utilizada nos testes anteriores apresenta cores diferentes, pouco ruído e menos detalhes que uma CNH real, visto isso, o terceiro teste da aplicação foi realizado com base em amostras de sete imagens de CNHs reais, onde foram extraídas as mesmas

áreas de interesse e realizado o reconhecimento OCR. Foram realizados cinco testes com cada amostra e a partir desses valores foram calculadas para cada caso a média de erros e seu desvio padrão. A Tabela 11 mostra o resultado obtido para cada amostra durante os cinco testes.

Tabela 11 – Quantidade de erros obtidos nos testes para cada amostra.

Amostra	teste 1	teste 2	teste 3	teste 4	teste 5	Média	Desvio Padrão
1	13	34	11	42	21	24,2	13,44
2	57	23	23	19	16	27,6	16,70
3	21	11	15	14	17	15,6	3,71
4	15	10	42	37	15	23,8	14,58
5	8	12	9	8	16	10,6	3,43
6	16	14	25	13	27	19	6,52
7	12	14	12	16	15	13,8	1,79

Fonte: elaborado pelo autor.

A partir do melhor resultado obtido para cada amostra nos testes realizados, foram calculadas as acurácias que estão dispostas na Tabela 12.

Tabela 12 – Acurácia obtida nos testes com CNHs reais com base no melhor resultado

Amostra	Quantidade de Caracteres	Quantidade de erros	Acurácia
1	55	11	80,00%
2	63	16	74,60%
3	46	11	76,08%
4	57	10	82,45%
5	43	8	81,39%
6	54	13	75,93%
7	46	12	73,91%

Fonte: elaborado pelo autor.

A partir da média obtida para cada amostra nos testes realizados, foram calculadas as acurácias que estão dispostas na Tabela 13.

Tabela 13 – Acurácia obtida nos testes com CNHs reais com base na média de erros

Amostra	Quantidade de Caracteres	Média de erros	Acurácia
1	55	24,20	56,00%
2	63	27,6	56,19%
3	46	15,6	66,08%
4	57	23,8	58,24%
5	43	10,60	75,35%
6	54	19,00	64,81%
7	46	13,80	70,00%

Fonte: elaborado pelo autor.

5 CONCLUSÕES

Retomando ao objetivo geral proposto, isto é, desenvolver uma aplicação Android que utiliza a tecnologia OCR para reconhecer campos de um formulário, é possível concluir que o mesmo foi alcançado com êxito, visto que como comprovado na seção anterior, foi possível utilizar a aplicação desenvolvida para encontrar as áreas de interesse presentes na CNH e aplicar a elas a função de OCR, obtendo assim o texto contido nas mesmas com uma acurácia satisfatória.

Ao analisar os resultados obtidos nos testes das ferramentas ML kit e Cloud Vision percebe-se que a segunda se mostrou mais eficiente que a primeira, visto que o reconhecimento de caracteres realizado pela mesma obteve uma acurácia maior, principalmente em casos em que o texto a ser reconhecido apresenta rotação. Ao ser utilizado no reconhecimento de campos de imagens complexas, o Cloud Vision se mostrou eficiente, reconhecendo até mesmo pequenos textos presentes em imagens com muito ruído, obtidas a partir da câmera do dispositivo.

Com base nos resultados obtidos dos testes da aplicação final, pode-se concluir que o reconhecimento OCR em campos de uma imagem padrão funciona muito bem para imagens com poucos ruídos, porém em imagens obtidas a partir da câmera do dispositivo que apresentam rotação e ruídos nota-se que existe uma queda na eficiência da aplicação, mas ainda assim, o reconhecimento é satisfatório.

Conclui-se que é possível utilizar a aplicação desenvolvida neste trabalho para diversos fins relacionados a reconhecimento de campos em imagens padronizadas, como é o caso de cartões de crédito, documentos de identificação, formulários, fichas, gabaritos, prontuários médicos e vários outros casos.

5.0.1 *Trabalhos Futuros*

Para uma versão futura deste trabalho, é proposto implementar a idéia de que o próprio usuário possa configurar a aplicação Android de acordo com as características da imagem padronizada da qual ele deseja efetuar o reconhecimento OCR, possibilitando assim criar as áreas de interesse, bem como seus respectivos campos correspondentes.

Além da implementação citada anteriormente, algo útil seria substituir a opção do usuário de selecionar as bordas da imagem em que deseja realizar o reconhecimento por algum algoritmo ou ferramenta que faça isso de forma inteligente.

Outra proposta é implementar um banco de dados, possibilitando que ao ser efetuado

o reconhecimento de texto, o mesmo seja enviado para seu determinado campo editável de texto, para que seja conferido pelo usuário e posteriormente salvo, podendo depois ser modificado ou excluído. Para completar a eficiência da aplicação, é interessante utilizar uma ferramenta de OCR capaz de reconhecer textos manuscritos, possibilitando assim a automatização de algumas tarefas, como por exemplo a digitação de fichas preenchidas manualmente para que depois sejam incluídas em determinado sistema.

REFERÊNCIAS

- ANDRADE, R. P. Desenvolvimento de um aplicativo movel com ocr e reconhecimento de voz para leitura de consumo de agua e gas em condominios. Andrade, 2016.
- ANDROIDPRO. **Visão geral das intents**. 2019. Disponível em: <<https://www.androidpro.com.br/blog/desenvolvimento-android/intents/>>. Acesso em: 20 mai. 2019.
- ARAUJO, J. V. F. Análise e classificação de imagens para aplicação de ocr em cupons fiscais. 2017. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/182212/TCC_JOSE_VICTOR_FEIJO%CC%81.pdf>.
- AZEVEDO, A. L. M. Caso de uso de ferramentas de ocr para automação da inserção de informações em banco de dados. 2018. Disponível em: <https://app.uff.br/riuff/bitstream/1/8923/1/TCC_ANA_LUCIA_MENDES_DA%20_CUNHA_AZEVEDO_E_ANA_LUISA_PEREIRA_RIEDEL_DE_CARVALHO.pdf>.
- CHERIET, M. **Character Recognition Systems**. New Jersey: wiley, 2007. v. 1.
- CONCRETE. **Use machine lernaing para resolver problemas do mundo real**. 2018. Disponível em: <<http://micreiros.com/firebase-o-que-e-e-como-funciona/>>. Acesso em: 12 mai. 2019.
- CPE TECNOLOGIA. **Precisão e acurácia**. 2019. Disponível em: <<https://blog.cpetecnologia.com.br/precisao-e-acuracia-voce-sabe-a-diferenca/>>. Acesso em: 20 mai. 2019.
- EIKVIL, L. **Optical Character Recognition**. [S.l.: s.n.], 1993. v. 1.
- FAHLEN, E. Androidapplikation för digitalisering av formulär – minimering av inlärningstid, kostnad och felsannolikhet. 2019. Disponível em: <<http://www.diva-portal.org/smash/get/diva2:1288189/FULLTEXT01.pdf>>.
- FIREBASE. **Reconhecimento de texto (OCR)**. 2019. Disponível em: <<https://firebase.google.com/docs/ml-kit/recognize-text?hl=pt-br>>. Acesso em: 15 mai. 2019.
- G1. **Em SP, radar começa a multar carro sem licenciamento**. 2010. Disponível em: <<http://g1.globo.com/brasil/noticia/2010/11/em-sp-radar-comeca-a-multar-carro-sem-licenciamento.html>>. Acesso em: 10 nov. 2018.
- GONZALEZ, R. C.; WOODS, R. E. **Processamento digital de imagens**. São Paulo: Pearson Prentice Hall, 2010. v. 3.
- GOOGLE CLOUD. **Análise avançada de imagens**. 2019. Disponível em: <<https://cloud.google.com/vision/?hl=pt-br>>. Acesso em: 13 mai. 2019.
- HERNANDEZ, A. J. Diseño y desarrolo de una aplicación android para el reconocimiento óptico de caracteres. 2018. Disponível em: <<http://uvadoc.uva.es/bitstream/10324/32987/1/TFG-G3410.pdf>>.
- KNOOW. **Aplicacoes de OCR**. 2016. Disponível em: <<http://knoow.net/ciencinformtelec/informatica/ocr-optical-character-recognition/>>. Acesso em: 02 nov. 2018.
- LEGAL SCANS. **Optical Charactere Recognition**. 2018. Disponível em: <<http://www.legalscans.com/ocr.html>>. Acesso em: 01 nov. 2018.

LINS, L. F. Reconhecimento ótico de caracteres e análise de sistemas ocr baseados em código aberto. Lins, 2012.

MEDEIROS, J. D. Ferramenta de detecção da região da placa de veículos. 2009.

MICREIROS. **Firestore: como funciona**. 2016. Disponível em: <<http://micreiros.com/firebase-o-que-e-e-como-funciona/>>. Acesso em: 12 mai. 2019.

MITH, R. Optical character recognition: International journal of recent technology and engineering. IJRTE, 2013.

NEOMIND. **Benefícios do OCR**. 2018. Disponível em: <<http://www.neomind.com.br:81/blog/ocr-estruturacao-de-dados/>>. Acesso em: 01 nov. 2018.