Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. XX, Num. XX (2018) 11-XX

RESEARCH ARTICLE

Prototype for word translation using techniques for optical character recognition (OCR)

Protótipo para tradução de palavras utilizando técnicas para reconhecimento ótico de caracteres (OCR)

João Gabriel Ferro Beani[1]*, Gustavo Poli[2]

Resumo: The learning of foreign languages is a little raised by the public schools in Brazil always with a lower than expected teaching, many students finish teaching without a good knowledge base in new languages. Several researches with the use of computational approaches have been directed to assist in the teaching of foreign languages. This project aims to prove the feasibility of collaborative computer vision applications using different APIs, thus performing the extraction of characters from words in the English language through optical character recognition and soon after that the same will be translated into Portuguese. A desktop platform application with the Python programming language was developed in conjunction with the OpenCV and Tesseract OCR libraries linked to the Google Translate API, the development environment used was the Atom because of its intimacy with the chosen language and the fact that it is a free tool. Where a picture of the desired word is taken, it is processed inside the prototype by the character extraction method and after that is used of the Google Translation REST API and the value returned is the Portuguese word. Efficiency is achieved when the character extraction method through the Tesseract OCR library removes all characters without any mismatching of the form of the word. This article proposes the construction of a prototype for word translation using optical character recognition techniques. He achieved an effective performance in the objective of character extraction and translation of the characters. In future works the goal will be to have the possibility of translation into other languages, not only the English language, and also the possibility that the user can hear the translation of the text taken from the image.
Keywords: Image Processing — Python — OCR — Education

Resumo: O aprendizado de línguas estrangeiras é um tema pouco levantado pelas escolas públicas no Brasil sempre com um ensino abaixo do esperado, muitos alunos terminam o ensino sem uma boa base de conhecimento em novas línguas. Diversas pesquisas com a utilização de abordagens computacionais têm se encaminhado para auxiliar no ensino de línguas estrangeiras. Este projeto apresenta como objetivo provar a viabilidade de aplicações de visão computacional colaborativas utilizando diferentes APIs, realizando assim a extração de caracteres vindos de palavras no idioma inglês por meio de reconhecimento ótico de caracteres e logo após isso as mesmas serão traduzidas para o idioma português. Foi desenvolvido uma aplicação para plataforma Desktop com a linguagem de programação Python em conjunto das bibliotecas OpenCV e Tesseract OCR atreladas a API do Google Tradutor, o ambiente de desenvolvimento utilizado foi o Atom devido à sua intimidade com a linguagem escolhida e o fato de ser uma ferramenta gratuita. Onde é retirada uma foto da palavra desejada, processa-se a mesma dentro do protótipo pelo método de extração de caracteres e após isso se utiliza da API REST de tradução do Google e o valor retornado é a palavra em português. Obtêm-se uma eficácia quando o método de extração de caracteres por meio da biblioteca Tesseract OCR retira todos os caracteres sem nenhuma descaracterização da forma do vocábulo. Este artigo propôs a construção de um protótipo para tradução de palavras utilizando de técnicas de reconhecimento ótico de caracteres. Ele obteve uma performance eficaz no objetivo de extração de caracteres e tradução dos caracteres. Em trabalhos futuros à meta será a aplicação ter possibilidade de tradução para outras línguas, não somente a língua inglesa, e também a possibilidade que o usuário possa ouvir a tradução do texto retirado da imagem.
Palavras-Chave: Processamento de Imagem — Python — OCR — Educação

[1] *Ciência da Computação, Centro Universitário da Fundação Educacional Guaxupé - UNIFEG, Brasil*
[2] *Ciência da Computação, Centro Universitário da Fundação Educacional Guaxupé - UNIFEG, Brasil*
***Corresponding author**: joaogg@unifeg.edu.br
DOI: http://dx.doi.org/10.22456/2175-2745.XXXX • **Received:** dd/mm/yyyy • **Accepted:** dd/mm/yyyy

1. Introdução

Ao decorrer das últimas décadas as áreas que envolvem o processamento digital de imagem vêm apresentando um crescente desenvolvimento. Esse crescimento tem explicação, atualmente o tema "processamento de imagens" é instrumento de pesquisas nas universidades de renome mundialmente, e não podemos esquecer também do meio industrial onde a todo o momento novas soluções envolvendo reconhecimento facial e reconhecimento ótico de caracteres são desenvolvidas para facilitar, melhorar e proporcionar maior segurança à população ao redor do globo.

Diversas pesquisas utilizando abordagens computacionais têm se encaminhado para auxiliar no ensino de línguas estrangeiras. Esses estudos tem como objetivo ajudar professores na identificação das dificuldades de seus alunos, podendo auxiliar no aprendizado de uma nova língua, desde que utilizando a tecnologia como principal forma de amparo.

Em um estudo feito pela rede estatística europeia Eurydice sobre o aprendizado de uma língua estrangeira, foi constatado que metade dos alunos europeus aprendem uma língua estrangeira já no 1º ciclo de ensino básico. Isso é apenas reflexo de mudanças feitas desde 1998 em vários países da Europa.[15] No Brasil a realidade é bem diferente, a obra Dimensões Comunicativas no Ensino de Línguas do autor José Carlos Paes de Almeida Filho mostra que, no Brasil, muitos estudantes da rede pública chegam ao 6º ano do Ensino Fundamental sem nenhuma experiência com uma língua estrangeira.[20]

No campo tecnológico, foram desenvolvidos diversos sistemas para auxiliar no ensino de uma língua estrangeira. Atualmente muitos aplicativos abordam o aprendizado de uma nova língua de diferentes métodos com seus usuários, seja por meio de jogos e escolas virtuais à até mesmo simples práticas de tradução em tempo real. Dentre eles podemos citar os dois mais utilizados nos dias de hoje: Duolingo, aplicativo com ensino de línguas através de interações entre seus usuários, e o Google Tradutor, aplicativo no qual a função é traduzir palavras para diversos idiomas.

Existem vários algoritmos para o processamento e análise de imagens visando a extração de caracteres por reconhecimento ótico. No entanto, diferentes grupos de desenvolvedores tendem a utilizar abordagens particularizadas para a obtenção de um melhor desempenho em suas aplicações. Este projeto apresenta como objetivo provar a viabilidade de aplicações de visão computacional colaborativas utilizando diferentes APIs, realizando assim a extração de caracteres vindos de palavras no idioma inglês por meio de reconhecimento ótico de caracteres e logo após isso as mesmas serão traduzidas para o idioma português. Com o objetivo de apresentar o novo método foram desenvolvidas aplicações desktop e mobile, utilizando de tecnologias como Tesseract OCR (*Tesseract Optical Character Recognition*) e OpenCV (*Open Source Computer Vision Library*).

2. Processamento de Imagem

A área de processamento digital de imagens atualmente é um setor em evidência mundialmente, com sua vasta importância em áreas bastante conhecidas pela sociedade ele agrega uma maior precisão e qualidade aos sistemas já existentes. Na medicina, na segurança e na educação o uso da visão computacional vem crescendo promissoramente, temos muitos avanços nas soluções que envolvem as àreas citadas utilizando de detecção facial, biometria, detecção de movimentos e reconhecimento ótico de caracteres (OCR).

Segundo o autor Herman Martins Gomes, o processamento digital de imagens (PDI) não é uma tarefa simples, na realidade envolve um conjunto de tarefas interconectadas. Iniciando-se sempre com a captura de uma imagem, a qual, normalmente, corresponde à iluminação que é refletida na superfície dos objetos, realizada através e um sistema de aquisição. Após a captura por um processo de digitalização, uma imagem precisa ser representada de forma apropriada para seu tratamento computacional. [28] O processamento de uma imagem geralmente é divido em etapas: aquisição, pré-processamento, segmentação, extração de características e interpretação.

Figura 1. Etapas no processamento de uma imagem.
FONTE: O autor (2018)

Segundo o autor Richard E. Woods e Rafael C. Gonzalez o processamento de imagem digital tem por finalidade tratar os dados de uma imagem de modo que seja possível da mesma ser utilizada para a análise computacional e humana. Assim, para alcançar este propósito, algumas etapas são usu-

almente executadas, dentre elas as principais são: aquisição de imagens, segmentação e a representação. [21]

- Aquisição de imagens
- Segmentação de imagens
- Representação de imagens

2.1 Aquisição de imagens

A etapa de aquisição de imagens, a imagem é capturada e convertida para o formato digital. Esta captura, usualmente utiliza de um sensor representado por uma câmera ou um digitalizador.[22] No artigo o processo de aquisição das imagens virá através da câmera de um disposível móvel ou uma webcam, a qual será retirada pelo usuário. Em muitos casos na área do processamento de imagens, a aquisição da imagem para extração de características é sempre a etapa mais díficil, diferente do que ocorre no projeto deste artigo.

2.2 Segmentação de imagens

A etapa de segmentação de imagens é onde diante da imagem digitalizada, produzida na etapa de aquisição. Tem-se o processo de análise de pixels em sua forma isolada, onde ocorrem transformações nesta imagem para alterar suas características. [22]

Abaixo pode-se visualizar um exemplo simples onde temos a respresentação das etapas de aquisição e segmentação respequitivamente.

Figura 2. Etapa para aquisição de imagens. FONTE: OpenCV-Python Tutorials (2018)

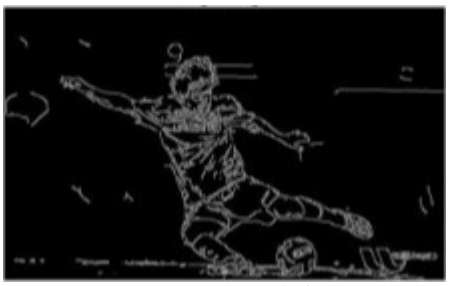

Figura 3. Etapa para segmentação de imagens. FONTE: OpenCV-Python Tutorials (2018)

2.3 Representação de imagens

A etapa de representação de imagens consiste em representar e descrever os pixels agrupados para extração de características obtidos na etapa de segmentação, seu resultado é a extração de informação numérica dos objetos de interesse,armazenando-a em uma estrutura de dados denominada de vetor de características. [22]

Para finalizar o autor Rafael C. Gonzalez enfatiza que o processamento digital de imagens vem em uma evolução contínua ao longo dos anos, com um aumento significativo de estudos envolvendo morfologia matemática, redes neurais, processamento de imagens coloridas, compressão de imagens, reconhecimento de imagens e sistemas de análise de imagens baseados em conhecimento.[21]

3. OCR: extração de caracteres de imagem

Antes do início aos fundamentos teóricos da extração de caracteres de uma imagem, o artigo vai apresentar uma pequena introdução sobre a criação e evolução das primeiras formas de escrita.

Segundo o autor Marcos Emílio Ekman Faber, editor do site História Livre, não sabe-se certamente onde teve-se início as primeiras formas de escrita. Estudos arqueológicos apontam registros que ocorreram na mesma época no Egito antigo e na Mesopotâmia, por volta do ano 3500 a.C.[6]

No Egito a escrita começou a se desenvolver na forma de ideogramas e ficou conhecida como escrita hieroglífica. Os hieróglifos pareciam obras de arte, neles eram registradas informações sobre a produção de alimentos, histórias do cotidiano egipicio e histórias dos faraós.[6]

Na Mesopotâmia a escrita de desenvolveu na forma de cunhas, também conhecidas como ferramentas de metal ou madeira em forma de um prisma agudo.[26] De início a escrita cuneiforme tinha por objetivo registrar as transações comerciais e a produção de alimentos, após muito tempo a escrita começou a ser utilizada para registrar fatos históricos do reino e seus monarcas. [6]

O primeiro alfabeto surgiu com o povo fenício por volta do ano 1 mil a.C. Após os fenícios outros povos começaram a adaptar e aperfeiçoar o alfabeto à suas necessidades. Os gregos foram os responsáveis por criar o primeiro alfabeto universal, ele começou a ser utilizado por vários povos, especialmente em transações comerciais internacionais. Do grego, os romanos criaram o alfabeto latino, que utilizamos atualmente.[6]

Segundo o autor Eikvil, as origens do reconhecimento de caracteres podem ser encontradas em 1870. Este foi o ano em que C.R.Carey, de Boston, Massachusetts, inventou o scanner de retina que era um sistema de transmissão de imagens usando um mosaico de fotocélulas. Duas décadas depois, o polonês P. Nipkow inventou o scanner sequencial, que foi um grande avanço tanto para a televisão moderna quanto para as máquinas de leitura. Durante as primeiras décadas do século 19, várias tentativas foram feitas para desenvolver dispositivos para ajudar os cegos através de experimentos com OCR. No entanto, a versão moderna do OCR não apareceu até meados

da década de 1940 com o desenvolvimento do computador digital. A motivação para o desenvolvimento, a partir de então, foram as possíveis aplicações dentro do mundo dos negócios. [19]

No ano de 1953, a IBM (*International Business Machines*) adquiriu uma licença da IMR e iniciou o desenvolvimento de um software próprio que foi chamado de Optical Character Recognition, o termo OCR tornou-se padrão para a indústria nesta tecnologia. Atualmente a área de processamento de imagens vem sendo objeto de crescente interesse por permitir viabilizar um grande número de aplicações em duas categorias bem distintas: o aprimoramento de informações gráficas para interpretação humana e a análise automática de informações extraídas de uma imagem por computadores. [2]

Nas figuras 1 e 2 exibi-se um exemplo simples de uma forma de extração de caracteres através do método OCR.

Figura 4. Texto para extração. FONTE: O autor (2018)

```
['Asia shares, euro pressured by\n
Turkish crisis']
```

Figura 5. Texto extraído da imagem. FONTE: O autor (2018)

Segundo o autor Daisuke Yamakawa, é sempre esperado que as técnicas de reconhecimento ótico de caracteres - (*Optical Character Recognition*) tenham a capacidade de traduzir imagens escritas a mão, datilografadas ou impressas. O alvo principal do OCR é em documentos no qual caracteres em cor preta estão em fundo branco com uma fonte unida. [24]

Com a grande melhoria dos sistemas de hardware com relação a processamento e custo, bem como o aumento de pesquisas voltadas para o reconhecimento de caracteres novas bibliotecas de OCR foram desenvolvidos. Podemos citar como as mais utilizados atualmente:

- PyTesseract
- Tess-Two
- JavaOCR
- PyOCR
- Tess4J

3.1 PyTesseract

O *Python-tesseract* é uma das bibliotecas padrões da linguagem de programação Python para o uso da biblioteca *Tesseract OCR* da Google.[3]

O *Python-tesseract* é uma ferramenta de reconhecimento óptico de caracteres (OCR), ele é uma ramificação para o conjunto de ferramentas para a utilização do mecanismo *Tesseract OCR* do *Google*. Ele pode ler todos os tipos de imagem suportados pela *Python Imaging Library*, incluindo jpeg, png, gif, bmp e outros. Uma das grandes vantagens desta biblioteca é o fato de *Python-tesseract* imprimir o texto reconhecido ao em vez de gravá-lo em um arquivo.[3]

3.2 Tess-Two

O *Tess-two* é uma das bibliotecas padrões da linguagem de programação *Java*, utilizada para programação de aplicações que utilizam o sistema android. Ele faz o uso da biblioteca *Tesseract OCR* da *Google*.[12]

Ele é uma ramificação do *Tesseract Tools* para *Android* (*tesseract-android-tools*) que possui algumas funcionalidades adicionais. O Tesseract Tools para Android é um conjunto de *APIs* do *Android* que cria arquivos para as bibliotecas de processamento de imagem OCR e Leptonica da Tesseract.[12]

Esta ramificação da biblioteca *Tesseract* trabalha com:

- Tesseract 3.05
- Leptonica 1.74.1
- libjpeg 9b
- libpng 1.6.25

O módulo *tess-two* possui ferramentas para compilar as bibliotecas *Tesseract* e *Leptonica* para uso na plataforma *Android*. Ele fornece uma *API Java* para acessar as *APIs Tesseract* e *Leptonica* para serem compiladas de forma nativa.[12]

3.3 JavaOCR

O *JavaOCR* é um conjunto de bibliotecas Java para processamento de imagem voltado para o reconhecimento ótico de caracteres (OCR). Ele fornece uma estrutura modular para facilitar sua implantação. Esta biblioteca possui uma grande falta de depências para o desenvolvidos de aplicações voltadas para a plataforma Android. [4]

3.4 PyOCR

O *PyOCR* é um conjunto de ferramentas para reconhecimento ótico de caracteres (OCR) formado para a linguagem de programação python. Ele nos ajuda a utilizar de várias ferramentas de OCR para um programa em Python. Para a extração de caracteres de uma imagem ele utiliza da biblioteca *Tesseract OCR*, obtendo saídas somente em texto. [5]

Essa biblioteca foi testada em sistemas GNU / Linux. Podendo não funcionar no *Windows, MacOSX*, etc. Suporta todos os formatos de imagem suportados pelo *Pillow* (biblioteca da linguagem de programação *Python* que adiciona suporte à abertura e gravação de muitos formatos de imagem diferentes) é uma, incluindo jpeg, png, gif, bmp e outros. [5]

3.5 Tess4J

O *Tess4J* é um conjunto de bibliotecas Java para a utilização da biblioteca Tesseract OCR voltada para a linguagem de programação Java, ele é distribuído sob a Licença Apache, v2.0 e também está disponível no Repositório Central do Maven. Esta biblioteca não é muito utilizado atualmente devido ao seu suporte para poucos formatos de imagem: TIFF, JPEG, GIF, PNG e BMP.[7]

4. Bibliotecas

Antes do início a esta parte do artigo, deve-se exibir um conceito básico: as bibliotecas diferentemente das *APIs* são coleções de classes e métodos soltos, que podemos usar para qualquer fim, de forma mais limpa podemos dizer que elas são como *plugins* que você adiciona a um projetos.[16]

4.1 Numpy

A biblioteca *NumPy* é um pacote básico da linguagem de programação Python que possibilita o trabalho em arranjos, vetores e matrizes com N dimensões. Ela possui uma sintaxe semelhante ao software Matlab, mas tem uma eficiência muito maior. A *NumPy* nos provê diversas funções e operações sofisticadas como: ferramentas para álgebra linear, ferramentas sofisticadas para geração de números aleatórios, transformadas de Fourier básicas e objetos diferentes para o cálculo de matrizes. [8]

A biblioteca *NumPy* no processamento de imagens trata da manipulação e processamento básico usando módulos científicos, ela oferece funções operando matrizes de N dimensões facilitando tarefas comuns no processamento de imagens como a extração de recursos, a segmentação de imagens e a filtragem de imagens.[8]

4.2 OpenCV

A biblioteca *OpenCV* (*Open Source Computer Vision Library*) é gratuita para uso acadêmico e comercial. Ela possui interfaces nas linguagens de programação: *C++, Python e Java*. Ela pode ser utilizada nas plataformas: Windows, Linux, Mac OS, iOS e Android. O OpenCV está dividido em cinco grupos de funções: processamento de imagens, análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões, calibragem de câmera e reconstrução 3D. Ela é utilizada em todo o mundo em suas diversas áreas de atuação e tem mais de 47 mil pessoas da comunidade de usuários e um número estimado de downloads que ultrapassam 14 milhões. [9]

Dentro da principal área tratada neste trabalho, no processamento de imagens a biblioteca *OpenCV* faz a implementação de ferramentas de interpretação de imagens, indo desde operações simples como um filtro de ruído, até operações complexas, tais como a análise de movimentos, reconhecimento de padrões e reconstrução em 3D. [9]

4.3 GoogleTrans

A biblioteca *Googletrans* é um pacote básico da linguagem de programação Python que foi implementada a partir da API do *Google* Tradutor. Ela utiliza a API do Google Tradutor para fazer chamadas em Ajax para métodos de detecção e tradução de palavras. [10]

Ela possui algumas características que possibilitam vantagens diantes de outros meios de tradução, a *Googletrans* é rápida e confiável devivo ao fato de utilizar os mesmos servidores que o translate.google.com, ela também detém de uma URL personalizável podendo utilizar de traduções em massa com detecções de idioma. [10]

A biblioteca envia às requisições de tradução através de do protocolo HTTP (Protocolo de Transferência de Dados) que é um protocolo ou padrão de rede implementado em cima do TCP para que browsers e servidores possam se comunicar. O nosso navegador, Chrome, Firefox, Safari é um cliente HTTP que se comunica com servidores e seus principais métodos de comunicação são: GET e POST. [10]

Quando é enviado a palavra sem tradução para receber a mesma traduzida, é utilizado do protocolo HTTP com o método GET, requisitando uma representação do recurso especificado, desta forma a respresentação da tradução é retornada por meio de JSON.

Código simples em JSON:

”code”:200,”lang”:”en-pt”,”text”:[”Olá”]

4.4 Tesseract OCR

O *Tesseract-OCR* é uma ferramenta livre de reconhecimento ótico de caracteres (OCR), ele é usado para detecção de texto em dispositivos móveis, em vídeo e na detecção de spam de imagens do *Gmail*. [11]

O *Tesseract-OCR* foi provavelmente a primeira biblioteca de OCR para lidar com textos em cor preta e fundo brando. Nesta fase, os contornos são reunidos, puramente poraninhamento, em blobs.[27]

Os *blobs* são regiões pequenas e isoladas da imagem digitalizadas que estão delineadas pelo seu contorno. O *Tesseract* às manipula para visualizar se alguma melhoria no OCR foi efetuada. Em muitos casos essas combinações melhoram o resultado final.[23]

Em meio ao processo de extração de caracteres, os *blobs* são organizados em linhas de texto, onde essas regiões são analisadas para o passo fixo ou texto proporcional. Linhas de texto são divididas em palavras diferentemente de acordo com o tipo de espaçamento. Texto de tom fixo é cortado imediatamente por células de caráteres. Texto proporcional é dividido em palavras usando espaços definidos.[27]

Abaixo a figura exibe todo o processo de extração de caracteres da biblioteca Tesseract-OCR.

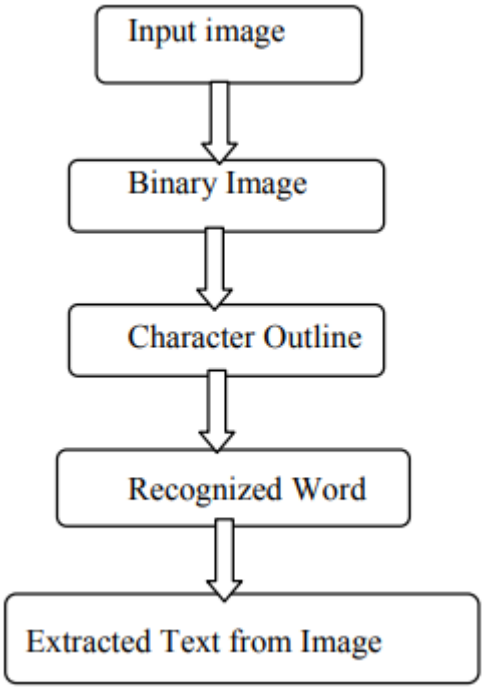

Figura 6. Arquitetura da biblioteca Tesseract OCR. FONTE: A Different Image Content-based Retrievals using OCR Techniques, Miss- Poonam A. Wankhede e Dr. Sudhir W. Mohod (2017)

5. Web Services

Um *Web service* é uma aplicação composta por um conjunto de métodos que executam suas instruções através de parâmetros que são enviados, essa aplicação pode ser utilizada por outros aplicativos, sites e programas através da tecnologia Web.[13]

O *Web service* também pode ser utilizado para fazer tranferências de dados através de protocolos de comunicação para diferentes plataformas. Um fato importante é que estes dados podem ser utilizados em outras aplicações independentemente das linguagens de programação utilizadas serem diferentes das do Web service.[13]

Um *Web service* é muito útil aos desenvolvedores devido ao fato do reaproveitamento de sistemas já existentes, podendo apenas acrescentar novas funcionalidades e informação facilitando a codificação de forma simples e rápida.[13]

Segundo o autor Sehrish Malik, que fez um comparativo entre os protocolos *REST* e *SOAP*, os serviços *RESTful* estão se tornando amplamente populares com a passagem do tempo e estão substituindo *SOAP-based* serviços *web* em muitas áreas devido à facilidade de uso e implantação. Assim, surge uma questão vital de interesse quanto a melhor desempenho entre serviços baseados em *RESTful* e *SOAP*. *REST* é um estilo arquitetônico que usa *URL* para expor a logíca de negócios. *REST* geralmente tem muitos recursos e é uma arquitetura orientada a recursos, enquanto o *SOAP* é um protocolo que usa interfaces de serviços para expor a lógica de negócios. A largura de banda e os requisitos de recursos do *REST* são menores comparação com o *SOAP*, e é por isso que o *REST* é considerado leve. *SOAP* é tudo sobre serviços onde, como no *REST*, existem muitos recursos, mas poucos métodos fixos usuários / programadores sem confusões. As diretrizes REST são chamadas de aplicativos *Restful*. Fornecendo mais flexibilidade e menos sobrecarga, resultando em uma melhor solução para a maioria das implementações.[25]

Abaixo a figura 7 exibe uma comparação dos protocolos REST e SOAP.

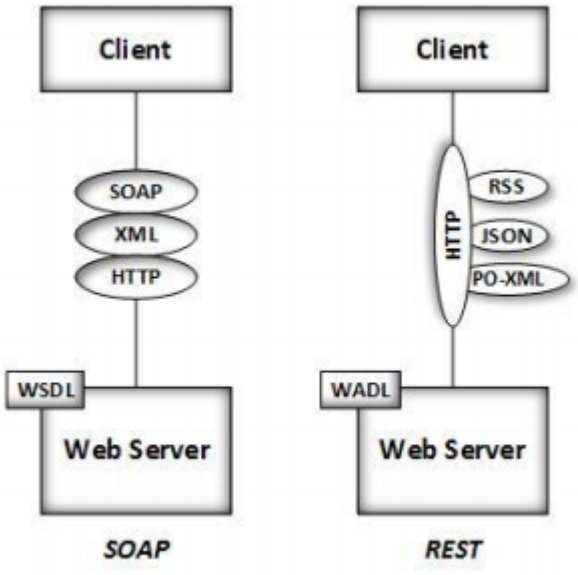

Figura 7. Diagrama Conceitual. FONTE: A Comparison of RESTful vs. SOAP Web Services in Actuator Networks, Sehrish Malik e Do-Hyeun Kim (2017)

5.0.1 Microsoft Translator Text API

O *Microsoft Translator Text API*, parte do Microsoft Cognitive Services , é um serviço de tradução automática baseado em nuvem que suporta mais de 60 idiomas. O tradutor pode ser usado para criar aplicativos, sites, ferramentas ou qualquer solução que exija suporte em vários idiomas. [14]

O *Microsoft Translator* é um serviço do Azure usado globalmente por milhares de organizações como parte de seus fluxos de trabalho existentes, tanto para conteúdo e aplicativos voltados para o lado interno quanto para o externo. eBay e Etsy estão entre as muitas empresas cujos aplicativos e serviços voltados ao consumidor oferecem traduções usando o Microsoft Translator e um grande número de clientes corporativos multinacionais, como Hewlett-Packard , Adobe e, claro, várias equipes da Microsoft em engenharia, marketing, finanças e vendas também usam o Microsoft Translator. [14]

5.0.2 Translation API

A *Translation API* oferece uma interface programática simples com a função de traduzir strings arbitrárias para qualquer idioma compatível por meio da tecnologia de tradução automática neural de ponta. A *Translation API* é altamente

responsiva. Assim, ao integrar sites e aplicativos a ela, você conseguirá traduções rápidas e dinâmicas de textos de um idioma de origem para um idioma de chegada (por exemplo, do francês para o inglês). A *API* também oferece detecção de idiomas para casos em que o idioma de origem é desconhecido. A tecnologia subjacente expande os limites da tradução automática. Além disso, ela é constantemente atualizada com novos idiomas e pares de idiomas para fornecer traduções cada vez melhores. [17]

5.0.3 Yandex Translate API

A *Yandex* é um serviço de buscas semelhante ao *Google*, que atua na Rússia, Ucrânia, Cazaquistão, Bielorússia e Turquia. A *API Yandex Translate* é uma ferramenta de tradução de texto universal que usa a tecnologia de tradução automática desenvolvida no *Yandex*. Ele permite que os desenvolvedores integrem a tradução automática em seus aplicativos, serviços e sites. [18]

Desenvolveu-se uma aplicação nomeada de ”A”para a plataforma *desktop* utilizando a biblioteca *Googletrans* que fará a comunicação com a *Translation API*. Essa biblioteca em combinação com sua *API* padrão na linguagem *Python* não possui nenhum tipo de cobrança pelos seus serviços, diferente da *Microsoft Translator Text API*. A biblioteca *Googletrans* tem como principal método de troca de informações um método *GET* por *JSON* que facilita todo o processo de atividade da aplicação ”A”.

O segundo desenvolvimento partiu-se de uma aplicação nomeada de ”B”para a plataforma mobile utilizando da *Yandex Translate API* que fará todo o processo de comunição entre os dados recebidos pelo aplicativo. Essa *API* voltada para o android é muito eficaz, as *APIs Microsoft Translator Text API* e *Translation API* tem cobrança por seus serviços, diferentemente da *Yandex* que oferece sua *API* de forma totalmente livre ao desenvolvedor. Ela também tem como principal método de troca de informações um método *GET* por *JSON* que facilita todo o processo de atividade da aplicação ”B”.

6. Projeto e Metodologia

Desenvolveu-se dois projetos utilizando diferentes linguagens de programação, bibliotecas, *API's* e plataformas:

- POC (Prova de conceitos)
- Linguagem: Python
- Biblioteca OCR: PyTesseract
- API de Tradução: Translation API
- Plataforma: Desktop

- Projeto comercial
- Linguagem: Java
- Biblioteca OCR: Tess-Two
- API de Tradução: Yandex API
- Plataforma: Mobile

O método proposto para criação dos projetos segue o fluxograma da figura 8. Esta metodologia parte de uma imagem com palavras ou frases escritas na língua inglesa que através de uma biblioteca OCR extrai seu texto em inglês e os envia para uma *API REST*, retornando por meio de *JSON* o texto traduzido para o português.

6.1 Prova de conceitos

A prova de conceitos foi desenvolvida para a plataforma *Desktop* com a linguagem de programação *Python* em conjunto das bibliotecas *OpenCV, PyTesseract e Numpy* atreladas a *Translation API*, o ambiente de desenvolvimento utilizado foi o *Atom* devido à sua intimidade com a linguagem escolhida e o fato de ser uma ferramenta gratuita.

O primeiro passo para a construção da aplicação é a importação das bibliotecas recorrentes da linguagem *Python* como visto no código abaixo.

```
import numpy
import cv2
import pytesseract
from googletrans import Translator
```

Após todas as bibliotecas importadas inicia-se o processo de análise da imagem. Por meio da variável ”imagem”faze-se o processo de aquisição da imagem utilizando da leitura da biblioteca *OpenCV* ”cv2.imread”, através do caminho ”imagem/imagemTraducao.png”.

```
imagem = cv2.imread("img/img.png")
```

Depois do início do processamento digital da imagem as bibliotecas *PyTesseract* e *Translator* são postas em uso.

```
translator = Translator()

pytesseract.pytesseract.tesseract_cmd =
 'C:/Program Files (x86)/Tesseract-OCR/
tesseract.exe'
```

Através da biblioteca *PyTesseract* obtém-se na análise da imagem o procedimento para pré-processamento e segmentação seguido da extração de características, que nada mais é que o reconhecimento ótico de caracteres OCR, utilizando o método pytesseract image to string(imagem, lang='eng') para interpretar nossa imagem anteriormente citada na sua linguagem de origem inglês.

```
print('Prototipo OCR:')
print('\nTexto traduzido:\n')

translations = translator.translate

([pytesseract.image_to_string(imagem, lang='eng')]
, dest='pt')
```

O método de extração das características da imagem foi concluído, agora tem-se um texto na linguagem inglês. A biblioteca *GoogleTrans* irá fazer a comunicação com a *Translation API* com dois parâmetros: texto em inglês, linguagem destino português.

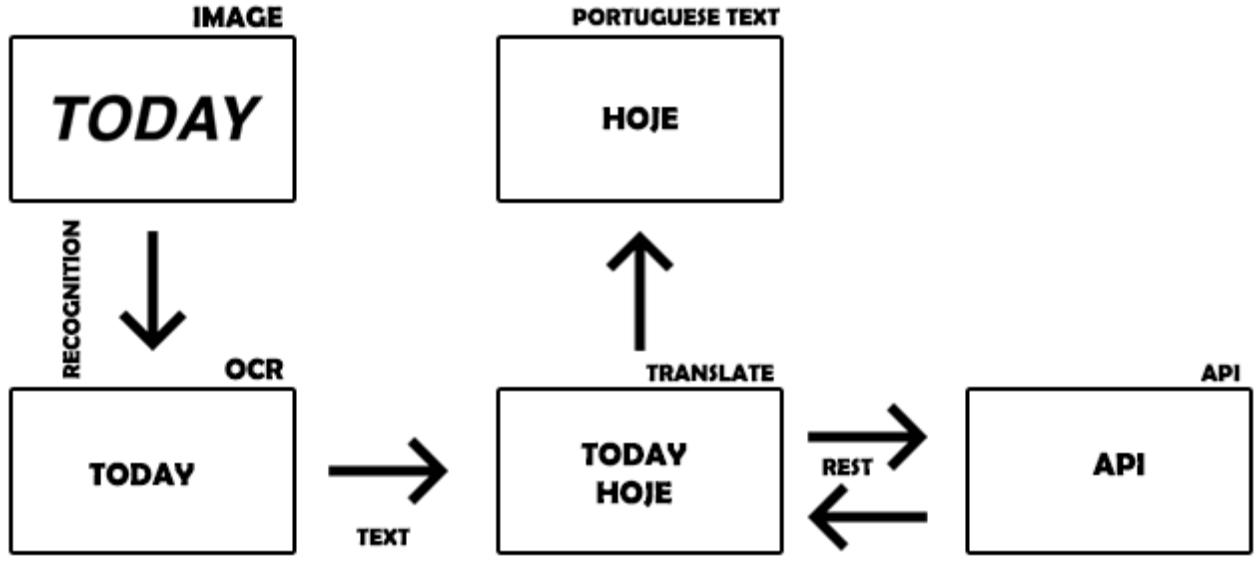

Figura 8. Diagrama de blocos da metodologia proposta para a viabilidade com aplicações de visão computacional de diferentes APIs. FONTE: O autor (2018)

```
print('Prototipo OCR:')

print('\nTexto traduzido:\n')

translations = translator.translate
([pytesseract.image_to_string(imagem,
lang='eng')]
, dest='pt')
```

A palavra em português e a imagem analisada são exibidas, mostrando o resultado da aplicação na prova de conceitos.

```
for translation in translations:
    print(translation.text)


cv2.imshow("Imagem original", imagem)
cv2.waitKey(0)
```

Com o código contruído, efetua-se a prova da aplicação. Executa-se a aplicação através do comando abaixo no *CMD*:

```
python AplicacaoOCR.py
```

A imagem retratada é uma manchete do site "www . reuters . com"(o Reuters é um site de uma agência de notícias britânica, a maior agência internacional de notícias do mundo, com sede em Londres).

Com todo o processamento executado, o resultado é exibido na tela.

Figura 9. Prova de conceitos - Texto. FONTE: O autor (2018)

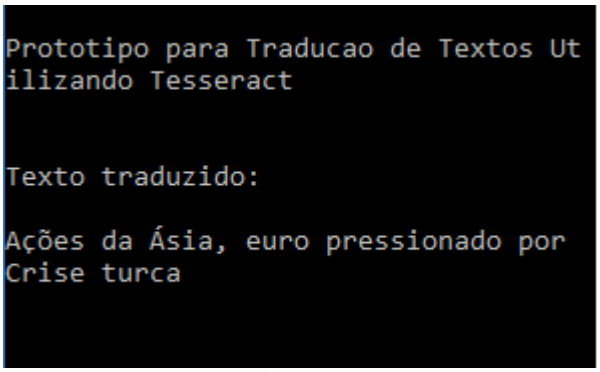

Figura 10. Prova de conceitos - Resultado. FONTE: O autor (2018)

Texto traduzido: Ações da Ásia, euro pressionado por Crise turca

6.2 Projeto comercial

O projeto comercial foi desenvolvido para a plataforma *mobile* com a linguagem de programação *java* em conjunto das bibliotecas *tess-two* e *OkHttp* atreladas a *yandex API*, o ambiente de desenvolvimento utilizado foi o Android Studio devido à sua intimidade com a linguagem escolhida e o fato de ser um ambiente essencial para o desenvolvimento de aplicações *android*.

Para iniciar o desenvolvimento do projeto, a aplicação deve fazer a importação, no arquivo ”build.gradle”, das duas bibliotecas essenciais para a comunicação entre o método *OCR* e o método de tradução gerado na *API*.

```
compile 'com.rmtheis:tess-two:8.0.0'
compile 'com.squareup.okhttp3:okhttp:3.11.0'
```

Após isso deve ser feito o *download* do arquivo ”eng . traineddata”, que é um conjunto de dados compactados interpretados para a análise de imagens contendo palavras da língua inglesa via OCR, o qual será anexado na pasta ”assets”dentro do projeto.

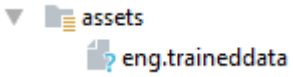

Figura 11. Projeto comercial - eng.traineddata. FONTE: O autor (2018)

Cria-se o método ”getText()”, por meio dele é enviado por parâmetro a imagem que passará pela extração dos caracteres.

```
private String getText(Bitmap bitmap){

}
```

Através do objeto ”tessBaseAPI”seta-se a imagem, o seu caminho e a linguagem a ser trabalhada no método OCR.

```
try{
tessBaseAPI = new TessBaseAPI();
}catch (Exception e){
Log.e(TAG, e.getMessage());
}
System.out.println(DATA_PATH);
String dataPath = getExternalFilesDir("/").
getPath() + "/";
System.out.println(dataPath);
tessBaseAPI.init(dataPath, "eng");
tessBaseAPI.setImage(bitmap);
String retStr = "No result";
```

Através do código abaixo é obtido na imagem o seu pré-processamento e segmentação seguido da extração de características, que nada mais é que o reconhecimento ótico de caracteres OCR, utilizando o método *tess-two* por meio do objeto *tessBaseAPI*.

```
retStr = tessBaseAPI.getUTF8Text();
System.out.println(retStr);
```

Seta-se o texto que foi retirado da imagem no campo de exibição visível ao usuário.

```
TRAD = this.findViewById(R.id.textView).
toString();
```

Ao decorrer do desenvolvimento do projeto comercial surgiram problemas na extração de caracteres em frases com duas linhas, uma abaixo da outra, ocorria um erro no processamento do texto causando uma falha na aplicação em um momento seguinte na parte de tradução. Abaixo uma parte do código utilizando o ”replaceAll”para deixar o texto corrente, a única solução encontrada.

```
String strOrigem = retStr;
strOrigem=strOrigem.replaceAll("\r", " ");
strOrigem=strOrigem.replaceAll("\t", " ");
strOrigem=strOrigem.replaceAll("\n", " ");
strOrigem=strOrigem.replaceAll(" ","%20");
```

Agora o método para fazer o parser do JSON entra em ação passando por parâmetro o texto extraído por OCR.

```
new JSONParse().execute(strOrigem);
```

Através da *URL* mostrada abaixo são passadas as seguintes variáveis (parâmetros): CHAVE (composta pelo código da chave para utilização da API Yandex), TEXTO (composta pelo texto extraído pelo OCR), LINGUAGEM (composta pela linguagem a qual vamos traduzir, português).

```
String url="https://translate.yandex.net/api/
v1.5/tr.json/translate?key=" + CHAVE + "
&text="+TEXTO+"&lang="+LINGUAGEM;
```

Resultado do *JSON* enviado com a palavra ”hello”.

```
{"code":200,"lang":"en-pt","text":["Olá"]}
```

Com o fim de todo o processe seta-se o resultado do texto traduzido para ser exibido ao usuário.

```
traducao1 = (TextView)findViewById
(R.id.traducao);
traducao1.setText(traducao);
```

Com o código construído, efetua-se a prova da aplicação. Com o fato de ser um aplicativo voltado para a plataforma *mobile* se torna bem simples a sua execução, podendo ser executado em qualquer aparelho que possua *android* na versão 4.1 ou superior.

Tela inicial do aplicativo pré captura da imagem.

Figura 12. Projeto comercial - Tela inicial pré captura da imagem. FONTE: O autor (2018)

Tela de captura da imagem.

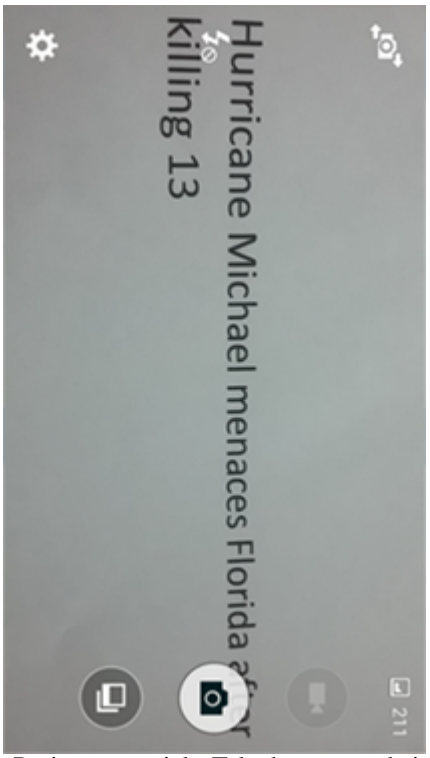

Figura 13. Projeto omercial - Tela de captura da imagem. FONTE: O autor (2018)

Tela inicial do aplicativo pós captura da imagem.

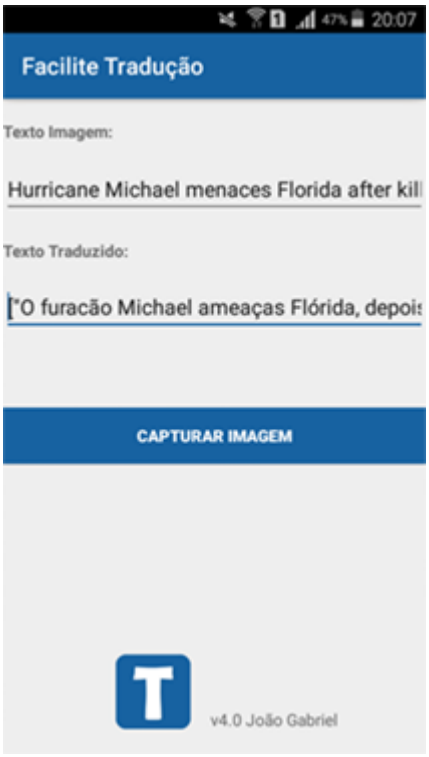

Figura 14. Projeto comercial - Tela inicial pós captura da imagem. FONTE: O autor (2018)

7. Resultados

As bibliotecas utilizadas para o reconhecimento ótico de caracteres foram a *Tess-Two OCR* e a *PyTesseract*, ambas são ramificações da biblioteca principal da *Google Tesseract OCR*. As *APIs* utilizadas para a tradução do texto retirado da imagem foram a *Translation API* e a *Yandex API*. Todo o processo pode ser visualizado na figura 8.

A prova de conceitos e o projeto comercial provaram a viabilidade das aplicações de visão computacional entrelaçando diferentes tipos de APIs, colaborando entre si para um mesmo resultado absoluto que seria de fato o outro objetivo de extrair os caracteres da imagem e em seguida tranduzindo-os de forma a retornar ao usuário.

Vale ressaltar que ambos os projetos utilizaram também de diferentes linguagens de programação (*Java* e *Python*) e ambientes de desenvolvimento (*Atom* e *Android Studio*).

8. Conclusão

Muitos trabalhos identificados na literatura, demonstram que as pesquisas têm se direcionado para a área de reconhecimento ótico de caracteres, na qual o uso de APIs e bibliotecas colaborativas em um mesmo projeto torna-se um pouco incomum. Neste artigo, foi exposto uma estratégia para o uso de OCR para tradução de palavras através do texto contido na imagem, com o uso de diferentes APIs, bibliotecas e linguagens de

programação. Provando de fato se é possível utilizar ténicas de visão computacional trabalhadas ao contexto de diferentes APIs e bibliotecas de reconhecimento ótico de caracteres para o auxílio na tradução de palavras.

Os resultados apresentados são promissores. Entretando, podem ser melhorados. Como trabalhos futuros, propõe-se que a aplicação terá como meta a criação de novas funcionalidades para o usuário. Dentre elas a possibilidade de tradução para outras línguas, não somente o a língua inglesa, e também a possibilidade que o usuário possa ouvir a tradução do texto retirado da imagem.

Referências

1 AZEVEDO, E. CONCI, Aura. Computação Gráfica. Geração de Imagens. *Visão Geral*, Brasil, v. 01, n. 1, p. 03–30, maio 2003.

2 MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. *Processamento de Imagens: breve histórico e exemplos de aplicações* , Brasil, v. 01, n. 1, p. 01–19, maio 1999.

3 pytesseract 0.2.4 *Disponível em: https://pypi.org/project/pytesseract/* . Acesso em: 25 de set. de 2018.

4 Java OCR *Disponível em: https://sourceforge.net/p/javaocr/wiki/Home/* . Acesso em: 25 de set. de 2018.

5 PyOCR *Disponível em: https://gitlab.gnome.org/World/OpenPaperwork/pyocr/* . Acesso em: 25 de set. de 2018.

6 Marcos Emílio Ekman Faber. *Disponível em: http://www.historialivre.com. Acesso em: 05 de jun. de 2018.*, 2018.

7 Tess4J *Disponível em: http://tess4j.sourceforge.net/* . Acesso em: 25 de set. de 2018.

8 NumPy *Disponível em: http://www.numpy.org/* . Acesso em: 25 de set. de 2018.

9 OpenCV *Disponível em: https://opencv.org/* . Acesso em: 25 de set. de 2018.

10 GoogleTrans *Disponível em: https://py-googletrans.readthedocs.io/en/latest/* . Acesso em: 25 de set. de 2018.

11 Tesseract OCR *Disponível em: https://opensource.google.com/projects/tesseract* . Acesso em: 25 de set. de 2018.

12 Tess-two *Disponível em: https://github.com/rmtheis/tess-two* . Acesso em: 24 de out. de 2018.

13 Web service: o que é, como funciona, para que serve? *Disponível em: https://www.opensoft.pt/web-service/* . Acesso em: 24 de out. de 2018.

14 API do Microsoft Translator *Disponível em: https://azure.microsoft.com/pt-br/services/cognitive-services/translator-text-api/* . Acesso em: 25 de set. de 2018.

15 Metade dos europeus aprendem uma língua estrangeira no 1º ciclo do ensino básico *Disponível em: https://www.publico.pt/2005/02/03/jornal/metade-dos-europeus-aprendem-uma-lingua-estrangeira-no-1 C2 BA-ciclo-do-ensino-basico-3934* . Acesso em: 1 de jun. de 2018.

16 Qual é a diferença de API, biblioteca e Framework? *Disponível em: https://cursos.alura.com.br/forum/topico-qual-e-a-diferenca-de-api-biblioteca-e-framework-38546* . Acesso em: 25 de out. de 2018.

17 Documentação da Cloud Translation API *Disponível em: https://cloud.google.com/translate/docs/* . Acesso em: 25 de set. de 2018.

18 Documentação da Yandex API *Disponível em: https://tech.yandex.com/translate/doc/dg/concepts/About-docpage/* . Acesso em: 25 de set. de 2018.

19 MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. *OCR Optical Character Recognition* , 1993.

20 José Carlos Paes de Almeida Filho. *Dimensões Comunicativas no Ensino de Línguas*, 1993.

21 Rafael C. Gonzalez e Richard E. Woods. *OCR Optical Character Recognition* , 1977.

22 Image Processing in OpenCV *Disponível em: https://docs.opencv.org/* . Acesso em: 12 de out. de 2018.

23 Glossary of OCR terms (as used in Tesseract) V0.04 *Disponível em: https://tesseract-ocr.repairfaq.org/tess*$_g$*lossary.html*.*Acessoem* : 24*deout*.*de*2018.

24 Daisuke Yamakawa e Noriaki Yoshiura. *Applying Tesseract-OCR to Detection of Image Spam Mails*, 2012.

25 Sehrish Malik e Do-Hyeun Kim. *A Comparison of RESTful vs. SOAP Web Services in Actuator Networks* , 2017.

26 Renata Dariva Costa. *A escrita além dos cuneiformes* , 2012.

27 Miss- Poonam A. Wankhede e Dr. Sudhir W. Mohod. *A Different Image Content-based Retrievals using OCR Techniques*, 2017.

28 José Eustáquio Rangel de Queiroz e Herman Martins Gomes . *Introdução ao Processamento Digital de Imagens*, 2001.