Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia, Projeto de nível Superior de Educação

Projeto de graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Eletricista Ênfase em Telecomunicações, à Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Sistema de leitura de caracteres numéricos em visores de máquinas de cartão para portadores de deficiência visual

Autor(a): Luiza Santos Thimoteo

Orientador: Prof. Michel Pompeu Tcheou, D.Sc.

Aprovado em 10 de Julho de 2019.

Banca Examinadora:

Michel Pompeu Tcheou, D.Sc (Orientador) - Faculdade de Engenharia UERJ

Lisandro Lovisolo, D.Sc - Faculdade de Engenharia UERJ

Douglas Mota Dias, D.Sc - Faculdade de Engenharia UERJ

Rio de Janeiro

2019

Ficha elaborada pelo autor através do
Sistema para Geração Automática de Ficha Catalográfica da Rede Sirius - UERJ

T443 Thimoteo, Luiza Santos
Sistema de leitura de caracteres numéricos em visores de máquinas de cartão para portadores de deficiência visual / Luiza Santos Thimoteo. - 2019.
56 f.

Orientador: Michel Pompeu Tcheou
Projeto Final apresentado à Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia, para obtenção do grau de bacharel em Engenharia Elétrica.

1. Reconhecimento de caracteres - Monografias. 2. OCR - Monografias. 3. Síntese de voz - Monografias. 4. TTS - Monografias. 5. Técnicas de pré-processamento - Monografias. I. Tcheou, Michel Pompeu. II. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. III. Título.

CDU 621.3

RESUMO

THIMOTEO, Luiza Santos. *Sistema de leitura de caracteres numéricos em visores de máquinas de cartão para portadores de deficiência visual*, no Rio de Janeiro. Projeto Final (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

A partir da aplicação da inteligência artificial, junto à visão computacional, sistemas voltados ao reconhecimento de caracteres (OCR) foram desenvolvidos, tais como o Tesseract. Tecnologias como esta convertem textos presentes em imagens para carácteres em formato digital, através de comparações de formas e contornos com base em seu banco de dados. Paralelamente, a tecnologia Text-to-Speech (TTS) foi aperfeiçoada utilizando sintetizadores de voz para identificação de características linguísticas de um texto, a fim de reproduzi-lo em formato de áudio. O áudio gerado é construído através da estimação dos padrões como idioma, entonação e duração de pronúncia. Esse trabalho foi elaborado para atender às demandas de portadores de deficiência visual por tecnologias assistivas, que visem incentivar a integração social destes. Já o sistema desenvolvido em Python, tem como objetivo receber uma foto de uma máquina de cartão contendo o valor da transação em seu visor. Através do pré-processamento e processamento da imagem, o sistema identifica os carácteres presentes, reconhece o campo do "valor" e realiza a síntese de voz, reproduzindo o áudio. Os testes foram realizados usando câmeras distintas, e em diferentes máquinas. O pré-processamento da imagem para aumento de eficiência nos resultados foi alcançada através da aplicação do ajuste na intensidade de brilho e contraste da imagem, seguido de um filtro em escalas de cinza. Dos resultados obtidos destaca-se a boa performance do Tesseract para diferentes tipos de letra, e a capacidade do TTS de gerar voz com pronúncia clara.

Palavras-Chave: Reconhecimento de caracteres. OCR. Síntese de voz. TTS. Técnicas de pré-processamento.

ABSTRACT

THIMOTEO, Luiza Santos. *Sistema de leitura de caracteres numéricos em visores de máquinas de cartão para portadores de deficiência visual*, no Rio de Janeiro. Projeto Final (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

The study of artificial intelligence applied to visual computing allowed the development of technologies dedicated to character recognition written on an image to text into machine-encoded (OCR), such as Tesseract. Tesseract compares characters' shapes and edges of an image and its database to provide a digital text as a result. Meanwhile, Text-to-Speech (TTS) technology was improved as well. Through linguistics patterns identification, TTS is able to converts normal language text into speech. The audio is built based on the language, intonation, and the duration of phonetic segments. The present work is dedicated to meet a necessity in assistive technology which focuses on people who have visual disabilities and who often experience difficulty performing activities of daily living. The program developed in Python aims to receive a card machine photo containing the price of a transaction in the screen. Through pre-processing and processing steps, the system recognizes the characters on an image, select the value part of the text and synthesizes speech to return it in audio format.

Pre-processing techniques were tested in different sources of photos and types of cards machines. The results optimization was achieved due to an adjustment of intensity in the brightness and contrast along with a gray scale filter. It is worth talking into consideration the Tesseract ability to recognize characters, along with the TTS capability to produce voice with a clear pronunciation.

Keywords: Character recognition. OCR. Synthesized speech. TTS. Pre-processing techniques.

LISTA DE FIGURAS

Figura 1 - Censo Demográfico sobre pessoas que se declararam com alguma deficiência, seus tipos e severidades, 2010 – IBGE.....7
Figura 2 - Diagrama do sistema desenvolvido. 9
Figura 3 - Estrutura do sistema.....11
Figura 4 - (a) Imagem original. (b) Imagem original aplicando um limiar de 30. (c) Imagem original aplicando um limiar de 10 [13]. 13
Figura 5 - Erosão do elemento estruturante B aplicado à imagem A repetidas vezes consecutivas [15]. 14
Figura 6 - – (a) Figura A. (b) Elemento B (c) Resultado da operação C [15]. 15
Figura 7 - Abertura seguido de Fechamento [16]. 15
Figura 8 - Reconstrução de caracteres por predição de contorno [16] 16
Figura 9 - Modelo Fujisaki [21]. 20
Figura 10 - Imagem teste (1) colorida.
Figura 11 - Imagem teste (1) escalas de cinza.
Figura 12 - Imagem (1) binarizada. 21
Figura 13 - Teste de ajuste da intensidade de brilho e contraste. 23
Figura 14 - Imagem de teste (1)
Figura 15 - Imagem de teste (1) binarizada
Figura 16 - Terminal Pycharm Teste (1). 23
Figura 17 - Imagem de Teste (2).
Figura 18 - Imagem de Teste (2) binarizada.
Figura 19 - Terminal do Pycharm Teste (2) 24
Figura 20 - Imagem de Teste (3).
Figura 21 - Imagem de Teste (3) binarizada.
Figura 22 - Terminal Pycharm Teste (3). 24
Figura 23 – Imagem de Teste (4).
Figura 24 – Imagem Teste (4) binarizada.
Figura 25 – Terminal Pycharm Teste (4).....25
Figura 26 – Imagem de Teste (5).
Figura 27 – Imagem de Teste (5) binarizada.
Figura 28 – Terminal Pycharm Teste (5).....25

Figura 29 – Imagem de Teste (6).
Figura 30 – Imagem de Teste (6) binarizada.
Figura 31 – Terminal Pycharm Teste (6).....26
Figura 32 – Imagem de Teste (7).
Figura 33 - Imagem de Teste (7) binarizada.
Figura 34 – Terminal Pycharm Teste (7).....27
Figura 35 – Imagem de Teste (8).
Figura 36 – Imagem de Teste (8) binarizada.
Figura 37 – Terminal Pycharm Teste (8).....27
Figura 38 – Imagem de Teste (9).
Figura 39 - Imagem de Teste (9) binarizada.
Figura 40 – Terminal Pycharm Teste (9).....28
Figura 41 – Imagem de Teste (10).
Figura 42 – Imagem de Teste (10) binarizada.
Figura 43 - Terminal Pycharm Teste (10). 29

SUMÁRIO

INTRODUÇÃO 7

MOTIVAÇÕES 7

OBJETIVO 8

ORGANIZAÇÃO TEXTUAL 10

1.SISTEMA DE RECONHECIMENTO DE CARACTERES NÚMERICOS 11

1.1.PRÉ-PROCESSAMENTO DA IMAGEM 12

1.1.1.BINARIZAÇÃO/THRESHOLD 12

1.2.PROCESSAMENTO DA IMAGEM 13

1.2.1.MORFOLOGIA 13

1.2.2.RECONHECIMENTO DE CARACTERES 16

1.2.3.FILTRO DE TEXTO 16

2.SISTEMA DE SÍNTESE DE VOZ 17

2.1.PROCESSAMENTO LINGUÍSTICO-PROSÓDICO 18

2.1.1.MODELO DE DETERMINAÇÃO DE DURAÇAO SEGMENTAL 19

2.1.2.MODELOS DE ENTONAÇÃO 19

2.2.PROCESSAMENTO ACÚSTICO 20

2.2.1.*SÍNTESE CONCATENATIVA* 20

3.RESULTADOS 21

3.1.SOLUÇÕES DE CONTORNO 21

3.2.TESTES DE SUCESSO 24

3.3.TESTES COM FALHA E ANÁLISE DE DEFICIÊNCIAS 28

4.CONCLUSÃO 29

REFERÊNCIAS 32

INTRODUÇÃO

MOTIVAÇÕES

O mercado está em constante busca por soluções inovadoras com o propósito de melhorar a qualidade de vida aos usuários. No entanto, grande parte dessas inovações se limita a indivíduos não portador de deficiências em geral. A área de Tecnologia Assistiva no Brasil ainda possui poucos recursos para estudos e pesquisas em desenvolvimento. No último Censo Demográfico realizado pelo IBGE em 2010, 45.6 milhões de pessoas declararam ter pelo menos um tipo de deficiência, seja do tipo visual, auditiva, motora ou mental/intelectual. O IBGE divulgou o levantamento ilustrado na Figura 1, onde a deficiência visual foi a mais abrangente nas declarações dos indivíduos, atingindo 3,5% da população.

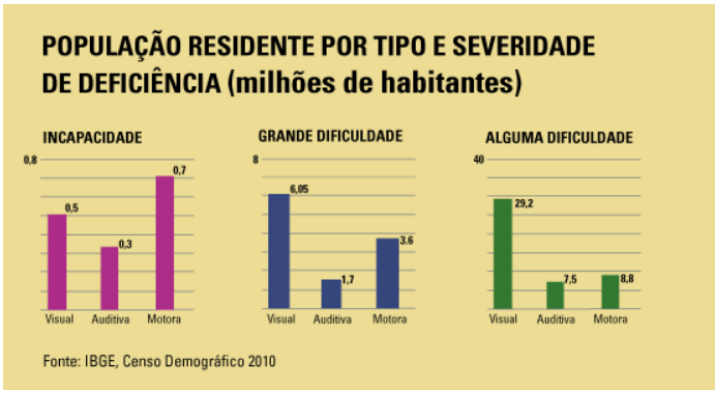


Figura 1 - Censo Demográfico sobre pessoas que se declararam com alguma deficiência, seus tipos e severidades, 2010 – IBGE

Frente a todos os problemas de falta de acessibilidade que um indivíduo portador de deficiência visual encontre no dia a dia, iremos neste projeto destacar a independência para realizar compras e efetuar pagamentos. Atualmente, todas as cédulas da segunda família do Real que estão em circulação têm marca tátil diferenciada para facilitar a acessibilidade. No entanto, esta técnica é muito criticada, pois além de ser somente a segunda família que apresenta a marca, há de se considerar o desgaste natural das células no decorrer do tempo.

Dentre as frentes de desenvolvimento de tecnologia assistiva, a mais notória se refere à aplicação da inteligência artificial e à visão computacional aplicada a técnicas correlacionadas ao reconhecimento de caracteres (OCR - *Optical Character Recognition)*. A tecnologia OCR converte texto de imagens em caracteres, sendo capaz de reconhecer diferentes linguagens a partir de comparações de formas e contornos, com padrões de um banco de dados, e contando ainda com a opção de treinamento do banco de dados para um novo idioma, que o usuário pode aplicar. Nesse contexto, diversos sistemas de reconhecimento de caracteres vêm sendo desenvolvidos, como o Tesseract [1].

Ao mesmo tempo em que tecnologias que exploram o campo visual são desenvolvidas, há também um grande avanço na área da síntese de voz. Os sintetizadores de voz, também conhecidos por Text-to-Speech ou TTS, são sistemas capazes de ler um texto, independentemente de sua fonte de escrita, e realizar a sua leitura em formato de áudio.

OBJETIVO

Visando melhor aproveitar as tecnologias disponíveis, este trabalho tem por objetivo o desenvolvimento de um sistema de reconhecimento de caracteres numéricos e, em seguida, síntese por voz de texto. Foi elaborado um sistema que atende a uma demanda de tecnologia assistiva voltada para portadores de deficiência visual.

O sistema proposto auxilia o portador de deficiência visual nos pagamentos do dia a dia, com objetivo de prover independência e segurança. Para os portadores de deficiência que não se encaixam na categoria de cegueira total, de acordo com a classificação para deficientes visuais segundo o Ministério da Educação e do Desporto do Brasil [2], consideramos que o indivíduo está apto a tirar uma foto aproximada da máquina de cartão. O sistema será então capaz de receber uma foto, identificar os caracteres que aparecerem na imagem, detectar o campo referente ao “valor” junto aos caracteres numéricos em seguida, para, então, realizar a síntese de voz, retornando em formato de áudio, o valor para o usuário.

O código do sistema está escrito em Python, utilizando o ambiente de desenvolvimento integrado Pycharm [3]. Tanto a linguagem quanto a ferramenta de interpretação escolhida visam a simplicidade na criação do código fonte. O Pycharm dispõe para o usuário o recurso do *virtualenv*, que permite ao usuário criar um ambiente virtual, totalmente isolado do sistema operacional da máquina do usuário. Essa ferramenta é ideal para a criação de códigos testes, pois torna possível ao desenvolvedor configurar cada ambiente com versões diferentes, por exemplo, a fim de se analisar em qual deles o sistema apresentará melhor desempenho. O Python oferece uma biblioteca com módulos de tecnologias já existentes adaptadas para a linguagem, como, por exemplo, OCR [4], TTS [5], além de uma biblioteca específica para a entrada de uma imagem no sistema [6].

Dessa forma, o sistema desenvolvido usa o PIL e o CV2, módulos de visão computacional que serão definidos e explicados a diante, para carregar e pré-processar a imagem. Em seguida, ela será trabalhada pelo Tesseract OCR Engine [7], responsável pelas técnicas de reconhecimento de caractere. O texto da imagem é então digitalizado e filtrado, de acordo com a região de interesse. Após ser recortado o campo referente ao “valor” junto aos

caracteres numéricos referentes a ele, será realizada a síntese de voz pelo Text-to-Speech, que realizará a detecção do idioma do texto e converterá o sinal de fala em formato de áudio, assim como mostrado na Figura 2.

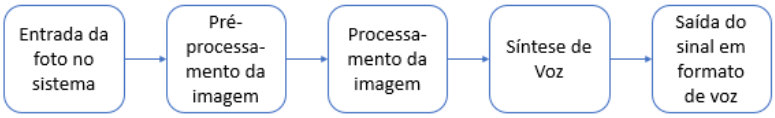


Figura 2 - Diagrama do sistema desenvolvido.

Para o desenvolvimento deste trabalho foram utilizados quatro módulos: CV2, PIL, TESSERACT e TTSX. Tais módulos são versões escritas em Python de sistemas originalmente desenvolvidos e fornecidos pelo OpenCV, que é uma biblioteca de visão computacional de código aberto.

- O **CV2** disponibiliza ao usuário o acesso a todos os códigos pertencentes ao OpenCV, possibilitando a importação de códigos com suas funções ali armazenadas.
- A biblioteca de imagens do Python responsável pela extensão nos tipos de formatos de imagem a serem reconhecidos pelo interpretador é chamada de **PIL**. A extensão também realiza a varredura dos bits gerando o histograma da imagem, que permite extrair informações da foto [6].
- O **Tesseract** pode ser utilizado como um programa, ou via API como uma aplicação de terminal. É um software gratuito e de código aberto, escrito em C++, que se classifica como um OCR Engine. A sigla OCR é definida por *Optical Character Recognition*, que significa reconhecimento ótico de caracteres, uma das áreas mais proeminentes na área de reconhecimento de padrões devido as suas inúmeras aplicações para a indústria e para o ramo de pesquisa. Resumidamente, OCR consiste no método de extração e identificação de caracteres, a partir de arquivos digitais. O sistema é capaz de reconhecer padrões de objetos presentes na imagem, coletando dados e os reunindo em um sistema, que será capaz de realizar a identificação automática deles [4].
- O **TTS** é um sistema de síntese de voz produzida artificialmente pela concatenação isolada de palavras, ou partes de sentenças. Os sistemas podem se diferenciar pelo banco de dados que utilizam, de acordo com o tamanho das unidades de fala armazenadas [8].

ORGANIZAÇÃO TEXTUAL

Este trabalho está estruturado em 4 capítulos, sendo o capítulo de introdução responsável por abordar a motivação, e o objetivo do desenvolvimento do sistema de leitura de máquina de cartão para portadores de deficiência visual.

Os próximos capítulos deste trabalho estão organizados da seguinte forma: o Capítulo 2 contém a arquitetura do sistema desenvolvido, composta pelas etapas de pré-processamento e processamento da imagem, seguido pela síntese de voz. O Capítulo 3 é composto pela descrição da fase referente a síntese de voz, abordando as técnicas aplicadas para geração do sinal de saída em formato de áudio. O Capítulo 4 apresenta uma análise sobre os principais resultados obtidos a partir do sistema desenvolvido neste trabalho, abordando os problemas encontrados, e avaliando possíveis técnicas de soluções de contorno, demonstrando a abordagem final utilizada para a otimização do sistema. O capítulo 5 é composto pela conclusão, com uma análise geral sobre os resultados alcançados, e uma reflexão sobre os métodos escolhidos.

1. SISTEMA DE RECONHECIMENTO DE CARACTERES NÚMERICOS

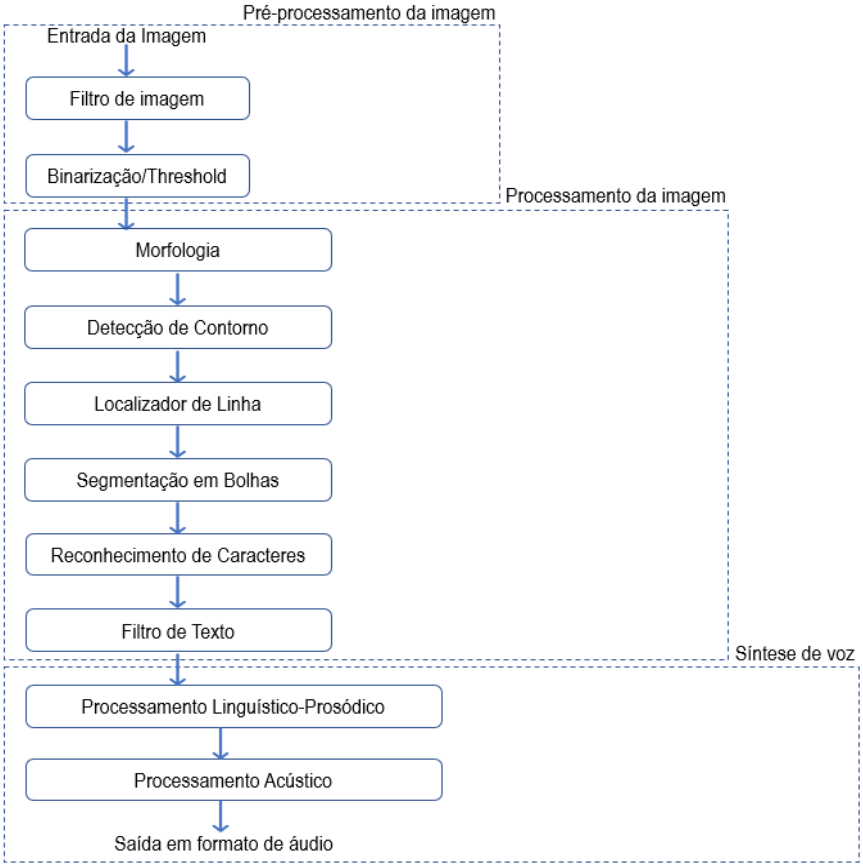


Figura 3 - Estrutura do sistema.

A Figura 3 ilustra a arquitetura do sistema desenvolvido, sendo composta pelas etapas direcionadas ao reconhecimento de caracteres, seguido da síntese de voz. A entrada é dada em formato de imagem, do tipo .PNG, e a saída é gerada em formato de áudio.

Após receber a entrada, técnicas de pré-processamento são aplicadas para otimizar a qualidade da imagem. O módulo CV2 é o mais recente desenvolvido para trabalhar de forma ágil e estável com a manipulação de arquivo de imagem. Ele contém todos os dados do OpenCV, disponibilizando para o usuário buscar dentro dele as funções de visão computacional desejadas para o *script.* Embasado em algoritmos que buscam modelar e replicar a visão humana através de softwares, a visão computacional utiliza uma imagem como parâmetro de entrada, para retornar um conjunto numérico tipo matriz e vetores como saída [9].

O texto da imagem será lido após seu processamento, realizado através de etapas que possibilitem ao sistema alcançar uma melhor taxa de acerto no reconhecimento de caracteres. O Pytessarct, sistema de OCR, é o responsável por buscar a correspondência do seu banco de

dados, com os caracteres identificados. Para que o sistema de OCR realize a leitura da imagem, o PIL é utilizado. Após a digitalização do conteúdo presente no arquivo de entrada, métodos de manipulação de *string* que visam a filtragem dos caracteres numéricos reconhecidos, são aplicados.

A fase de reconhecimento de caracteres do sistema desenvolvido é composta pelas etapas de pré-processamento e processamento da imagem recebida. O resultado do reconhecimento está diretamente ligado à qualidade da imagem que será lida e à qualidade da imagem está por sua vez, relacionada ao número de pixels presente na mesma [10].

1.1. PRÉ-PROCESSAMENTO DA IMAGEM

Iniciando a etapa de pré-processamento, um filtro preto e branco é aplicado à imagem. Visando a agilidade do sistema, a primeira técnica aplicada é o mapeamento dos pixels presentes na imagem em um sistema de coordenadas cartesianas (x,y). Utilizando (0,0) como o canto superior esquerdo, assume-se o referencial de que centro do primeiro pixel se encontra em (0.5,0.5), ou seja, o primeiro pixel termina em (1,1). Os dados de pixel podem ser recuperados pela indexação do mapa de pixels como uma matriz. No entanto, quando é chamado um comando para alteração, a imagem será alterada na sua origem, uma vez que o mapa de pixels é apenas uma referência aos dados originais da imagem [11].

1.1.1. BINARIZAÇÃO/THRESHOLD

O Limiar Adaptativo, também conhecido como o processo de Threshold, é responsável pela binarização dos pixels, sendo a próxima técnica aplicada ao sistema. Independentemente do método de processamento que se deseja aplicar na imagem, é necessário obter primeiramente a sua descrição. A segmentação por similaridade realiza uma classificação de pixel a pixel, dividindo a imagem em regiões de pixels com propriedades e classes semelhantes, segundo um determinado critério a ser estipulado. Na binarização, será aplicado um limiar de separação de intensidade para classificação entre “claro” e “escuro”.

Após definir-se o limiar de intensidade, todos os pixels da imagem serão analisados e comparados a este valor. Quando a intensidade do pixel for maior que o valor de limite estabelecido, ele passará a ser branco, senão, ele será preto, tornando a imagem binária.

O histograma de uma imagem, obtido a partir da varredura dos pixels, fornece características da imagem, que nos permite ter uma noção instantânea sobre o conteúdo dela. Considerado como uma boa aproximação à densidade de probabilidade, ele apresenta de forma

gráfica, os níveis presentes na escala de cinza contidos na imagem. É a forma mais usualmente utilizada de representação da distribuição de valores de pixel em uma imagem, podendo fornecer informações relevantes como características da intensidade [12].

Após a binarização, a imagem terá apenas preto e branco, possuindo assim apenas dois valores de intensidade em seu histograma: 0 (preto) e 255 (branco). Dessa forma, a técnica de limiarização é principalmente utilizada quando se deseja separar os objetos do fundo da imagem, realizando a distinção a partir da atribuição de preto, ao fundo, e branco, ao objeto, conforme equações 1 e 2 indicam. A Figura 4 ilustra três processos de limiarização aplicados com diferentes valores de Threshold(T). Observa-se que, quanto menor o limiar, maior a quantidade de pixels assumindo o valor de preto após o processo.

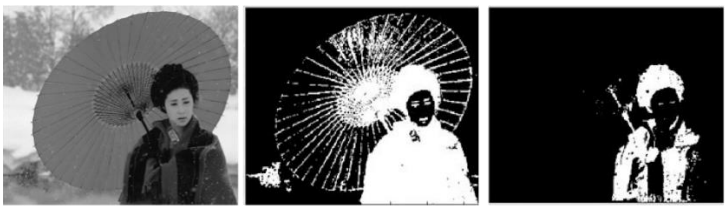


Figura 4 - (a) Imagem original. (b) Imagem original aplicando um limiar de 30. (c) Imagem original aplicando um limiar de 10 [13].

1.2. PROCESSAMENTO DA IMAGEM

1.2.1. MORFOLOGIA

Para iniciar a fase de reconhecimento, técnicas para extração de contorno dos caracteres serão aplicadas, a fim de segmentar as informações de contorno presentes na imagem. O conceito do método se baseia em identificar os pixels subjacentes que formam um único caractere. De modo a aumentar o contraste de intensidade entre os pixels de contorno e a região que o cerca, reforçando a delimitação do objeto em relação ao fundo, operadores morfológicos são aplicados.

A morfologia matemática se refere ao estudo e análise de imagens usando operadores não lineares, aplicando teoria dos conjuntos para a representação de objetos numa imagem. A técnica utiliza uma estrutura de pequeno porte para analisar pixel a pixel, comparando-os com seus vizinhos, buscando se determinar os contornos dos objetos.

A erosão e a dilatação são denominadas operadores elementares, pois fundamentam a base para a construção de transformações complexas. As operações são realizadas para

aprofundar o conhecimento topológico acerca da imagem, com o intuito de facilitar sua manipulação das mesmas a partir de um confronto de um conjunto desconhecido [14].

A **erosão** é a operação aplicada para diminuir a quantidade de informação presente na imagem, eliminando áreas de tamanho inferior ao elemento estruturante, e aumentando a região de separação entre duas regiões de informação vizinhas. A operação é composta por um conjunto A, de entrada, e um conjunto B, chamado de elemento estrutural, sendo matematicamente descrita por:

$$A \ominus B = \{ x : B + x \subseteq A \} = \{x: B_x \subseteq A\} \quad (1)$$

onde B_x é a translação de B pelo vetor x: $B_x = \{b + x | b \in B\}$

A operação descrita pela equação 9 consiste, em palavras, na translação do elemento estruturante B, por uma imagem de entrada A. O resultado da operação pode ser interpretado como a interseção de todas as possíveis translações de B dentro dos limites de A [15].

A Erosão da Figura 5 ilustra do elemento estruturante B sobre a imagem A, resulta no conjunto de pontos laranjas, que representam o centro do elemento estruturante B que já foi transladado por A.

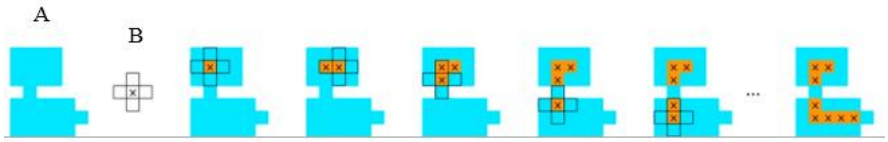


Figura 5 - Erosão do elemento estruturante B aplicado à imagem A repetidas vezes consecutivas [15].

A **dilatação** é a segunda operação morfológica básica, após a erosão. O objetivo da dilatação é o oposto da erosão: enquanto a erosão visa a diminuição de informação na imagem, a dilatação aumenta os limites de contorno do conteúdo pertencente à imagem, elevando a quantidade de informação contida. Dessa maneira, o buraco entre duas regiões que contenham informação e sejam vizinhas tende a diminuir, segundo [15], e conforme ilustrado na Figura 6. Matematicamente, a operação pode ser descrita por:

$$A \oplus B = [(A)' \ominus \check{B}]' \quad (2)$$

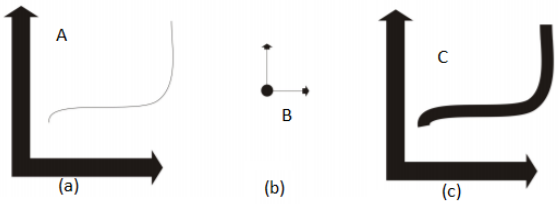


Figura 6 - – (a) Figura A. (b) Elemento B (c) Resultado da operação C [15].

A **abertura** é a operação de erodir e depois dilatar um conjunto, suavizando seus contornos a partir da eliminação de pontas, por exemplo, e tornando-o menos rico em detalhes. Ela visa nivelar os contornos presentes na imagem original, transformando-a em uma forma mais regular, a partir de mudanças pelo seu interior.

Ao contrário da abertura, o **fechamento** realiza a dilatação, seguido da erosão. É capaz de eliminar pequenos espaços entre contornos, assim como também preenche as fendas nos mesmos. Assim, o resultado será um conjunto mais regular e menos rico em detalhes. O objetivo da operação é buscar a reestruturação do conjunto resposta, sem modificar o tamanho e a forma dos respectivos conjuntos da operação.

O processo de abertura e fechamento tem por objetivo a melhoria da qualidade da imagem, eliminando-se ruídos, e aumentando os traços detalhados presentes na imagem. São dois processos largamente implementados em uma etapa de processamento da imagem.

Na Figura 7 observa-se inicialmente a figura ruidosa processada. Podemos ver os resultados das seguintes operações: a erosão, a abertura, a dilatação da abertura, e por fim o fechamento da abertura.

Figura 7 - Abertura seguido de Fechamento [16].

1.2.2. RECONHECIMENTO DE CARACTERES

Na etapa de reconhecimento de caracteres, as regiões trabalhadas nas fases anteriores serão avaliadas. Essa avaliação é feita com base na busca por correspondências de contorno que estejam presentes tanto na imagem processada, como na base de dados do Tesseract.

O processo de reconhecimento é finalizado após duas leituras. Após a primeira, cada palavra que já foi reconhecida é utilizada como base no banco de dados treinado para o reconhecimento de caracteres ainda não identificados [4]. Para os casos em que permanecerem a indeterminação, uma segunda leitura é realizada.

Enquanto o resultado continuar insatisfatório, o sistema busca cortar os caracteres das bolhas, como mostra a Figura 8, em que a reconstrução do caractere é feita por predição de seu contorno. Nesse momento, também são tratados os espaçamentos imprecisos (Espaços Fuzzy).

Somente as palavras com insucesso são encaminhadas a um associador, que elege uma lista de correspondências possíveis, atribuindo possíveis candidatos de associação dos fragmentos para os estados seguindo uma ordem de fila prioritária [17].

Figura 8 - Reconstrução de caracteres por predição de contorno [16]

Por fim, as características dos caracteres são comparadas com os padrões de alfabetos treinados, armazenados no banco de dados do Tesseract. Após o reconhecimento do alfabeto e dos caracteres, deve-se fazer a associação entre eles com a finalidade de se obter uma sequência de caracteres com significado, ou seja, eles devem ser agrupados de modo a formar palavras.

1.2.3. FILTRO DE TEXTO

Ao final da fase de processamento, o resultado obtido é a digitalização do texto reconhecido na imagem. Para o objetivo do sistema desenvolvido, é necessário a aplicação de um filtro capaz de recortar somente a região de interesse do texto que deve ser retornado em formato de voz. As imagens que serão processadas neste sistema contêm um conjunto de diferentes superfícies, porém apenas o visor da máquina de cartão apresenta uma superfície não-curvilínea, o que torna o processamento da região ideal para o reconhecimento de texto pelo Tesseract. Tendo em vista que o usuário não irá capturar a imagem apenas do visor da máquina de cartão, parte-se do pressuposto que há uma grande quantidade de informação

presente nas imagens. Dessa maneira, o sistema desenvolvido deve produzir a saída de áudio contendo apenas o preço a ser pago descrito no visor da máquina de cartão da imagem [18].

Com o propósito de facilitar seu processamento, foram avaliadas diferentes bibliotecas para realizar a seleção do texto que contém o preço a ser pago pelo usuário, como métodos capazes de reconhecer o visor, e técnicas de identificação de contornos retos, para uma segregação da superfície retangular do visor. No entanto, muitas máquinas ou não possuíam intensidade mínima na luz do led para sua detecção, ou não possuíam contornos em formato retangular para o visor. Algumas considerações foram pontuadas para obtenção da técnica de seleção de texto mais adequada:

1. Na mesma linha, precedendo o valor numérico referente ao preço a ser pago, o visor exibe: “valor”, “Valor”, “VALOR”, “R$”, “RS”, “Credito” ou “Debito”. Devido à dificuldade conhecida do Tesseract em reconhecer caracteres especiais como $, parte-se do pressuposto que o reconhecimento do cifrão como “S” também é uma leitura válida. A fim de simplificar o projeto, antes de buscar no texto essas palavras, é necessário a conversão de todo texto reconhecido para minúsculo.
2. A única vírgula identificada nas máquinas de cartão é a que determina a separação do valor inteiro em real, em relação aos caracteres referentes aos centavos.
3. Assume-se que há apenas duas casas numéricas possíveis para representar o valor em centavos referente ao preço.

Sendo assim, a técnica mais apropriada para a finalidade deste projeto foi obtida a partir de manipulação de *string*, levando em considerações os pontos de atenção detectados. O código final desenvolvido utilizando este método de filtragem pode ser obtido através de [19].

2. SISTEMA DE SÍNTESE DE VOZ

O objetivo da tecnologia text-to-speech (TTS) é receber como entrada um texto genérico, com vocabulário irrestrito, que retorne como resposta a síntese de voz em boa qualidade, realizando a conversão ortográfica-fonética.

O sintetizador ideal é considerado aquele em que a fala é natural e inteligível, se afastando da voz computadorizada característica que esses sistemas possuíam no início do desenvolvimento da tecnologia. A qualidade desses sistemas é julgada por sua semelhança com a voz humana e pela sua capacidade compreensão [20]. Uma voz mais verossímil requer uma variação correta de parâmetros prosódicos, que, por sua vez, são parâmetros foneticamente

descritos pela sua intensidade, variação melódica e organização temporal do discurso, o que fundamenta os principais objetos de estudo na manipulação para síntese de voz [21].

Dessa maneira, a intensidade da voz está relacionada com a energia presente no movimento vibratório, medido em Decibéis. A variação melódica é função da curva melódica, que depende da frequência média de vibração das cordas vocais, medida em Hertz. Já a organização temporal do discurso, é o tempo para reprodução falada, onde há pausas, tempo de elocução e o tempo total da articulação. A partir da definição dessas características fundamentais, é gerado a resposta ao TTS [21].

A arquitetura dessa parte do sistema que se refere síntese de voz é constituída por duas partes: Front-end e back-end. A primeira etapa realiza o processamento linguístico, enquanto a segunda realiza o processamento acústico [5].

2.1. PROCESSAMENTO LINGUÍSTICO-PROSÓDICO

Na primeira fase do sistema de síntese de voz, também identificada pelo front-end, há a normalização do texto como pré-processamento. Nela, há a conversão de números e abreviações em suas equivalências por escrito. A conversão é realizada com base em tabelas de referência que associam os símbolos dos caracteres às suas formas escritas. Ainda nesta parte, há as transcrições fonéticas para cada palavra, seguida da categorização do texto a ser lido de acordo com sua pontuação, que divide o texto em sentenças e frases. Dessa forma, o front-end é responsável pela prosódia do texto, característica fonética que representa a forma de emissão do som, assim como sua entonação, que é dada de acordo com os três parâmetros acústicos: frequência fundamental (f_o), duração e intensidade do som [22].

A etapa de análise linguística, que deve ser realizada antes do seu processamento, tem por objetivo identificar a separação silábica e a sílaba tônica. Em seguida, tem-se o processamento linguístico prosódico, que será dado através de parâmetros acústicos. Os parâmetros acústicos têm por finalidade dar entonação e ritmo à reprodução de voz. A entonação é obtida a partir da determinação da evolução temporal da fala, que é dada pela evolução temporal da frequência fundamental f_o. O ritmo é definido a partir da designação de duração dos segmentos de som, juntamente com a inserção de pausas. Por fim, a curva de energia do sinal acústico é o parâmetro acústico responsável pela intensidade sonora [5].

A representação fonético-prosódica pode ser vista como a etapa de transcrição fonética, sendo constituída por um conjunto de regras, que recorre a um dicionário com transcrições

fonéticas das palavras, ou com máquinas de estado que realizam a associação entre a escrita e a fonética [5].

Desta forma, o front-end é composto por duas etapas que visam fornecer naturalidade ao sinal acústico gerado: a modulação da duração segmental e a modulação da entonação. A modulação da duração segmental é a fase responsável pela definição do tempo de reprodução sonora por cada segmento a ser lido. A etapa do gerador segmental produz a concatenação efetiva dos fonemas, com a finalidade de produzir entonação, ritmo e intensidade de leitura semelhantes a energia que seria produzida com a leitura do trecho escrito por um ser humano [23].

2.1.1. MODELO DE DETERMINAÇÃO DE DURAÇAO SEGMENTAL

Diferentes modelos foram desenvolvidos com a finalidade de fornecer o ritmo adequado a leitura, a partir da definição apropriada da duração de segmentos sonoros para cada unidade fonética. A maior dificuldade para estes modelos está na identificação automática dos limites dos segmentos, uma vez que esta etapa é fundamental para que as associações ao banco de dados sejam feitas corretamente, a partir de modelos estatísticos. [5].

2.1.2. MODELOS DE ENTONAÇÃO

Os modelos de entonação são os responsáveis por mapear as informações linguísticas do contorno de f_o, com objetivo de prover modelização da frequência fundamental, que soma entonação à fala produzida. Os sistemas text-to-speech são formados por duas etapas para obtenção da entonação: a etapa linguística, e a etapa de geração de componentes de fo.

A etapa linguística consiste na transformação do texto de prosódia para uma representação abstrata codificada do contorno de f_o, a partir de modelos como Fujisaki. Fujisaki produz a melhor representação fonológica para leituras, possuindo comandos para separar f_o em duas componentes de características fundamentais, como pontuação e acentuação. Dessa forma, quanto maior a quantidade de marcação textual e de acentos, melhor tende a ser a aproximação de f_o. A sobreposição das funções logarítmicas de cada uma dessas características irá gerar as curvas de f_o. A Figura 9 mostra a descrição de contorno de f_o, feita através de eventos discretos, que entram em filtros lineares de segunda ordem criticamente amortecidos [21].

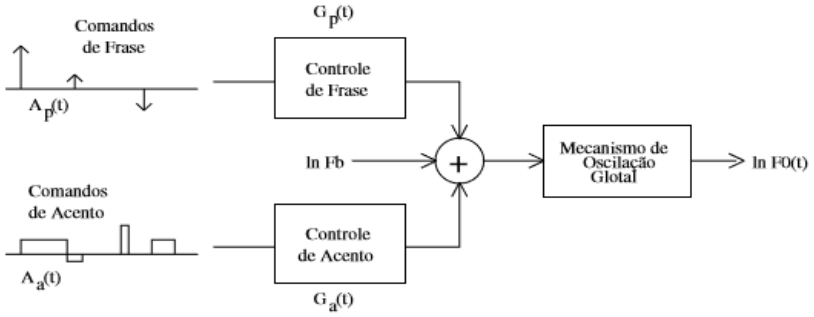


Figura 9 - Modelo Fujisaki [21].

Já a etapa de geração de componentes de f_o produz parâmetros de descrição para o comportamento do contorno de f_o. Esta saída da segunda etapa é que será a entrada para o modelo acústico, que reproduzirá as informações levantadas nas componentes para a geração da curva do sinal final de f_o [21].

2.2. PROCESSAMENTO ACÚSTICO

A segunda parte do sistema é o sintetizador, que realiza a conversão da representação linguística, gerada como produto do front-end, para o formato de áudio, levando em consideração os parâmetros para sua reprodução. Nesta fase, o sistema acessa um dicionário pré-estabelecido, que tem por fim relacionar a palavra escrita à sua pronúncia correta contextualizada disponível em um banco de dados. Dessa forma, a finalidade do back-end é a geração de uma forma de onda que carregue a informação de áudio a ser lida pelo alto-falante [23].

2.2.1. *SÍNTESE CONCATENATIVA*

É o método que realiza a concatenação de segmentos de voz previamente gravados para gerar o sinal resposta. Como concatenação significa encadeamento, essa técnica é a responsável por realizar a justaposição de duas unidades de dois segmentos de voz, visando a formação de uma saída única.

A partir de um dicionário, realiza as associações entre a unidade de texto a ser lido e o segmento de voz referente armazenadas em um banco de dados, que possui todas as unidades possíveis gravadas. Neste método não há necessidade de regra para a transição, pois a informação já está contida nas gravações da memória utilizada para a concatenação [21].

3. RESULTADOS

Após a estruturação do sistema para receber a imagem e realizar o processamento pelo Tesseract, observou-se a necessidade de técnicas de pré-processamento de imagem para que os resultados de reconhecimento de caracteres fossem bem-sucedidos. A síntese de voz não apresentou falhas, de forma que o retorno do sistema em formato de áudio está vinculado exclusivamente a leitura correta do texto contido na imagem.

Para o sistema desenvolvido neste projeto, o Text-to-Speech, que realiza a síntese concatenativa através do MBROLA [24], foi capaz de produzir a fala com qualidade satisfatória. Dessa forma, as soluções de contorno estudadas foram elaboradas para otimizar a taxa de acerto do sistema no reconhecimento de caracteres.

3.1. SOLUÇÕES DE CONTORNO

- TESTE 1

O sistema foi elaborado para primeiramente aplicar à imagem colorida uma escala de cinza. Em seguida, é aplicada a binarização à imagem, juntamente ao processo de limiar adaptativo. A Figura 10 apresenta a imagem colorida que foi lida no primeiro teste realizado usando o sistema desenvolvido. O código disponível em [19] retornou a Figura 11 após aplicar o filtro de escala de tons de cinza, e a Figura 12 apresenta a imagem binarizada.

Figura 10 - Imagem teste (1) colorida.

Figura 11 - Imagem teste (1) escalas de cinza.

Figura 12 - Imagem (1) binarizada.

A perda na qualidade da imagem da Figura 11 foi significativa, de forma que o reconhecimento de caracteres da Figura 12 falhou ao detectar os caracteres presentes na

imagem. A Figura 12 ilustra a imagem do texto no visor com degradação, devido ao mau desempenho do filtro de escala de cinza.

Métodos alternativos foram implementados para o ajuste do sistema, utilizando da biblioteca CV2 e seus módulos de visão computacional para tratar as falhas em cada etapa no pré-processamento da imagem. Pontuando-se as soluções de contorno abaixo, e suas respectivas avaliações, foram testados [25]:

1. Hough Line Transformation (HLT):

É uma transformação originalmente utilizada para a detecção de linhas retas em uma imagem binária. O reconhecimento de contornos é dado pela mudança brusca na intensidade de contraste de um pixel para o pixel vizinho. Durante o processamento de imagens, é comum surgirem problemas para a detecção de formas. Em muitos casos, a limiarização pode ser utilizado para reconhecimento de contornos, no entanto, devido a imperfeições tanto na imagem quanto no detector, podem ocorrer falhas como pixels faltantes, ou mesmo falhas no contorno que são originadas por ruídos e interferências. Dessa maneira, o propósito da Transformada de Hough é minimizar o erro na detecção de contornos, realizando a formação de grupos de pontos que sejam candidatos em potencial para serem classificados como pontos de contorno [26].

A aplicação do HLT no código de teste elaborado em [27], visa a detecção do contorno no formato retangular característico do visor da máquina de cartão, o que possibilitaria a implementação de um recorte dessa área de interesse da imagem. Conforme a técnica exige, a imagem de entrada foi primeiramente binarizada, passando pelo processo de limiarização, para então ser aplicada a transformada HLT. No teste realizado observou-se que o HLT não apresentou o reconhecimento de contornos conforme o desejado.

2. Controle e Ajuste de Brilho e Intensidade:

Diversos métodos para conversão da imagem colorida para escala em tons de cinza foram estudados, com a finalidade de se obter uma imagem filtrada que apresentasse maior resolução e melhor ajuste de brilho e intensidade, do que a imagem da Figura 11 gerada. A partir do código contido em [28], avaliou-se outras combinações de ajuste de brilho e contraste, conforme mostrado na imagem 13. Entretanto, as imagens finais não apresentaram melhores respostas.

Figura 13 - Teste de ajuste da intensidade de brilho e contraste.

Um novo teste foi realizado com a aplicação de um filtro em escala de tons de cinza precedendo a entrada da imagem original ao sistema. Para este teste, a imagem original foi previamente filtrada em escala de tons de cinza, de forma que a Figura 14 ilustra a imagem de entrada no sistema [19], e a Figura 15 é resultado da sua binarização. Em comparação à Figura 12, a Figura 15 apresentou menor degradação, o que permitiu o reconhecimento de caracteres no terminal do Pycharm, conforme Figura 16. A síntese de voz também não apresentou erros, produzindo o retorno de áudio conforme o esperado.

Figura 14 - Imagem de teste (1) com filtro de escala de cinza antes da sua entrada

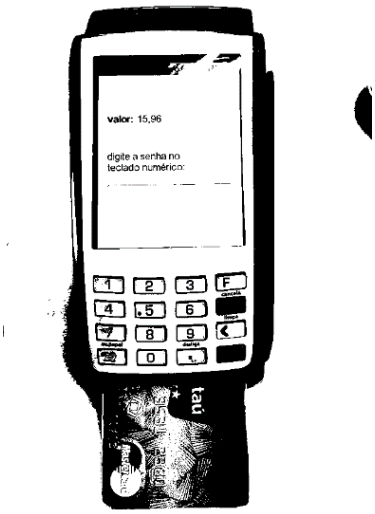


Figura 15 - Imagem de teste (1) binarizada

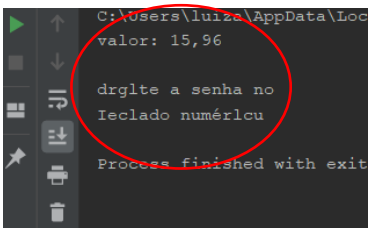


Figura 16 - Terminal Pycharm Teste (1).

3.2. TESTES DE SUCESSO

- TESTE 2

A Figura 17 mostra a imagem de entrada do sistema no teste 2, sendo a Figura 18 o resultado após sua binarização, e a Figura 19 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. A saída do sistema apresentou o valor correto em formato de áudio, sendo essa: “valor: duzentos e noventa e dois reais e zero zero centavos”.

Figura 17 - Imagem de Teste (2).

Figura 18 - Imagem de Teste (2) binarizada.

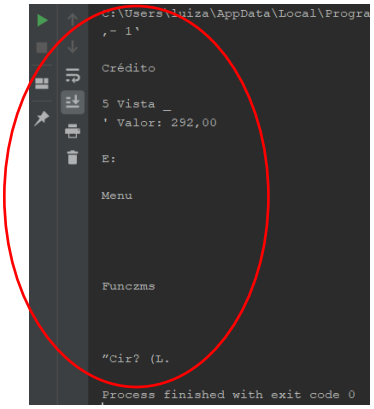


Figura 19 - Terminal do Pycharm Teste (2)

- TESTE 3

A Figura 20 mostra a imagem de entrada do sistema no teste 3, sendo a Figura 21 o resultado após sua binarização, e a Figura 22 o terminal do Pycharm. A saída do sistema apresentou o valor correto em formato de áudio, sendo essa: “valor: RS vinte e cinco reais e noventa e nove centavos”. Devido a deficiência do Tesseract em identificação de caracteres especiais, parte-se do princípio de que “R$” pode ser lido como “RS” sem afetar a compreensão do usuário.

Figura 20 - Imagem de Teste (3).

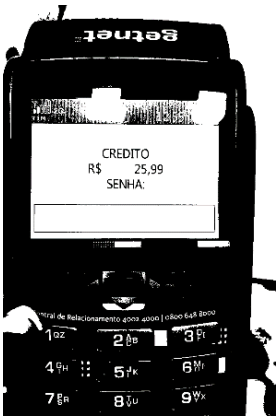


Figura 21 - Imagem de Teste (3) binarizada.

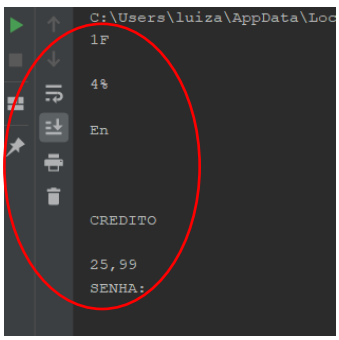


Figura 22 - Terminal Pycharm Teste (3).

- TESTE 4

A Figura 23 mostra a imagem de entrada do sistema no teste 4, sendo a Figura 24 o resultado após sua binarização, e a Figura 25 o terminal do Pycharm. A saída do sistema apresentou o valor correto em formato de áudio: “RS treze reais e noventa e nove centavos”. Devido a deficiência do Tesseract em identificação de caracteres especiais, parte-se do princípio de que “R$” pode ser lido como “RS” sem afetar a compreensão do usuário.

Figura 23 – Imagem de Teste (4).

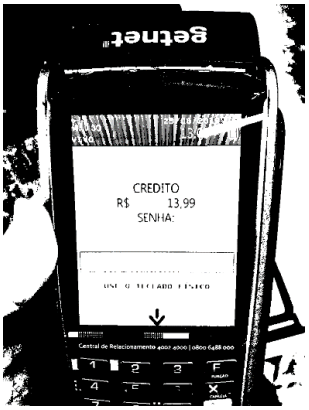


Figura 24 – Imagem Teste (4) binarizada.

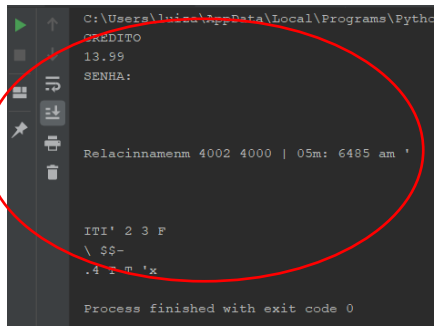


Figura 25 – Terminal Pycharm Teste (4).

- TESTE 5

A Figura 26 mostra a imagem de entrada do sistema no teste 5, sendo a Figura 27 o resultado após sua binarização, e a Figura 28 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. A saída do sistema apresentou o valor correto em formato de áudio, de acordo com a leitura da frase: “valor: vinte e três reais e noventa e oito centavos.

Figura 26 – Imagem de Teste (5).

Figura 27 – Imagem de Teste (5) binarizada.

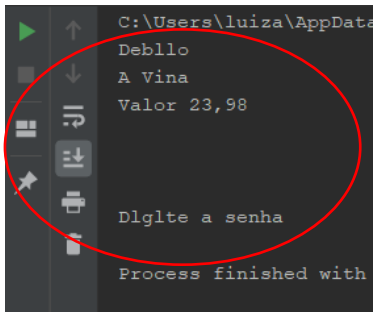


Figura 28 – Terminal Pycharm Teste (5).

- TESTE 6

A Figura 29 mostra a imagem de entrada do sistema no teste 6, sendo a Figura 30 o resultado após sua binarização, e a Figura 31 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. A saída do sistema apresentou o valor correto em formato de áudio, de acordo com a leitura da frase: “valor: treze reais e oitenta centavos”.

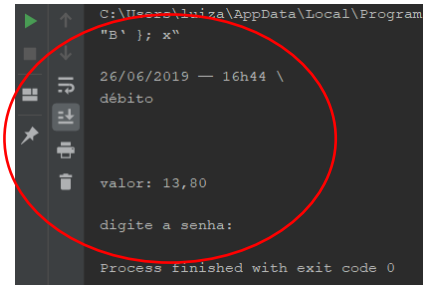
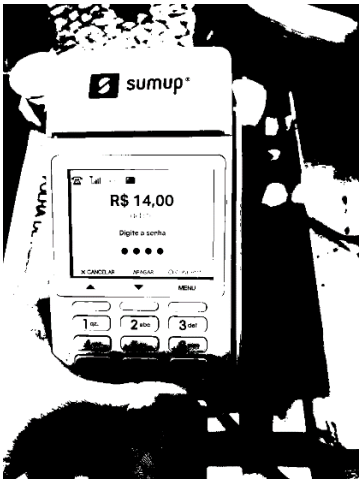


Figura 29 – Imagem de Teste (6). Figura 30 – Imagem de Teste (6) binarizada. Figura 31 – Terminal Pycharm Teste (6).

- TESTE 7

A Figura 32 mostra a imagem de entrada do sistema no teste 7, sendo a Figura 33 o resultado após sua binarização, e a Figura 34 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. A saída do sistema apresentou o valor correto em formato de áudio, de acordo com a leitura da frase: “RS catorze reais e zero zero centavos”. Devido a deficiência do Tesseract em identificação de caracteres especiais, parte-se do princípio de que “R$” pode ser lido como “RS” sem afetar a compreensão do usuário.



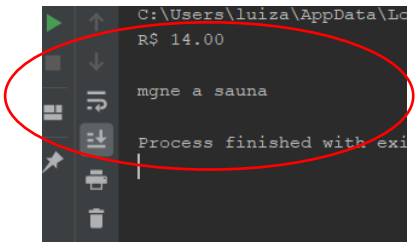
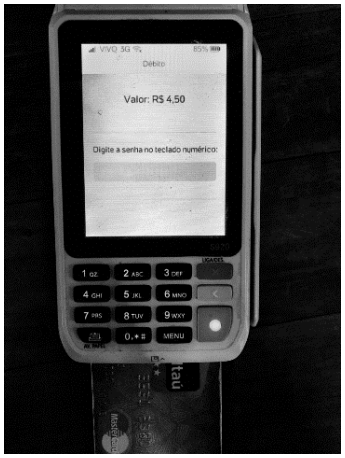
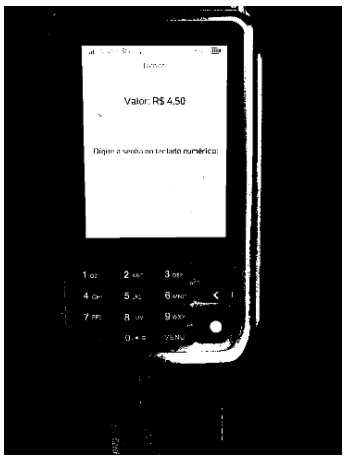


Figura 32 – Imagem de Teste (7). Figura 33 - Imagem de Teste (7) binarizada. Figura 34 – Terminal Pycharm Teste (7).

- TESTE 8

A Figura 35 mostra a imagem de entrada do sistema no teste 8, sendo a Figura 36 o resultado após sua binarização, e a Figura 37 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. A saída do sistema apresentou o valor correto em formato de áudio, de acordo com a leitura da frase: “valor: RS quatro reais e cinquenta centavos”. Devido a deficiência do Tesseract em identificação de caracteres especiais, parte-se do princípio de que “R$” pode ser lido como “RS” sem afetar a compreensão do usuário.





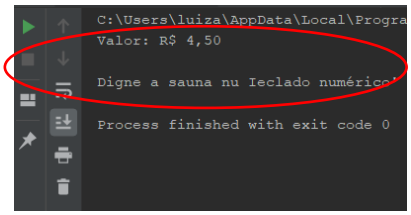


Figura 35 – Imagem de Teste (8). Figura 36 – Imagem de Teste (8) binarizada. Figura 37 – Terminal Pycharm Teste (8).

3.3. TESTES COM FALHA E ANÁLISE DE DEFICIÊNCIAS

- TESTE 9

A Figura 38 mostra a imagem de entrada do sistema no teste 9, sendo a Figura 39 o resultado após sua binarização, e a Figura 40 o terminal do Pycharm com os caracteres reconhecidos circulado em vermelho. Apesar do reconhecimento dos caracteres referentes a "Valor" e dos números corretos referentes ao preço da imagem da Figura 38 de entrada, a síntese de áudio não retornou o esperado devido a leitura incorreta de um ponto após a palavra "Valor".

A identificação incorreta de ":" como "." ocorreu devido à baixa qualidade da imagem de entrada, em que a intensidade de preto no símbolo ":" apresentou pouco contraste em relação a sua região de entorno. Dessa forma, o filtro de texto aplicado apresentou uma falha, que implicou no insucesso do teste devido a não leitura do trecho desejado em formato de voz.

Para que a fala produzida pelo sistema fizesse distinção dos reais e dos centavos, o caractere da vírgula foi utilizado para segmentar uma matriz que tivesse na coluna 1 os caracteres presentes depois de "valor" até a ocorrência da "vírgula", e, para a coluna 2, o sistema salva os 2 caracteres numéricos seguintes a ocorrência da "vírgula". Com o objetivo de minimizar os erros de reconhecimento entre "." e ",", todos os pontos foram transformados para vírgula antes da leitura em formato de voz, porém isso não preveniu o erro deste teste. Sendo assim, o resultado do teste 11 falhou devido a identificação de duas vírgulas na mesma linha da região de interesse do texto lido, o que tornou o sistema incapaz de compreender qual das duas vírgulas deveriam ser utilizadas para separar os reais dos centavos no momento da leitura.

Figura 38 – Imagem de Teste (9).

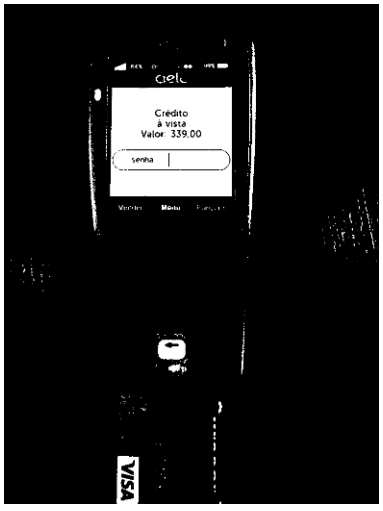


Figura 39 - Imagem de Teste (9) binarizada.

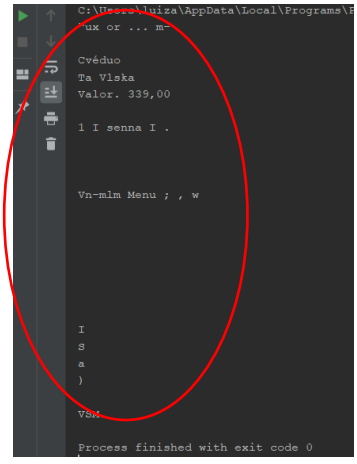
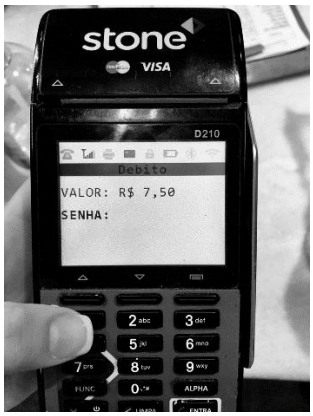
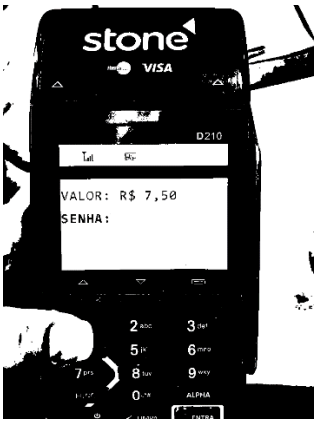


Figura 40 – Terminal Pycharm Teste (9).

- TESTE 10

A Figura 41 mostra a imagem de entrada do sistema no teste 10, sendo a Figura 42 o resultado após sua binarização, e a Figura 43 o terminal do Pycharm. Apesar da imagem binarizada apresentar texto legível, o sistema não foi capaz de reconhecer os caracteres presentes. A falha é passível de ser atribuída ao traçado fino da letra utilizada pelo formato de letra usados nesta máquina, o que pode ter gerado erro na etapa de limiarização.





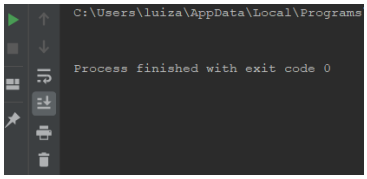


Figura 41 – Imagem de Teste (10). Figura 42 – Imagem de Teste (10) binarizada. Figura 43 - Terminal Pycharm Teste (10).

4. CONCLUSÃO

O sistema desenvolvido neste trabalho atingiu seu principal objetivo de retornar, em formato de áudio, o valor mostrado em visores de máquinas de cartões de diferentes tipos. As taxas de acerto foram consideradas boas, tendo em vista as dificuldades encontradas ao se trabalhar com imagens digitais provenientes de diferentes fontes, e, por vezes, capturadas em condição de luz ambiente deficiente.

Observou-se neste trabalho, a simplicidade do Python em produzir respostas de elevado nível de processamento, a partir de poucas linhas de código. A linguagem permite, por meio da importação de diferentes bibliotecas, a utilização de tecnologias capazes de gerar resultados complexos, fazendo utilização de um vasto banco de dados sem sobrecarregar o sistema operacional, ou mesmo causar lentidão de processamento. A interface do Pycharm foi essencial, pois permitiu o desenvolvimento de códigos testes em ambientes virtuais isolados, criados de forma prática, até que se fosse alcançada a melhor compatibilidade entre as versões do programa Python e das bibliotecas adicionadas.

Dentre as tecnologias utilizadas, destaca-se a habilidade do Tesseract em realizar o reconhecimento de caracteres, tendo em vista a grande variabilidade de tipos de máquinas de cartão avaliadas. O maior obstáculo no desenvolvimento desse sistema, foi alcançar a técnica a melhor técnica para segregação do texto a ser retornado como sinal de áudio na saída, de forma a atender o objetivo principal do sistema de realizar apenas a leitura do campo do "valor" referido no visor.

O Text-to-Speech não apresentou pontos de atenção, sendo capaz de gerar voz de fácil compreensão, e com pronúncia clara. Além disso, notou-se a habilidade da tecnologia TTS em realizar a leitura correta de caracteres numéricos em formato de numerais cardinais, ao invés de somente reproduzir o som referente a cada número em individual, sem reconhecer o conjunto de classes a que pertencem.

A partir dos resultados obtidos, observou-se a dificuldade do Tesseract em identificar corretamente caracteres numéricos de um determinado formato de letra encontrado nas imagens referente ao teste 10. Ao todo, 19 testes foram realizados neste trabalho, porém, somente 12 foram neste apresentado. Dentre os 19 testes feitos, 7 apresentaram falha. Realizando uma análise sobre os 5 testes sem sucesso, 4 continham o mesmo padrão de letra do teste número 10. Para os outros 3 testes que falharam, atribuiu-se o erro a má resolução da imagem, ou das condições de ruídos e luminosidade presentes, que resultaram na degradação da imagem. Dentre os testes de sucesso, observou-se a habilidade do Tesseract em realizar a identificação dos textos mesmo sem a correção de angulação entre a imagem capturada e o alinhamento referente ao texto da imagem.

Foram avaliados 2 possíveis métodos com o propósito de otimização dos resultados: o HLT, e um ajuste de intensidade de brilho e contraste. A transformada HLT, aplicada com o intuito de possibilitar o recorte da região do visor da máquina de cartão, gerando uma nova imagem com menos informação a ser processada, não demonstrou funcionar para o objetivo desejado. A implementação do filtro em escala de tons de cinza na fase que precede a entrada da imagem ao sistema, demonstrou que é possível a obtenção de uma combinação de valores para intensidade de brilho e contraste, que seguido da aplicação da escala em tons de cinza, fornece a melhor condição da imagem para o seu processamento. Imagens filtradas em escala de cinza que forneceram resultados corretos também foram avaliadas no sistema com a aplicação do HLT, de forma que o resultado obtido passou a apresentar erro para esses casos.

Como sugestão para trabalhos futuros, indica-se um estudo aprofundado sobre as possíveis adaptações de implementação para métodos de detecção de contorno, buscando outras técnicas e funções que forneçam resultados eficientes. Esse trabalho foi desenvolvido com o intuito de fundamentar a posterior criação de um aplicativo voltado para smartphone, que realize em tempo real, a identificação dos caracteres visto a partir da câmera do celular, retornando-os em formato de áudio para o seu usuário.

Recomenda-se também a atualização da versão do Tesseract para a nova versão 4.00, que adiciona um novo mecanismo de OCR baseado em redes neurais do tipo LSTM (*Long short-term memory*). Esse mecanismo do Tesseract 4.00 possui uma linguagem de descrição de rede chamada VGSL (*Variable Graph Specification Language*), que se baseia nos fundamentos de um ramo da inteligência artificial, denominado de machine learning. O machine learning representa o aprendizado de máquinas a partir de dados e reconhecimentos de padrões, trazendo um grande avanço para a tecnologia OCR [29].

Esse trabalho precisou ser reduzido para 34 páginas para se enquadrar dentro do regulamento do IX PRÊMIO CREA RJ 2020/2021.

REFERÊNCIAS

1. TESSERACT-OCR. Google. Disponivel em: <https://code.google.com/p/tesseract-ocr>.
2. QUEM é o deficiente Visual e o Cego. Portal da Educação. Disponivel em: <https://www.portaleducacao.com.br/conteudo/artigos/educacao-fisica/quem-e-o-deficiente-visual-e-o-cego/31979>.
3. ISLAM,. Mastering Pycharm. Birmingham : Packt Publishing , 2015.
4. PATEL, C.; PATEL, A.; PATEL, D. Optical Character Recognition by Open Source OCR, v. 55, n. International Journal of Computer Applications, 2012.
5. TEXEIRA, J. P.; BARROS, M. J.; FREITAS, D. Sistemas de Conversão Texto-Fala. Instituto Politécnico de Bragança e Faculdade de Engenharia da Universidade do Porto. Porto.
6. PILLOW Manual. Pillow Read the Docs. Disponivel em: <https://pillow.readthedocs.io/en/stable/handbook/overview.html>.
7. OHLSSON, V. Optical Character and Symbol Recognition using Tesseract. Luleå University of Technology. Bótnia Setentrional. 2016.
8. SPEECH synthesis. Semantic Scholar.
9. TUTORIAL PyThresholding. OpenCV. Disponivel em: <https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html>.
10. MOREIRA, L. M. C. Tamanho e resolução da image. Faculdade de Engenharia do Porto. Porto.
11. PILLOW Handbook. Pillow Documents. Disponivel em: <https://pillow.readthedocs.io/en/5.2.x/handbook/index.html>.
12. PADILHA, A. J. Processamento de Imagens - Segmentação de Imagem. Faculdade de Engenharia da Universidade do Porto. Porto. 2016.
13. MELO, N. Abordagens do processo de Segmentação: Limiarização, Orientada a Regiões e Baseada em Bordas. Jornal da Universidade Federal de Campina Grande, Campina Grande, Setembro 2011. Disponivel em: <http://www.dsc.ufcg.edu.br/~pet/jornal/setembro2011/materias/recapitulando.html>.
14. CONCI, A. Morfologia Matemática em Análise de Imagens. Universidade Federal Fluminense. [S.l.].
15. NETO, R. A. V. Introdução à Morfologia Matemática Binária e em Tons de Cinza. Universidade Estadual de Londrina. Londrina. 2010.
16. BEBIS, G. et al. Advances in VIsual Computing. [S.l.]: Springer.
17. SMITH, R. An Overview of the Tesseract OCR Engine. Google Inc.

18. THIMOTEO, L. Reading with Tesseract and TTS. Github, 2019. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/testing-tesseract-and-tts>.
19. THIMOTEO, L. Reading Card Machine. Github, 2019 Junho 2019. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/reading-card-machine>.
20. DUTOIT, T. An Introduction to Text-To-Speech Synthesis. Bélgica: [s.n.], 1997.
21. SILVA, S. D. S. Um estudo de modelos básicos de prosódia para o português brasileiro. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2004.
22. PACHECO, F. S. Sistema de Síntese de Fala. Ilha Digital, 2010.
23. SÁ, F. C. D. Geração de prosódia para o português brasileiro em sistemas text-to-speech. Universidade Federal do Rio Grande do Norte. Natal. 2018.
24. GOOGLE. MBROLA Voices. Disponivel em: <https://chromium.googlesource.com/chromiumos/third_party/espeak-ng/+/HEAD/docs/mbrola.md>.
25. MUKHOPADHYAY, S. Improving the Efficiency of Tesseract OCR Through Super Resolution. University of Hyderabad. Hyderabad. 2017.
26. JI, J.; CHEN, G.; SUN, L. A novel Hough transform method for line detection by enhancing accumulator array. Pattern Recognition Letters, China, 2009.
27. THIMOTEO, L. Applying HLT. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/testing-HLT>.
28. THIMOTEO, L. Adjusting Brightness and Contrast. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/testing-brightness-contrast>.
29. NEURAL Nets In Tesseract 4.00. Disponivel em: <https://github.com/tesseract-ocr/tesseract/wiki/NeuralNetsInTesseract4.00>.
30. MARTELLI, A. PYTHON IN A NUTSHELL - A desktop Quick and Reference. [S.l.]: O'Reilly, 2003.
31. LICENÇA Pytesseract. Github. Disponivel em: <https://github.com/madmaze/pytesseract/blob/master/LICENSE>.
32. PATENT, U. S. Dialec Translator For a Speech Application Enviroment Extended For Interactive Text Exchange. US 8,204,182 B2 , 19 Junho 2012.
33. BLOB. Techterms. Disponivel em: <https://techterms.com/definition/blob>.

34. SILVA, F. R. Desenvolvimento de Software Multiplataforma aplicado ao Diagnóstico de Consumo de Energia Elétrica. Universidade Estadual de Londrina. Londrina. 2016.

35. RIBEIRO, L. M. Desenvolvimento de Biblioteca Paralela em Python. Universidade de São Paulo. São Paulo. 2011.

36. MAGALHÃES, J. P. Detecção de Objetos Usando Características de Histograma Espacial. Universidade Federal de Pernambuco. Pernambuco. 2008.

37. DIAS, F.; COUSTY, J.; NAJMAN, L. Dimensional operators for mathematical morphology on simplicial complexes. Université Paris-Est. Paris. 2018.

38. BASTOS, E.; FREITAS, J. Interpretador/Compilador Python. Universidade Catolica de Pelotas. Pelotas.

39. LABAKI, J. Introdução a Python. Universidade Estadual Paulista. [S.l.].

40. BAUCHSPIESS, A. Introdução aos Sistemas Inteligentes. Universidade de Brasília. Brasília.

41. PAULA, E. M. S. D.; SOUZA, M. J. N. D. Lógica Fuzzy como técnica de apoio ao Zoneamento Ambiental. Universidade Estadual do Ceará. Fortaleza. 2007.

42. MBROLA. espeak source forge. Disponivel em: <http://espeak.sourceforge.net/mbrola.html>.

43. TOROK, L. Método de Otsu. Universidade Federal Fluminense. Rio de Janeiro.

44. AFONSO, A.; VICENTE, N.; ANTUNES, R. Microsoft Speech API (SAPI). Instituto Politécnico de Castelo Branco. Castelo Branco. 2007.

45. OPENCL. OpenCV. Disponivel em: <https://opencv.org/opencl/>.

46. PYCHARM. JetBrains. Disponivel em: <https://www.jetbrains.com/pycharm/>.

47. PYTHON e Virtualenv - Como programar em ambientes virtuais. Python Academy. Disponivel em: <https://pythonacademy.com.br/blog/python-e-virtualenv-como-programar-em-ambientes-virtuais>.

48. PYTTSX3. Read the Docs Pyttsx3. Disponivel em: <https://pyttsx3.readthedocs.io/en/latest/support.html>.

49. THIMOTEO, L. Reading Card Machine with CED. Github, 2019. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/reading-card-machine-withCED>.

50. THIMOTEO, L. Reading Card Machine With HLT e CED. Github, 2019. Disponivel em: <https://github.com/luizathimoteo/reading-price-in-card-machine/blob/master/reading-card-machine-with-HLT-CED>.

51. CONCI, A.; MONTEIRO, L. H. Reconhecimento de Placas de veículos por imagem. Universidade Federal Fluminense. Rio de Janeiro.