

**INSTITUTO FEDERAL DE MATO GROSSO DO SUL
COORDENAÇÃO DE TECNOLOGIA E INFORMAÇÃO
CURSO TÉCNICO INTEGRADO EM INFORMÁTICA PARA INTERNET**

**GABRIEL DE SOUZA GÓIS
GABRIEL TOMAZINI MARANI**

**TEXT RECOGNITION: UMA ABORDAGEM BASEADA EM
INTELIGÊNCIA ARTIFICIAL PARA ACESSIBILIDADE DE
DEFICIENTES VISUAIS**

**NAVIRAÍ
2019**

GABRIEL DE SOUZA GÓIS
GABRIEL TOMAZINI MARANI

**TEXT RECOGNITION: UMA ABORDAGEM BASEADA EM
INTELIGÊNCIA ARTIFICIAL PARA ACESSIBILIDADE DE
DEFICIENTES VISUAIS**

Trabalho de Conclusão de Curso
apresentada como requisito para a
conclusão do Curso Técnico
Integrado em Informática para
Internet.

Orientador: Maximilian Jaderson de
Melo
Co - Orientador: Guilherme
Figueiredo Terenciani

GABRIEL DE SOUZA GÓIS
GABRIEL TOMAZINI MARANI

**TEXT RECOGNITION: UMA ABORDAGEM BASEADA EM
INTELIGÊNCIA ARTIFICIAL PARA ACESSIBILIDADE DE
DEFICIENTES VISUAIS**

Trabalho de Conclusão de Curso
apresentada como requisito para a
conclusão do Curso Técnico
Integrado em Informática para
Internet.

Este exemplar corresponde à
redação final da dissertação
defendida e aprovada pela banca
examinadora em __/__/____.

BANCA EXAMINADORA

Prof. Maximilian Jaderson de Melo – orientador
IFMS

Prof. Guilherme Figueiredo Terenciani– co - orientador
IFMS

Prof. Danilo Mikucki - Avaliador Banca
IFMS

Prof. Rodrigo Oliveira - Avaliador Banca
IFMS

DECLARAÇÃO DE AUTORIA TEXTUAL E DE INEXISTÊNCIA DE PLÁGIO

Nós, Gabriel de Souza Góis e Gabriel Tomazini Marani, estudantes do Curso Técnico Integrado em Informática para Internet, declaramos que este texto final de dissertação é de nossa autoria e não contém plágio, estando claramente indicadas e referenciadas todas as citações diretas e indiretas nele contidas. Estamos cientes de que o envio de texto elaborado por outrem e também o uso de paráfrase e a reprodução conceitual sem as devidas referências constituem prática ilegal de apropriação intelectual e, como tal, estão sujeitos às penalidades previstas no IFMS e às demais sanções da legislação em vigor.

Naviraí, de dezembro de 2019.

Gabriel de Souza Góis

Gabriel Tomazini Marani

Dedicamos este trabalho acima de tudo para nossas mães que sempre foram compreensivas e serviram de base para esta realização.

AGRADECIMENTOS

Acima de tudo, agradecemos a Deus, por guiar-nos nesta jornada da longos três anos, agradecemos também o Professor Maximilian por sempre estar disponível e disposto a ajudar tendo uma enorme contribuição não só para este trabalho mas para o curso inteiro e o Professor Guilherme Terenciani que também fez parte deste trabalho e contribuiu para nossa formação acadêmica. Agradecemos nossas mães por sempre acreditarem em nós e serem compreensivas ao longo de todo o curso e também agradecemos amigos que fizeram esta caminhada juntos mesmo às vezes com alguns tropeços.

"A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo"

(Albert Einstein)

TEXT RECOGNITION: UMA ABORDAGEM BASEADA EM INTELIGÊNCIA ARTIFICIAL PARA ACESSIBILIDADE DE DEFICIENTES VISUAIS

RESUMO

Pessoas com qualquer tipo de deficiência sofrem em seu cotidiano com a falta de acesso a meios que visem a melhora de suas vidas sem que sua claudicância influencie ao ponto de não ser possível a inclusão. Em decorrência dessa realidade é necessário que haja o desenvolvimento de um dispositivo composto em parte de hardware e em parte de software, no qual ao final de todos os seus respectivos desenvolvimentos exista em mãos um tipo diferenciado de óculos, que será voltado às pessoas que possuem algum tipo de deficiência visual. Assim, o objetivo deste projeto é o estudo e desenvolvimento de um software de inteligência capaz de reconhecer caracteres presentes em imagens (OCR), de compreender e aprender novas palavras e sintaxe, de modo a ser empregado na construção do dispositivo de acessibilidade.

Palavras-chave: Acessibilidade, Processamento Digital de Imagens, Tecnologia Assistiva.

ABSTRACT

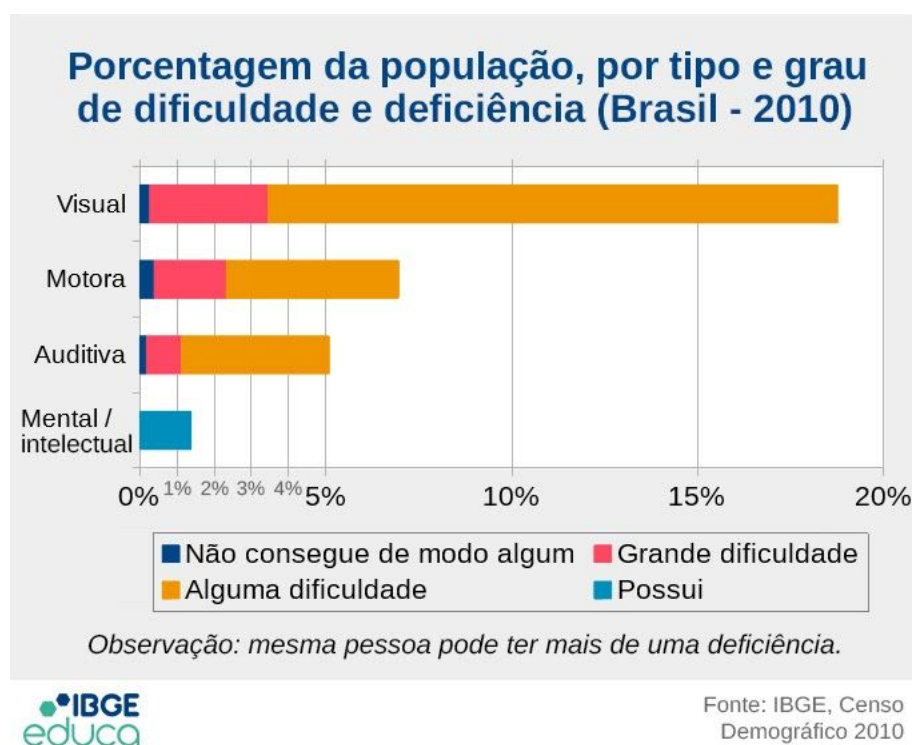
People with any kind of disability suffer in their daily lives from lack of access to technologies intended to improve their life quality and social inclusion. These facts justify the design of a device composed by both hardware and software. This device aims in the concept of a modified kind of glasses intended to visually impaired people. Thus, this project proposes the study and development of an artificial intelligence based software capable of recognize optical characters (OCR) in digital images, to comprehend and to learn new words and syntax. This software is meant to be part of the described device.

Keywords: Accessibility, Digital Image Processing, Assistive Technology.

1.0 - INTRODUÇÃO

Segundo dados da IBGE 2010, existem aproximadamente 6,5 milhões de pessoas no Brasil com deficiência visual sendo 528.624 pessoas com deficiência visual total. Pessoas com qualquer tipo de deficiência sofrem em seu cotidiano com a falta de acesso a meios que visem a melhora de suas vidas sem que sua claudicância influencie ao ponto de não ser possível a inclusão. “Deve-se promover a busca por meios de acessibilidade para que haja supressão de barreiras e obstáculos em locais públicos para pessoas com deficiência ou mobilidade reduzida” (BRASIL, 2000). Pode-se identificar pessoas que possuem algum nível ou grau de deficiência visual, estas pessoas encontram dificuldades em seu cotidiano, por exemplo lendo um cardápio num restaurante, ou uma placa de trânsito.

Figura 01 - Gráfico da pessoas com dificuldade ou deficiência



Fonte: IBGE¹

¹

O gráfico acima descreve a porcentagem da população brasileira por tipo e grau de dificuldade e deficiência, de modo que é possível observar que a maior parte desta porcentagem sofre com algum tipo de dificuldade quanto a questão visual, em segundo com algum tipo de dificuldade motora, em terceiro com algum tipo de dificuldade auditiva e em quarto com algum tipo de dificuldade mental/intelectual. Desse modo conclui-se que uma considerável parte da população brasileira enfrenta em seu cotidiano uma luta que vai, além de seus limites.

Atualmente, percebe-se a escassez de projetos voltados à acessibilidade, isto é, ao auxílio de pessoas com algum tipo de deficiência, como: o relógio inteligente em Braille que utiliza ímãs e um conjunto de pinos para criar caracteres em Braille; bengala sensorial, acessório que detecta o ambiente por meio de sensores que emitem sinais transformados em vibração; e o *bookreader*, dispositivo que identifica e converte o texto para a pronúncia, tem a mesma estética de uma impressora, possuindo objetivos que inspiraram este projeto. A maioria das poucas soluções encontradas acabam apresentando limitações no uso, seja por valor comercial, seja por limitações em virtude de licenças de uso e direitos autorais.

“As pessoas com deficiência visual necessitam conhecer estratégias, recursos e equipamentos que facilitem o aprendizado das atividades diárias e de leitura e escrita” (GASPARETTO et. al, 2012, página 03).

Em decorrência dessa realidade é necessário que haja o desenvolvimento de um dispositivo composto em parte de *hardware* e em parte de *software*, no qual ao final de todos os seus respectivos desenvolvimentos exista em mãos um tipo diferenciado de óculos, que será voltado às pessoas que possuem algum tipo de deficiência visual, o qual nesses óculos serão acopladas duas câmeras que terão por função capturar e registrar o campo de visão da pessoa, a imagem gerada após a captura desse campo de visão será enviada para a parte de *software* que será responsável por verificar se a imagem possui algum texto, se sim o *software* irá fazer um

processo de digitalização da mesma e retornar em áudio para a pessoa o texto contido na imagem.

O objetivo deste projeto é o desenvolvimento de um *software* de inteligência capaz de compreender e aprender sintaxes de palavras e de frases, de modo que seja viável para substituir o *tesseract* (*software*), ferramenta proprietária da *google* que na versão anterior era utilizada para adquirir os textos presentes em uma imagem. Como objetivos específicos podem ser identificados: aprender a linguagem de programação *Python*, fazer levantamento bibliográfico sobre técnicas de processamento digital de imagens (PDI), implementação de PDI por meio do *OpenCV*, estudar técnicas de inteligência artificial relacionadas a detecção e aprendizado de reconhecimento de detecção da morfologias tipográficas, implementar as técnicas de IA (Inteligência Artificial, explicado na seção 2.4) por meio de bibliotecas como *SkLearn* do *Python*, capaz de reconhecer caracteres, de modo a realizar OCR (reconhecimento óptico de caracteres, em inglês), estudar técnicas de IA voltadas ao reconhecimento de estruturas sintáticas e semânticas, de modo a melhorar a acurácia das palavras reconhecidas e validar o *software* produzido, a partir de comparativo com *software* existente.

2.0 - FUNDAMENTAÇÃO TEÓRICA

2.1 - Processamento Digital de Imagens

Processamento digital de imagens(PDI) é o nome dado às técnicas voltadas principalmente para a análise de dados multidimensionais obtidos através de sensores, ou seja, é definido como a manipulação de imagens por um computador de modo que a entrada e saída sejam imagens, independente do(s) processo(s) realizado(s). Geralmente possui variados tipos de uso como o melhoramento do aspecto visual de certos traços estruturais para a análise humana e o fornecimento de outras informações para sua interpretação, além da geração de produtos que possam sofrer outros tipos de processamento. Abrange várias áreas do conhecimento como a análise de recursos naturais, a meteorologia por meio de imagens de satélite, transmissão digital de sinais de

televisão ou fac-símile, a análise de imagens biomédicas, a análise de imagens metalográficas e de fibras vegetais, a obtenção de imagens médicas por ultra-som, a radiação nuclear ou técnicas de tomografia computadorizada, as aplicações em automação industrial envolvendo o uso de sensores visuais em robôs, entre outras.

2.1.1 - Imagem Digital

A representação visual de um objeto é chamada de imagem, desse modo existem vários tipos de realizar esta representação como a pintura, a fotografia ou o vídeo. Mesmo que atualmente a palavra “imagem digital” tenha seu conceito utilizado no âmbito da tecnologia referenciando a representação de informações de modo binário. Com isso em mente podemos entender uma imagem digital como uma representação geralmente bidimensional construída a partir de uma matriz binária (composta de uns e zeros), sendo possível obtê-las de várias maneiras. Uma das maneiras é a utilização de um dispositivo que aplica a conversão analógica-digital, como no caso de uma câmara fotográfica digital ou um scanner. Com uma câmera desse tipo é possível tirar uma foto e armazená-la em formato digital, seja em um computador ou em algum outro tipo de suporte/dispositivo. No caso do *scanner* é possível registrar ou capturar uma imagem física e transformá-la em uma imagem digital. Outra maneira é a de criar imagens digitais através de um *software* com um programa básico como o *Microsoft Paint*, onde é possível fazer um desenho e criar uma imagem digital. Além disso, de acordo com a sua resolução uma imagem digital pode ser classificada de diferentes formas, assim neste sentido podemos nos referir ou a imagens matriciais ou a imagens vetoriais, ficando a critério do autor o tipo de formato da imagem digital.

“Uma imagem pode ser definida como uma função bidimensional $f(x,y)$, onde x e y são coordenadas espaciais(planas) e a amplitude de f em qualquer par de coordenadas é chamado de intensidade ou nível de cinza de uma imagem naquele ponto. Quando x e y e os valores

de amplitude de f são todos finitos, em quantidades discretas, chamamos a imagem de imagem digital”

-GONZALEZ & WOODS, 2000

2.1.2 Filtragem Espacial

A filtragem espacial refere-se ao plano da imagem, envolve a manipulação direta dos pixels da imagem pela aplicação de uma máscara espacial (*Kernels*, *Templates*, Janelas)², onde os valores das máscaras são chamados de coeficientes. O processo de filtragem é similar a uma operação matemática de convolução. Dentro do processo de filtragem cada pixel da imagem original é multiplicado pelo valor que reside no pixel correspondente da máscara, a soma desses resultados é definido como o novo valor do nível de cinza na nova imagem que será gerada. Por exemplo, se possuímos uma janela de $n \times n = k$ pixels definida por W , podemos calcular a somatória final em cada pixel da imagem pela Equação (1)³:

$$g(x,y) = \sum_{i=1}^k w_i f(x,y) \quad (1)$$

Onde $g(x,y)$ é o pixel da imagem resultante, k é o número de pixels da imagem original, w é a janela (máscara espacial que realizará a filtragem na imagem original), e $f(x,y)$ é o pixel da imagem original. Resumindo, esta é a fórmula para se calcular a somatória dos tons de cinza que serão aplicados no pixel da imagem resultante, mas se considerarmos o valor de ton de cinza de cada pixel da vizinhança (a,b,c,d,e,f,g,h,i) de $f(x,y)$, podemos definir a Equação (1) por:

² <http://www.facom.ufu.br/~backes/gsi058/Aula06-FiltragemEspacial.pdf>

³ <http://www.facom.ufu.br/~backes/gsi058/Aula06-FiltragemEspacial.pdf>

$$g(x,y) = w_1.a + w_2.b + w_3.c + w_4.d + w_5.e + w_6.f + w_7.g + w_8.h + w_9.i$$

Onde $g(x,y)$ é o pixel da imagem resultante, w_1, \dots, w_8 são o conjunto de pesos dos vizinhos ao pixel, definidos como a janela que realizarão a filtragem nos pixels da vizinhança, e a, \dots, i são os pixels da vizinhança.

2.1.3 - Filtragem de Média

Os filtros lineares passa-baixas, que também são conhecidos pelo nome de filtros de média são utilizados com o intuito de retirar informações muito discrepantes da imagem, caso existam, ou seja, ruídos da imagem digital, além de suavizar informações da imagem digital, de modo que após a aplicação do filtro as informações desejadas da imagem, fiquem mais homogêneas ou suavizadas⁴. Geralmente em filtrações passa-baixas são utilizadas as máscaras 3x3, mas nada impede a utilização de máscaras 5x5, 7x7, ou até maiores, salvo que quanto maior a máscara maior será a perda de nitidez e contraste na imagem⁵. No momento da aplicação da máscara nos pixels da imagem digital, a soma dos pesos da máscara tem que ser igual a 1, se essa soma for diferente de 1, é necessário dividir cada peso da máscara pela soma de todos os pesos da máscara, tendo como resultado após esse processo uma imagem digital com elementos mais homogêneos, contudo tem-se a perda de nitidez e contraste na imagem, sendo este ponto proporcional ao tamanho da máscara⁶. A filtragem de média ou filtros passa-baixas podem ser aplicados também em cascata, onde quanto maior for o número de iterações, maior será a homogeneidade dos elementos desta imagem⁷.

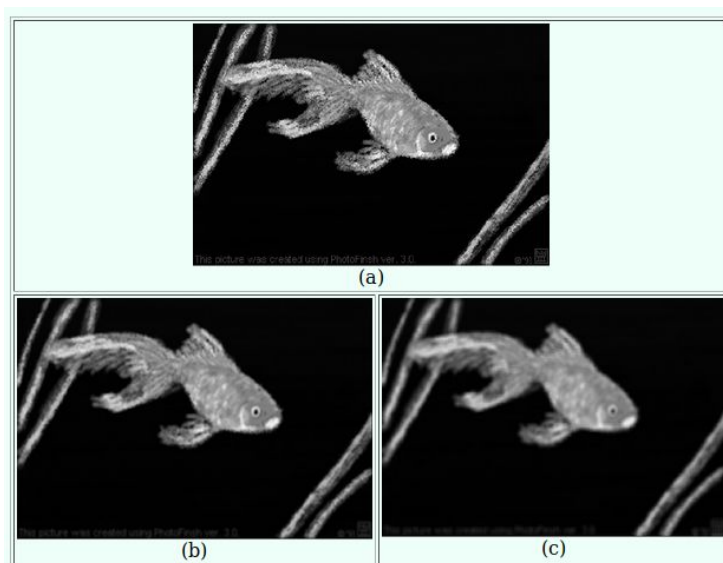
⁴ http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

⁵ http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

⁶ http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

⁷ http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

Figura 02 - Representação de uma filtragem de média aplicada à imagem (a), resultando nas imagens (b) e (c) com máscaras 3x3 e 5x5 respectivamente.



Fonte: INPE⁸

2.1.4 - Convolução

A convolução é um processo que atua no domínio espacial, e é realizada através de uma máscara que ao passar pela imagem original gerará uma nova imagem, isso se dá através da somatória dos pixels vizinhos ao pixel central, ponderados pelos seus respectivos pesos. Em termos de comparação a convolução e a correlação são baseadas no mesmo processo, sendo que a diferença entre elas é que a correlação atua no domínio da frequência, ou seja, com os níveis digitais dos pixels da imagem e a convolução atua no domínio espacial, ou seja, com níveis de tons de cinza, além de que diferente de um processo de correlação, a convolução possui o seu filtro com uma rotação de 180°⁹.

2.1.5 - Segmentação

É uma técnica de PDI para extração de elementos da imagem que consiste basicamente em analisar um agrupamento de pixels que possuem

⁸ http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm.

⁹ <http://www.facom.ufu.br/~backes/gsi058/Aula06-FiltragemEspacial.pdf>

características similares, para se realizar este processo é necessário dividir a imagem em regiões que seus pixels possuam a condição de similaridade citada acima. Um bom resultado para uma segmentação se daria apenas pela demarcação dos limites espaciais contidos nos objetos visíveis da imagem ou suas partes em componentes¹⁰. Vale ressaltar que geralmente a segmentação é uma forma simples no processo de reconhecer padrões, devido a sua característica de poder simplificar um devido problema, isolando regiões da imagem para uma segunda etapa de análise¹¹. Os algoritmos de segmentação podem ser classificados como globais ou locais, onde os métodos de segmentação global trabalham com a variação do nível digital da imagem, tentando assim formar grupos que possuam alguma similaridade, partindo da hipótese de que os objetos na imagem aparecem uniformemente em termos de cor, e quando nos referimos a métodos de segmentação locais, estamos falando do mesmo processo definido em uma região em específico da imagem¹².

2.1.6 - Sobel

O filtro não linear de sobel ou operador de sobel, tem por objetivo realçar linhas horizontais e verticais que estejam mais escuras que o fundo, sem a alteração de pontos isolados. A sua aplicação se dá pelo uso de duas máscaras, uma das máscaras servirá para detectar variações no sentido horizontal e a outra máscara no sentido vertical. O resultado obtido após a aplicação deste filtro é dado a seguir na Equação (2).

$$a' = \sqrt{a^2 + b^2} \quad (2)$$

¹⁰ <https://docs.ufpr.br/~centeno/uni/pdi/pdf/aulapdi05.pdf>

¹¹ <https://docs.ufpr.br/~centeno/uni/pdi/pdf/aulapdi05.pdf>

¹² <https://docs.ufpr.br/~centeno/uni/pdi/pdf/aulapdi05.pdf>

Onde a' é o valor de nível de cinza, correspondente à localização do pixel central da máscara, e (a) e (b) são referentes as duas máscaras que são necessárias na aplicação deste filtro.

2.1.7 - Canny

O filtro de *canny* é reconhecido como um dos mais eficientes quando o assunto é encontrar discontinuidades dentro de uma imagem, o seu resultado é uma imagem binária contendo os pontos da borda provindos da imagem original, sendo que para isso ele busca bordas situadas em locais máximos do gradiente de uma imagem¹³. O filtro de *canny* faz uma convolução na imagem com o filtro gaussiano para o cálculo da magnitude e do ângulo do gradiente, em seguida ele faz a afinação das cristas largas dos gradientes, e para esses processos este filtro classifica os pontos relacionados às orientações horizontal e vertical, com $+45^\circ$ e -45° (possuindo assim um intervalo de mais ou menos $22,5^\circ$), onde para os vizinhos em determinada orientação deve-se verificar os seus gradientes, desse modo para suprimir os gradientes não máximos ele aplica a condição de que quando o valor de magnitude de $M(x,y)$ é inferior a pelo menos um de seus vizinhos, deve-se fazer o gradiente de pixel ($gn(x,y)$) receber como novo valor o 0, e caso contrário deve-se fazer o gradiente do pixel ($gn(x,y)$) receber como novo valor a $M(x,y)$, onde $gn(x,y)$ é o gradiente com supressão¹⁴. No filtro de *canny* a Limiarização com histerese é utilizada na quebra do contorno (borda tracejada), sendo que a limiarização é um processo de segmentação de imagens baseado na diferença do nível de cinza dos objetos da imagem e a partir de um limiar estabelecido de acordo com as características dos objetos que se quer isolar, o processo citado acima é

¹³

https://agostinhobritojr.github.io/tutorial/pdi/#_detec%C3%A7%C3%A3o_de_bordas_com_o_algoritmo_de_canny

¹⁴

https://agostinhobritojr.github.io/tutorial/pdi/#_detec%C3%A7%C3%A3o_de_bordas_com_o_algoritmo_de_canny

executado¹⁵. Desse modo uma imagem limiarizada pode ser definida como na Equação (3).

$$g(x, y) = \begin{cases} 1 & \text{se } f(x, y) > T \\ 0 & \text{se } f(x, y) \leq T \end{cases} \quad (3)$$

No caso do filtro de canny, são utilizados dois limiares específicos sendo, $T_1 > T_2$ e $T_2.T_1 > T_2$, além de que se o gradiente $gn(x,y)$ do pixel for maior que T_1 , é assumido assim como um ponto forte de borda, para os pixels restantes em que $gn(x,y)$ é maior que T_2 são assumidos como pontos fracos de borda, e para todos os pontos fracos de borda deve-se procurar em seus 8 vizinhos se há algum ponto de borda forte, caso encontre, este é marcado como parte da fronteira¹⁶.

Figura 03 - Exemplo da aplicação do filtro de *canny*.



Fonte: Github¹⁷

2.1.8 - Morfologia Matemática

Segundo (Catarina,2019), em biologia, morfologia refere-se ao estudo das plantas, de sua estrutura e de sua topologia, logo quando fala-se de

¹⁵

https://agostinhobritojr.github.io/tutorial/pdi/#_detec%C3%A7%C3%A3o_de_bordas_com_o_algoritmo_de_canny

¹⁶

https://agostinhobritojr.github.io/tutorial/pdi/#_detec%C3%A7%C3%A3o_de_bordas_com_o_algoritmo_de_canny

¹⁷

https://agostinhobritojr.github.io/tutorial/pdi/#_detecção_de_bordas_com_o_algoritmo_de_canny

morfologia matemática, refere-se a estrutura geométrica das entidades presentes em uma imagem podendo realizar diversos processos na imagem, desde erosão, dilatação, abertura, fechamento, esqueletização, etc. Também tem como base matemática a teoria dos conjuntos.

2.1.9 - Erosão

Lida com imagens binárias, em preto e branco, caso a imagem seja colorida ou em tons de cinza é necessário sua conversão para binário. Pode ser representada na forma de Equação (4).

$$A \ominus B = \{x \mid (B)x \subseteq A\} \quad (4)$$

Onde **A** corresponde a imagem a ser erodida;

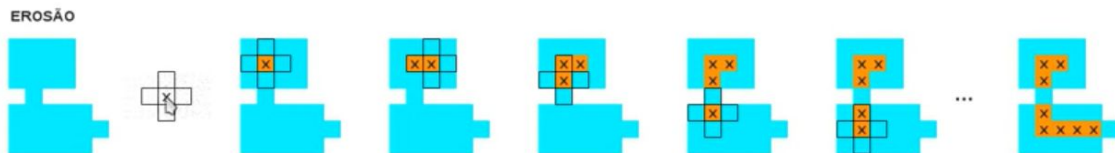
Onde \ominus representa a operação de erosão ;

Onde **B** representa o elemento estruturante;

Onde **x** representa a imagem resultante ;

A erosão “diminui” a imagem através da operações entre conjuntos, um dos conjuntos é uma imagem enquanto outro conjunto é o elemento estruturante, ao definir um pixel principal do elemento estruturante e uma separação do objeto com o fundo da imagem, o elemento estruturante passa por cima do objeto em todo local que ele encaixar, marcando seu principal pixel definido, no final, a imagem resultante é aquela onde todos os pixels principais foram marcados. A figura 06, representa um processo de erosão, os pontos em laranja representam o que seria a imagem final após o processo.

Figura 04 - Exemplo de um processo da erosão.



Fonte : Youtube¹⁸

2.1.10 - Dilatação

A dilatação “aumenta” a imagem através da operações entre conjuntos, um dos conjuntos é uma imagem enquanto outro conjunto é o elemento estruturante, ao definir um pixel principal do elemento estruturante e uma separação do objeto com o fundo da imagem. O processo pode ser representado no formato de Equação (5).

$$A \oplus B = \{x \mid (\)x \cap A \neq \varnothing\} \quad (5)$$

$$A \oplus B = \{x \mid [(\)x \cap A] \subseteq A\}$$

Onde **A** corresponde a imagem a ser dilatada;

Onde \oplus representa a operação de dilatação ;

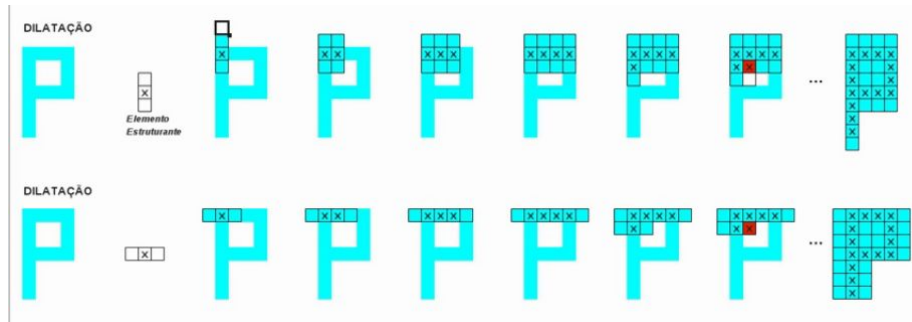
Onde **B** representa o elemento estruturante;

Onde **x** representa a imagem resultante ;

O elemento estruturante passa por cima do objeto em todo local que seu principal pixel encaixa, marcando o elemento estruturante por fora do objeto aumentando a imagem .

¹⁸ https://www.youtube.com/watch?v=E8cqrkK4L_M.

Figura 05 - Exemplo de um processo da dilatação, os pixels com x representam o objeto original e os azuis o objeto final após a dilatação.



Fonte : Youtube¹⁹

2.1.11 - Abertura

O processo da abertura suaviza ruídos e elimina contornos, nota-se que ela é resultado da dois processos seguidos, um de erosão da imagem A pelo elemento estruturante B em seguida de um processo de dilatação da imagem A pelo elemento estruturante B , pode ser definido pela Equação (6).

$$A \circ B = (A \ominus B) \oplus B \quad (6)$$

Onde **A** corresponde a imagem a ser dilatada;

Onde \circ representa a operação de fechamento ;

Onde **B** representa o elemento estruturante;

Onde \oplus representa o operador de dilatação;

Onde \ominus representa o operador de erosão;

¹⁹ https://www.youtube.com/watch?v=E8cqrkK4L_M.

Figura 06 - Exemplo de abertura em uma imagem



Fonte : UFPR²⁰

2.1.12 - Fechamento

Através desse processo é possível “ fechar buracos ”, nota - se que o processo é definido a partir de outros dois processos seguidos também, primeiro um da dilatação de imagem A pelo elemento estruturante B em seguida de um processo de erosão da imagem A pelo elemento estruturante B, o processo pode ser representado no formato de Equação (7).

$$A \bullet B = (A \oplus B) \ominus B \quad (7)$$

Onde **A** corresponde a imagem a ser dilatada;

Onde **•** representa a operação de fechamento ;

Onde **B** representa o elemento estruturante;

Onde \oplus representa o operador de dilatação;

Onde \ominus representa o operador de erosão;

²⁰ <http://www.inf.ufpr.br/lesoliveira/download/morfologia.pdf>.

Figura 07 - Exemplo de fechamento em uma imagem



Fonte: UFPR²¹

2.2 - Correlação

O processo de correlação atua no domínio da frequência e é realizado a partir de uma máscara aplicada sobre uma imagem digital, onde o pixel central da imagem relativo à posição da máscara receberá como novo valor digital, a somatória dos valores digitais dos pixels vizinhos, ponderados pelos respectivos pesos da máscara²². Considerando-se uma máscara 3x3 aplicada sobre um pixel qualquer, é possível calcular matematicamente a correlação da mesma sobre o mesmo como na Equação (8).

$$\begin{aligned}
 p[i,j] = & a * p[i-1,j-1] + b * p[i-1, j] + c * p[i-1,j+1] + \\
 & d * p[i ,j-1] + e * p[i , j] + f * p[i ,j+1] + \\
 & g * p[i+1,j-1] + h * p[i+1, j] + i * p[i+1,j+1]
 \end{aligned}
 \tag{8}$$

²¹ <http://www.inf.ufpr.br/lesoliveira/download/morfologia.pdf>.

²² http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

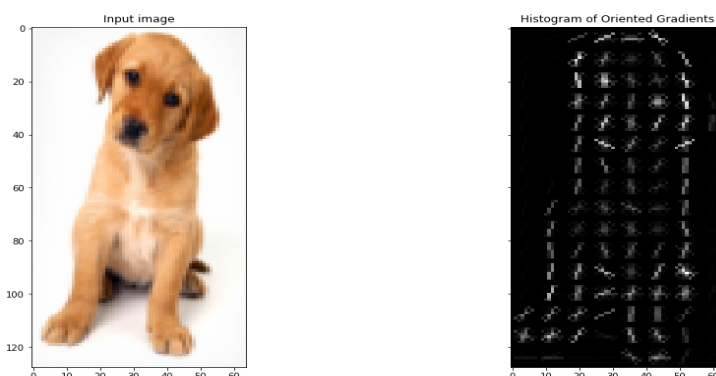
Onde $p[i,j]$ é o pixel em questão, p é o peso correspondente a cada pixel da máscara, e $a,...,i$ são os pixels vizinhos, salvo que ao aumentar a máscara, consequentemente aumentaram o número de pixels vizinhos.

2.3 - Histograma de Gradientes Orientados

HOG(*Histogram Oriented of Gradient*) é uma técnica utilizada para extração da características de uma imagem, tais características podem ser cor, textura, formato,etc. Elas podem ser medidas e mensuradas. Essas características são agrupadas em um vetor, que é chamado de descritor. Segundo (SANTOS et. al, 2012) “O algoritmo se baseia na ideia de que a forma e a aparência de um objeto podem ser descritas muitas vezes pela intensidade dos gradientes ou a direção das bordas, sem um conhecimento prévio da posição de tais bordas.”

Após este processo é feita a conversão da imagem para escala de cinza, computação dos gradientes, normalização e acúmulo do histograma e enfim é coletada uma janela da detecção sobre os histogramas , que consiste no descritor HOG e pode ser utilizado como entrada para algoritmos Classificadores.

Figura 08 - Exemplo de aplicação da HOG em uma imagem



Fonte: ANALYTICS VIDHYA²³

2.4 - Inteligência Artificial

Segundo (SILVA, Marco), Inteligência Artificial é uma área da ciência da computação que simula o raciocínio humano para resolução de problemas e performance de tarefas como seres humanos.

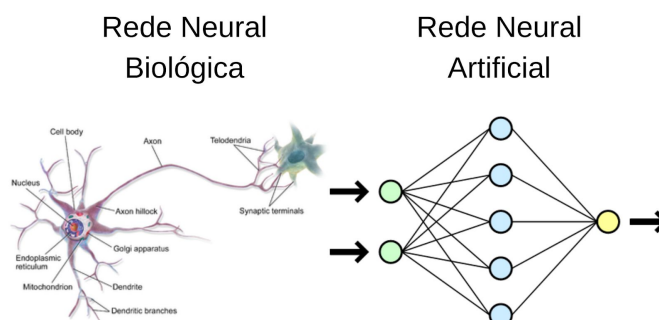
Algoritmos de Inteligência Artificial são utilizados frequentemente e estão em crescente para resolução de problemas e automatização de tarefas, sobretudo na área de Redes Neurais Artificiais e *deep Learning*.

2.4.1 - Redes Neurais Artificiais

Segundo (DUTRA, 2011) “Atualmente, as Redes Neurais vêm sendo umas das melhores opções dentre as possíveis ferramentas de reconhecimento de padrões”, desta forma Redes Neurais Artificiais podem atuar em diversas situações e formas, são adaptáveis ao meio e reconhecem e segmentam padrões tendo decisões autônomas que se inspiram na capacidade humana cerebral.

Redes Neurais Artificiais (RNA) são modelos computacionais inspirados nas redes neurais Biológicas, a seguir na primeira imagem temos uma representação de Rede Neural Biológica e na segunda temos uma representação de Rede Neural Artificial.

Figura 09 - Comparação Rede Neural Biológica e Rede Neural Artificial



Fonte: Wikipédia²⁴²⁵

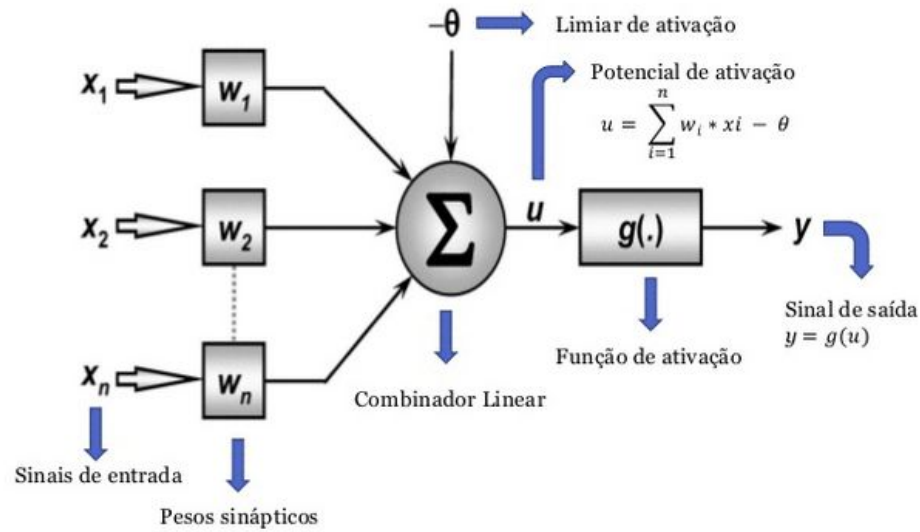
2.4.2 - Rede Perceptron de Única Camada

Uma Rede Perceptron de Camada Única é caracterizada por ser uma RNA simples, possuindo apenas um único neurônio, pode ter várias entradas, mas apenas uma única saída, pode ser utilizada para resolução de problemas simples, como por exemplo a resolução do operador lógico de conjunção (&). As entradas serão posições de um vetor $x[]$ e irão receber os dados do ambiente, em seguida são multiplicadas pelos pesos sinápticos e então somadas no neurônio, este resultado é passado como argumento para uma função de ativação, que em uma Rede de Camada única, pode ser degrau (valores entre 0 e 1) ou degrau bipolar (valores entre -1 e 1), embora não sejam as únicas existentes, são as que se encaixam melhor em casos e testes com perceptron pelo fato de conseguirem separar bem duas classes.

²⁴ https://pt.wikipedia.org/wiki/Redes_neurais_biol%C3%B3gicas

²⁵ https://pt.wikipedia.org/wiki/Rede_neural_artificial

Figura 10 - Modelo de uma Rede com Única Camada w



Fonte : Embarcados²⁶

Matematicamente podemos representar esse processo nos formatos das Equações (9) e (10).

$$y = g(u) \quad (9)$$

$$u = \sum_{i=1}^n w_i \cdot X_i - \theta \quad (10)$$

Onde:

- X_i - Entradas - Valor real ou binário
- W_i - Pesos Sinápticos - Valor Real aleatório
- θ - Limiar de Ativação - Valor Real aleatório

²⁶ <https://www.embarcados.com.br/rede-perceptron-de-uma-unica-camada/>

- y - Saída - Valor Binário
- $g(.)$ - Função de Ativação - Degrau ou Degrau Bipolar

3.0 - METODOLOGIA

Para o desenvolvimento do projeto, é utilizada a linguagem de programação *Python*. A linguagem possui entusiastas pelo mundo inteiro e comporta um nicho específico de desenvolvedores na área da visão computacional, embora não seja a única utilizada, também com diversas ferramentas robustas para essa finalidade e bibliotecas como por exemplo o *OpenCV* (Biblioteca utilizada para lidar com assuntos referentes à visão computacional, possui diversas funções que facilitam operações mais complexas), *SkLearn* (Biblioteca utilizada para tratar de assuntos referentes à Inteligência Artificial e aprendizado da máquina) e *NumPy* (Biblioteca com finalidade matemática para cálculos de *Arrays* multi-dimensionais, além de poder ser utilizada em tarefas relacionadas a PDI e aprendizado da máquina), tornando-a uma escolha atrativa para o desenvolvimento do projeto .

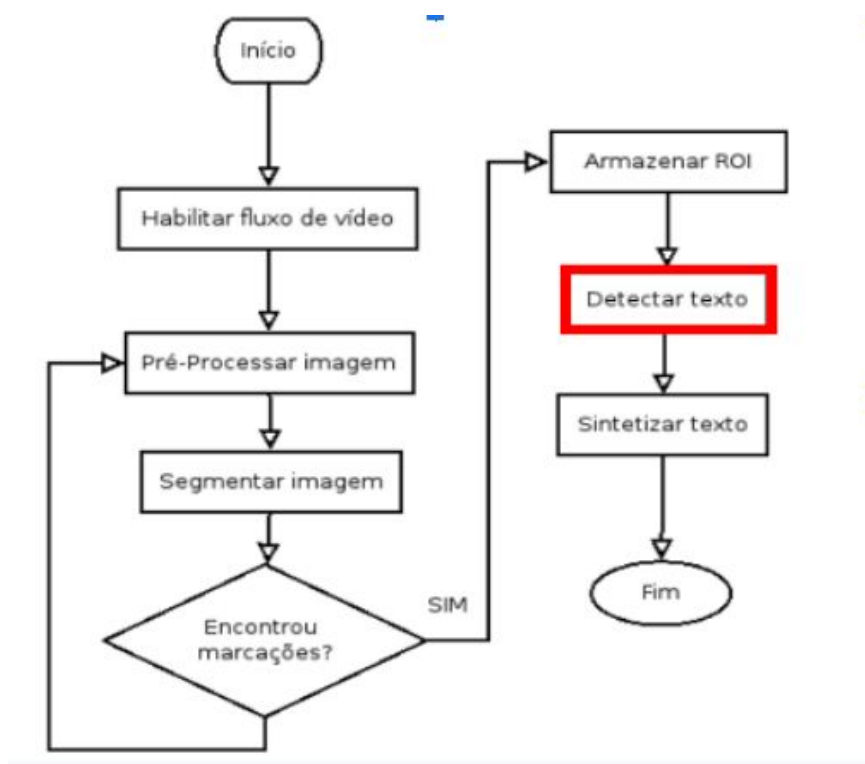
Para a construção de uma ferramenta equivalente ao *tesseract* propõe-se uma Rede Neural Artificial, toda RNA utiliza um *Dataset* (Conjuntos de dados) que servem para o aprendizado e treino, os tipos de dados que um *Dataset* contém varia de acordo com a necessidade, podendo ser um arquivo com vários números organizados ou ainda várias pastas contendo um conjunto de imagens, inicialmente houve o desenvolvimento de um *Dataset*, mas devido ao tempo necessário e a quantidade de dados necessários para um treinamento adequado, o qual foi baixado do departamento de engenharia elétrica e eletrônica do Reino Unido²⁷, o *DataSet* baixado contém sessenta e duas pastas com diferentes tipos de imagens de caracteres em várias resoluções, tamanhos e fontes diferentes.

Para isso o projeto utiliza o método de HOG para geração de um vetor de características de entrada para a Rede Neural Artificial, a RNA classifica os

²⁷ <https://www.surrey.ac.uk/departament-electrical-electronic-engineering>

dados e gera saídas (representando qualquer uma das vinte e seis letras do alfabeto (maiúsculas e minúsculas) e os dez números árabicos). O classificador escolhido foi a classe ²⁸*MPLClassifier* da biblioteca *Sci-Kit Learning* que implementa uma Rede Neural MLP (*Multi-Layer Perceptron*) a partir de um dataset e gera n saídas. O algoritmo que a biblioteca implementa utiliza de *back-propagation*. O treinamento necessita de uma matriz de $n_{amostras} \times n_{características}$ para manter as amostras de treinamento e também de um vetor para definição das classes. Após o treinamento, o algoritmo consegue prever possíveis saídas para determinadas entradas.

Figura 11 - Fluxograma do funcionamento do Software



Fonte: Autoria própria

Atualmente, o projeto emprega a ferramenta da *Google*, o *Tesseract*. Esta ferramenta consegue detectar letras via OCR(*Optical Character Recognition*), mas o uso de ferramenta é um complicador devido a sua licença

²⁸ https://scikit-learn.org/stable/modules/neural_networks_supervised.html

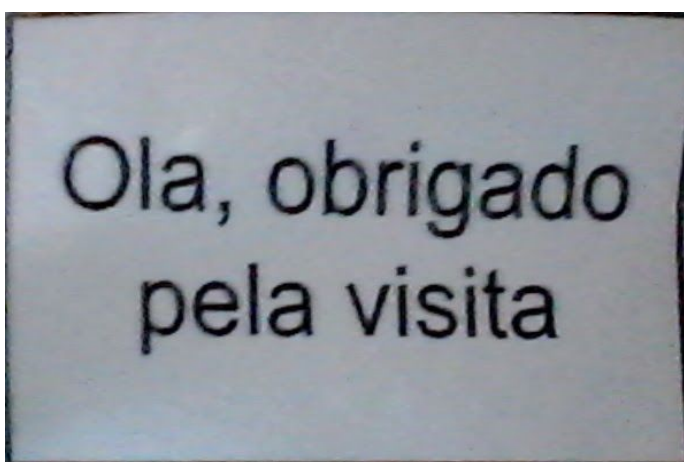
fechada. Assim o desenvolvimento o projeto consistiu em desenvolver uma ferramenta com funcionalidade equivalente.

4.0 - RESULTADOS

Os resultados obtidos na etapa anterior do projeto são promissores, foi possível ter pronto ao final do projeto um software baseado em Processamento Digital de Imagens, capaz de processar a imagem e extrair os caracteres. Em seguida, através do *Tesseract*, foi possível identificar e reconhecer o texto contido em uma imagem, além de sintetizar o áudio (reproduzi-lo) correspondente ao texto reconhecido através da ferramenta aberta *Espeak*. O maior complicador atual para finalizar o software é a licença apache do software OCR *Tesseract*. Assim, é necessário ainda desenvolver uma aplicação com funcionalidade equivalente.

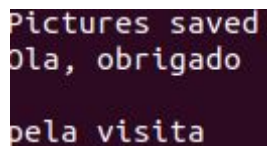
O software desenvolvido já consegue reconhecer um texto dentro de uma região de interesse (ROI) retangular e reproduzi-lo por meio de ferramenta aberta E- speak :

Figura 12 - Detecção da ROI



Fonte: Autoria própria

Figura 13 - Texto detectado no terminal do Linux

A screenshot of a Linux terminal window with a dark background. The text displayed is the result of an OCR process. The first line is 'Pictures saved' in a light blue font. The second line is 'ola, obrigado' in a light green font. The third line is 'pela visita' in a light red font.

```
Pictures saved  
ola, obrigado  
pela visita
```

Fonte: Autoria própria

No limiar do desenvolvimento esperou-se ao final deste projeto um software de reconhecimento, empregando uma rede neural artificial capaz de identificar caracteres além de extraí-los, de modo a substituir o tesseract, porém devido a complexidade e tempo disponível para desenvolvimento não foi possível concluí-lo, todavia a aplicação da rede encontra-se muito próxima.

Foi possível realizar a extração das características HOG das imagens do dataSet e armazená-la dentro de uma lista, também foi possível carregar todas as imagens dentro de uma lista de listas e mostrá-las, a extração poderia ser realizada fazendo o passo a passo e implementando um método próprio, porém escolheu-se o método *hog* da biblioteca *Scikit-Image*, pois ele consegue extrair uma grande quantidade de características, gerando assim um futuro treinamento mais eficiente. A foto do código desenvolvido pode ser visualizada no Apêndice A.

Figura 15 - Valores de Hog

```
0.27709894, 0.11580069, 0.11656013, 0.08802565, 0.03101531,
0.23816256, 0.37265684, 0.37265684, 0.13334906,
0.14277449, 0.09908079, 0.06011581, 0.01726463, 0.06047446,
0.01932442, 0.03110552, 0.23943972, 0.11721418, 0.06833109,
0.19168373, 0.15478921, 0.06533032, 0.08726904, 0.07618187,
0.07182962, 0.37265684, 0.27794198, 0.11207935, 0.11281439,
0.08519688, 0.03001861, 0.01707261, 0.22024562,
0.37265684, 0.10568673, 0.01698334, 0.00822171,
0.02403623, 0.29262281, 0.29262281, 0.1076518,
0.10640286, 0.24230378, 0.06750945, 0.1243981, 0.20059094,
0.20630645, 0.09469881, 0.0892887, 0.29262281, 0.29262281,
0.13932162, 0.14023532, 0.10590503, 0.037315, 0.07343263,
0.29262281, 0.09387866, 0.12426494, 0.1096123, 0.07343263,
0.03346679, 0.08282856, 0.0859632, 0.13818855, 0.19002412,
0.24889833, 0.29262281, 0.17772508, 0.14874816, 0.04605489,
0.05359185, 0.00270054, 0.01588858, 0.08772996, 0.08271798,
0.41012293, 0.32007405, 0.12906899, 0.12991546, 0.09811152,
0.034569, 0.01966058, 0.25363174, 0.41012293,
0.12170734, 0.01955777, 0.009468, 0.02767978,
0.17604031, 0.23058199, 0.41012293, 0.16464636,
0.13780184, 0.04266572, 0.04964804, 0.00250181, 0.01471934,
0.12750579, 0.19191685, 0.2028973, 0.09957537, 0.04554413,
0.0402602, 0.0384973, 0.07669328, 0.05737043]]]
```

Fonte: Autoria Própria

5.0 - CONCLUSÃO

O desenvolvimento de um software voltado à acessibilidade de pessoas com alguma dificuldade de leitura representa uma enorme possibilidade de inclusão, principalmente se ao final do desenvolvimento de todas as respectivas etapas for possível a concepção de um produto a custo reduzido. Como trabalhos futuros destacam - se o desenvolvimento de um aplicativo móvel para portar o software, utilizando *Android Studio* e também o desenvolvimento da própria armação de óculos através de uma impressora 3D ou de uma CNC, as câmeras a serem acopladas seriam compradas a partir de verba destinada para o projeto ou arrecadação própria. Também será necessário portar o software para uma biblioteca, de modo a integrá-lo com o software desenvolvido na etapa anterior do projeto e melhorar a parte já desenvolvida para realizar identificação de caracteres sem a necessidade de uma ROI.

6.0 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] BANON, BARRERA. “**Bases da Morfologia Matemática para Análise de Imagens Binárias**”. INPE. Recife. 1994. pg 99-128.

[2] BRASIL. Constituição (1988). “**Constituição da República Federativa do Brasil**”. Brasília, DF: Senado Federal: Centro Gráfico, 1988.

[3] CATARINA, Adair. “**Morfologia Matemática**”. 2019. Disponível em: <http://www.inf.unioeste.br/~adair/PID/Notas%20Aula/Morfologia%20Matematica.pdf>

Acesso em : 24/10/19

[4] DUTRA , VAGNER . “**Redes Neurais e o reconhecimento de padrões da texto**”. Itatiba: 2011.

[5] FLECK et. al. “**Redes Neurais Artificiais: princípios básicos**”. Cascavel: Inovação,2016.

[6] FELGUEIRAS, Carlos. “**Processamento Digital de Imagens: Filtragens espaciais**”. 2007. Disponível em : http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm

Acesso em : 24/10/19

[7] GASPARETTO et. al. **Utilização de Recursos de Tecnologia Assistiva por Escolares com Deficiência Visual** . Porto Alegre: 2012.

[8] GONZALEZ, RAFAEL C.; WOODS, RICHARD E. “**Digital Image Processing**”. 2.ed. Prentice Hall.2002

[9] OLIVEIRA, LUIZ . “**Processamento de Imagens: Morfologia Matemática Binária**”. 2016. Disponível em : <http://www.inf.ufpr.br/lesoliveira/download/morfologia.pdf>

Acesso em : 25/10/19

[10] PALMIERI, SERGIO . “**Rede Perceptron de uma única camada**”.2016. Disponível em: <https://www.embarcados.com.br/rede-perceptron-de-uma-unica-camada/>

Acesso em : 30/10/19

[11] PISTORI, HEMERSON. “**Morfologia Matemática com Exemplos**”. 2012. Disponível em : https://www.youtube.com/watch?v=E8cqrkK4L_M

Acesso em : 24/10/19

[12] SANTOS, et. al . “**Histograma de Gradientes Orientados na Detecção de Motocicletas**”. Teresina - PI: 2012.

[13] SANTOS, Marco Aurélio da Silva. "**Inteligência Artificial**"; *Brasil Escola*.

Disponível em:

<https://brasilecola.uol.com.br/informatica/inteligencia-artificial.htm>.

Acesso: 17/12/19.

[14] TURING, A. M. "**Computing Machinery and Intelligence**". Parsing the Turing Test. Springer, Dordrecht, 2009. 23-65.

[15] VILLANUEVA, Juan. "**Redes Neurais Artificiais(Definições)**". Paraíba. 2016.

7.0 - APÊNDICES

APÊNDICE A - FOTO DO CÓDIGO DESENVOLVIDO

```

1  import cv2
2  import os
3  from skimage.feature import hog
4  from skimage.io import imread, imshow
5  from skimage.transform import resize
6  import matplotlib.pyplot as plt
7
8  def main():
9      CAMINH001 = 'dataSet/Bmp/Sample00'
10     CAMINH002 = 'dataSet/Bmp/Sample0'
11     path = []
12     dataSet = []
13     train_vector = []
14     hogFeatures = []
15     test_vector = []
16     train = []
17     test = []
18     dim = (28,64)
19     for n in range(1,63):
20         if(n < 10):
21             train_ids = next(os.walk('dataSet/Bmp/Sample00'+str(n)))[2]
22             lenTrain = int(len(train_ids) * 0.9)
23             train.append(lenTrain)
24             lenTest = int(len(train_ids) * 0.1)
25             test.append(lenTest)
26             for m in train_ids:
27                 way = CAMINH001 + str(n)+ '/' + m
28                 path.append(way)
29
30         if(n >=10):
31             train_ids = next(os.walk('dataSet/Bmp/Sample0'+str(n)))[2]
32             lenTrain = int(len(train_ids) * 0.9)
33             train.append(lenTrain)
34             lenTest = int(len(train_ids) * 0.1)
35             test.append(lenTest)
36             for m in train_ids:
37                 way = CAMINH002 + str(n)+ '/' + m
38                 path.append(way)
39
40     cont = 0
41     for n in path:
42         img = cv2.imread(n,0)
43         resized_img = cv2.resize(img,dim)
44         fd = hog(resized_img, orientations=9, pixels_per_cell=(8, 8),cells_per_block=(2, 2), multichannel=False)
45         hogFeatures.append(fd)
46         dataSet.append(resized_img)
47
48     print(hogFeatures)
49     '''
50     for n in dataSet:
51         cv2.imshow('',n)
52         cv2.waitKey(0)
53         cv2.destroyAllWindows()
54     '''
55
56 if __name__ == "__main__":
57     main()

```

Fonte: Autoria Própria