UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Lucas Vinhal Pereira

Uma abordagem automática para leitura óptica de cartões: um estudo de caso

UBERLÂNDIA

2020

Lucas Vinhal Pereira

Uma abordagem automática para leitura óptica de cartões: um estudo de caso

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Universidade Federal de Uberlândia

Orientador: Prof. Dr. Mauricio Cunha Escarpinati

UBERLÂNDIA

2020

Lucas Vinhal Pereira

Uma abordagem automática para leitura óptica de cartões: um estudo de caso

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Dr. Mauricio Cunha Escarpinati
Orientador

Prof. Dr. Fabiano Azevedo Dorça
Membro da Banca

Prof. Dr. Ronaldo Castro de Oliveira
Membro da Banca

UBERLÂNDIA

2020

Dedico este trabalho à minha família,
à minha namorada e aos meus amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus por estar sempre me abençoando e guiando meus passos.

Agradeço ao meu orientador Professor Doutor Mauricio Cunha Escarpinati, pela sabedoria, apoio e acompanhamento nesta trajetória.

A Diretoria de Processos Seletivos da Universidade Federal de Uberlândia, pela cooperação.

Aos professores da Faculdade de Computação, por todos os ensinamentos passados e experiências compartilhadas.

Gostaria de deixar registrado também, o meu agradecimento e reconhecimento à minha mãe Lourdes, à minha irmã Letícia e à minha namorada Milenny, por estarem sempre ao meu lado e por toda a paciência e incentivo nos momentos mais difíceis.

Enfim, a todos os que por algum motivo contribuíram direta e indiretamente para a realização desse trabalho.

RESUMO

Este trabalho propõe o desenvolvimento de um *software* eficaz na leitura de cartões resposta de provas multiplo escolha. Nesse sentido, foi desenvolvido um sistema que, utilizando técnicas de visão computacional, foi capaz de reconhecer as alternativas assinaladas pelos candidatos bem como os dados de identificação do candidato em si. No que tange a metodologia empregada, foram utilizadas as técnicas de processamento de imagem, bem como a linguagem Python, com o auxilio da biblioteca OpenCV. Com isso, o presente trabalho foi divido em cinco etapas especificas: aquisição das imagens, segmentação, leitura/aquisição dos dados, exportação e comparação dos resultados. Para validação dos resultados obtidos, o sistema foi submetido à leitura dos cartões respostas da primeira etapa do vestibular da Universidade Federal de Uberlândia de 2018. Os resultados alcançados nos testes foram promissores com altas taxas de efetividade e acerto na leitura.

Palavras-chave: Reconhecimento Óptico. Processamento de Imagens. Linguagem Python. Sistemas Aplicados.

ABSTRACT

This work proposes the development of an efficient software for reading multiple choice test response cards. In this sense, the developed system, using computer vision techniques, was able to recognize the candidates indicated alternatives as well as the candidate identification data. Regarding the methodology employed, image processing techniques were used, as well as the Python language, with the help of the OpenCV library. Thus, the present work was divided into five specific steps: image acquisition, segmentation, data reading / acquisition, exportation and results comparison. To validate the obtained results, the system was submitted to the reading of the response cards of the first stage of the 2018 entrance exam of the Federal University of Uberlândia. The results obtained in the tests were promising with high effectiveness rates and correct reading.

Keywords: Optical Mark Recognition. Image Processing. Python Language. Entrance Exam Process.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFU	Universidade Federal de Uberlândia
OMR	Optical Mark Recognition 13
SOMR	Software Optical Mark Recognition 16
DPI	Dots Per Inch 16
PDI	Processamento Digital de Imagens 19
CHT	Circle Hough Transform 21
TDC	Tabela de Conteúdo 23
CSV	Comma Separated Value 30

LISTA DE EXCERTOS DE CÓDIGO-FONTE

SUMÁRIO

1 INTRODUÇÃO

O Reconhecimento Óptico de Marcas, ou o termo original em inglês Optical Mark Recognition (OMR) é um processo que digitaliza formulários em papel para detectar marcações em posições pré-determinadas. E teve sua primeira aparição em estudos como é o caso do IBM 805 Test Scoring Machine, criado em 1930, conseguia de ler marcas através do condução elétrica do lápis grafite (IBM. . . , 2006). Mas o primeiro *scanner* ótico seria criado com sucesso apenas em meados de 1960, por Everett Franklin Lindquist, por meio de um mecanismo de transporte de papel mimeográfico.

A popularização dos computadores pessoais nos anos 80 deram origem ao OMR por software, o qual não necessita de um *hardware* exclusivo e permite a qualquer pessoa com uma impressora a *laser* e um *scanner* a realizar a interpretação e a abstração dos dados por meio apenas de um software dedicado (BERGERON, 1998).

Assim, esse processo tem sido utilizado em diversas aplicações por quase 80 anos, aplicando-o para diversas necessidades, como para detecção de marcações em bilhetes de loterias ou para correção de gabaritos de provas.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Adentrando na temática desta pesquisa, este trabalho tem como objetivo desenvolver um *software* de OMR que reconheça as alternativas assinaladas em cartões respostas padronizados. O sistema deve ser capaz de identificar qual alternativa o candidato assinalou para cada uma das questões existentes na folha de resposta. Também deverá identificar o candidato através da leitura das informações contidas em códigos de barras existentes nas folhas de respostas.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Criação de um software OMR fazendo uso somente de bibliotecas e tecnologias de código aberto, com desempenho e assertividade similar aos softwares comerciais de mesmo fim;
- Redução do tempo de processamento e detecção das alternativas assinaladas em testes múltipla escolha;
- Possível redução do custo final do processo de correção das provas.

1.2 ORGANIZAÇÃO DO TRABALHO

O decorrer deste trabalho é organizado em 5 seções: Introdução onde é realizada uma breve introdução a cerca do tema, bem como seu objetivo; o Capítulo 2 abrange conceitos básicos relacionados ao reconhecimento óptico de marcas, imagens digitais e processamento digital de imagens; no Capítulo 3 é detalhado o processo de desenvolvimento do *software*, as técnicas aplicadas no mesmo, além de apresentar os testes e comparações utilizadas para a validação do trabalho; no Capítulo 4 são exibidos os resultados obtidos e a discussão acerca dos mesmos; e, por fim, o Capítulo 5 contempla as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta sessão serão discutidos aspectos que fundamentam o presente trabalho, tais como: OMR, imagens digitais, processamento de imagens, entre outros.

2.1 RECONHECIMENTO ÓPTICO DE MARCAS

O reconhecimento óptico de marcas, em inglês *Optical Mark Recognition* (OMR) é o processo de detecção e captura de marcações humanas em questões de múltipla escolha utilizando-se *software*, *hardware*, ou ambos. Existem várias aplicações para essa tecnologia, em que se destaca sua utilização no processo da aquisição de dados de testes, formulários, folhas de ponto, votações, geocodificações, questionários, entre outros.

A recuperação dos dados em OMR pode ser executada utilizando-se *hardware* e *software* (isto é, há um dispositivo fisicos de escaneamento dedicado ao processo), ou somente *software*. Falaremos mais sobre essas formas abaixo:

2.1.1 Hardware OMR

Na Figura 1 é apresentado um exemplo de *scanner* OMR comercial oferecido pela empresa Scantron.

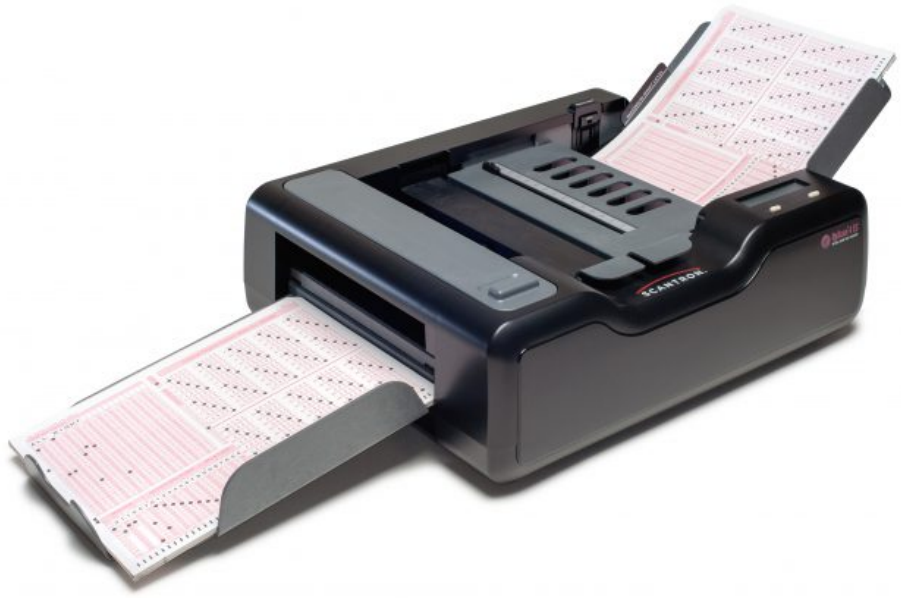


Figura 1 – Scanner OMR comercial OpScan® 4ES.

Um *hardware* OMR pode ser utilizado emparelhado com um *scanner* normal ou já integrado no interior do *scanner* específico. Esse processo é realizado com o auxílio da chamada

folhas de respostas ópticas ou folha OMR, que é uma folha com um formulário de múltipla escolha disposto em um alinhamento pré-determinado, a fim de poder ser detectado pelo *scanner* devido ao posicionamento das questões. Após a aquisição dos dados, as informações são convertidas em um arquivo de dados que pode ser enviado a um computador.

As folhas OMR quando escaneadas são lidas passando uma luz através da folha e usando fototubos ou fotodetectores (tubos a vácuo ou cheios de gás que são sensíveis à luz (CALVERT, 2002)) do outro lado, assim permitindo medir quanto da luz é refletida e quanto é bloqueada (BLOOMFIELD, 2006).

2.1.2 Software OMR

Software Optical Mark Recognition (SOMR) permite que, com uma impressora a laser e um *scanner* de imagens simples, possa-se projetar e imprimir formulários, adequando-os da melhor forma a suas aplicações, podendo posteriormente digitá-los e recuperar seus dados exclusivamente via software. (BERGERON, 1998)

Os campos de entrada nessa técnica precisam estar localizados no formulário digitalizado antes de serem detectados e analisados. As marcas de registro pré-impressas, conhecidas como âncoras, permitem que os locais dos campos sejam determinados com mais precisão (LOKE; KASMIRAN; HARON, 2018). O reconhecimento de padrões pode ser usado para extrair os campos, que são convertidos em imagens bitonais (consistem de apenas uma cor de primeiro plano e uma cor de plano de fundo), e marcas podem ser determinadas pela contagem mínima de *pixels*.

O SOMR possui como principal vantagem sobre o OMR baseado em *hardware* os baixos custos de fornecimento, requerendo apenas papel comum e tinta da impressora, enquanto o computador e o *scanner* podem ser usados para outros fins quando não estiverem processando formulários.

2.1.3 Desafios

O processo de OMR enfrenta alguns conhecidos inimigos que podem prejudicar a qualidade dos resultados obtidos (LOKE; KASMIRAN; HARON, 2018). São eles:

- Campos pequenos: caso os campos OMR sejam muito pequenos, em muitos casos o processo de decisão dependerá de apenas alguns *pixels*, tornando os dados disponíveis para o algoritmo de reconhecimento muito pequenos para tomar uma decisão precisa, assim, provavelmente aumentando as margens de erro.
- Imagens com baixa resolução: Se o número de Dots Per Inch (DPI) (número de pontos que podem ser encontrados em uma polegada na imagem) forem baixos, isso afeta a qualidade da imagem e, por sua vez, a qualidade dos campos OMR e o conteúdo deles.

- Imperfeições devido à digitalização: as imagens podem apresentar imperfeições (ruído) e perda de informações durante o processo de digitalização, isso pode ocorrer devido a limitações do equipamento utilizado, devido ao alongamento ao longo dos eixos, por causa do mecanismo de alimentação automático de documentos do *scanner* ou a fatores humanos (manter o alinhamento 100% correto na digitalização é uma difícil tarefa).

2.2 IMAGENS DIGITAIS

2.2.1 Representação

Uma imagem pode ser definida como uma função bidimensional f(x,y), em que x e y são coordenadas em um plano e a amplitude de f em qualquer par (x,y) representa a intensidade ou o nível de cinza no ponto. Quando essa função é discretizada, ou seja, os valores x e y e sua amplitude são finitos, tanto em coordenadas espaciais quanto no brilho, chamamo-a então de uma imagem digital (GONZALEZ; WOODS, 2010).

Essa função, por sua vez, para ser interpretada por um *software*, é geralmente representada como arranjos bidimensionais de pontos, denotados *pixels*, sendo esses conjuntos interpretados pelo *software* como *arrays* de números digitais.

Na Figura 2, apresenta-se a notação matricial que comumente utiliza a fim de facilitar sua varredura e a localização de um pixel na mesma. O primeiro índice (m) denota a posição da linha na qual o pixel se encontra, enquanto o segundo (n) denota a posição da coluna. Se a imagem digital contiver M linhas e N colunas, o índice m variará de 0 a M-1, enquanto n variará de 0 a N-1.

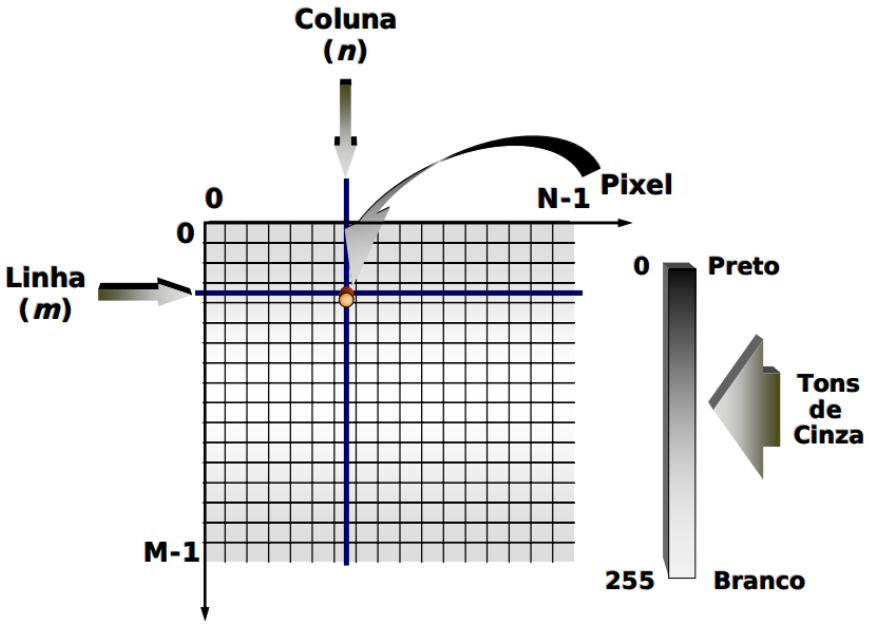


Figura 2 – Representação espacial de uma imagem bidimensional.

2.2.2 Amostragem e Quantização

Para a criação de uma imagem digital, durante o processo de aquisição da mesma por meio de sensores, são necessários dois processos, a amostragem e a quantização, sendo esses fatores diretamente relacionados com a qualidade da imagem (GONZALEZ; WOODS, 2010). Define-se amostragem como a digitalização dos valores de coordenadas, e quantização como a digitalização dos valores de amplitude.

Para exemplificação, na Figura 3 (a) mostra-se uma imagem contínua que pretende-se converter para o formato digital, para isso, primeiramente é obtido os valores de amplitude dos *pixels* de uma linha, como representados graficamente pela Figura 3 (b) os valores referentes ao segmento AB da Figura 3 (a). Após isso, colhem-se amostras igualmente espaçadas nesse agrupamento de valores, e esse conjunto dessas localizações discretas nos dá a função de amostragem, como representado na Figura 3 (c).

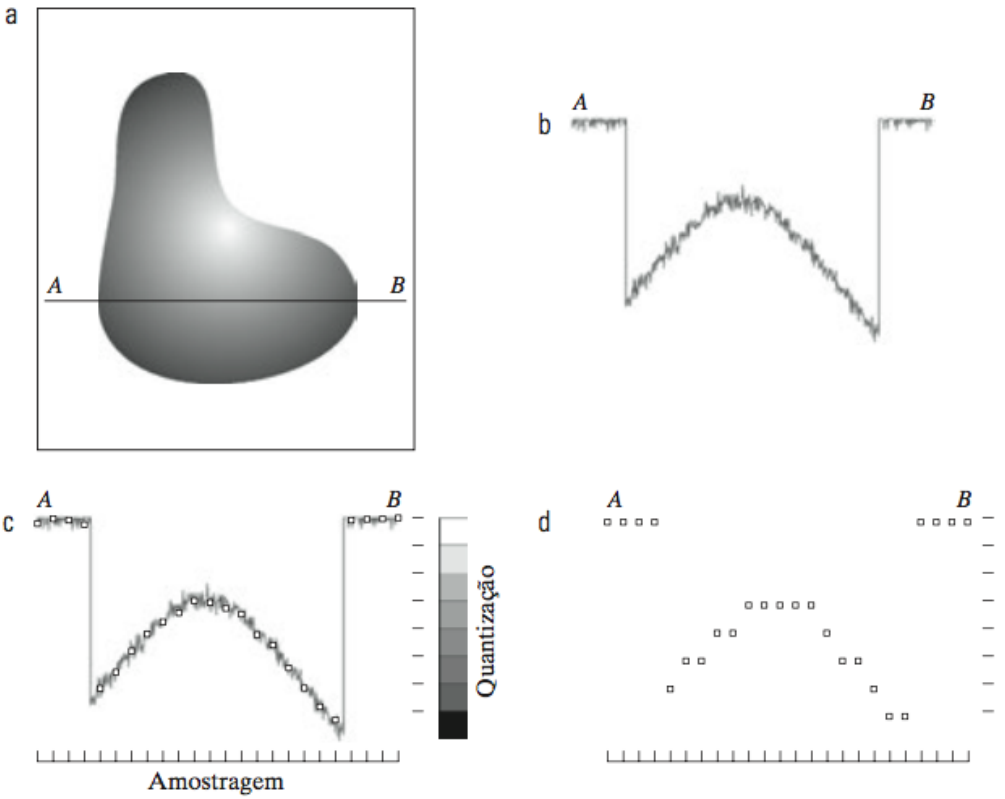


Figura 3 – Exemplo de amostragem e quantização.

Porém os valores das amostras ainda cobrem uma faixa contínua, logo é necessário converter também os valores de intensidade (quantização). Para tanto, cada valor é quantizado atribuindo-se um dos oito valores, apresentados na escala de intensidade presente à direita na Figura 3 (c), com base na proximidade vertical de uma amostra a uma marca indicadora. As amostras digitais resultantes da amostragem e da quantização podem ser vistas na Figura 3 (d). Esse processo é feito linha por linha, começando na parte superior, resultando em uma

imagem digital bidimensional.

Assim, presume-se que, na amostragem, quanto maior as descontinuidades entre os *pixels*, maior a perda de informação e consequente ruído, o que implica na afirmação de que durante a fase de aquisição das imagens surgem as principais fontes de ruídos (GONZALEZ; WOODS, 2010). Esse fato pode acarretar sérios prejuízos para as aplicações que utilizam essa imagem digitalizada, logo, para a tratativa dessas problemáticas, são utilizados métodos de processamento de imagens no domínio do espaço ou no domínio da frequência (MENESES, 2012).

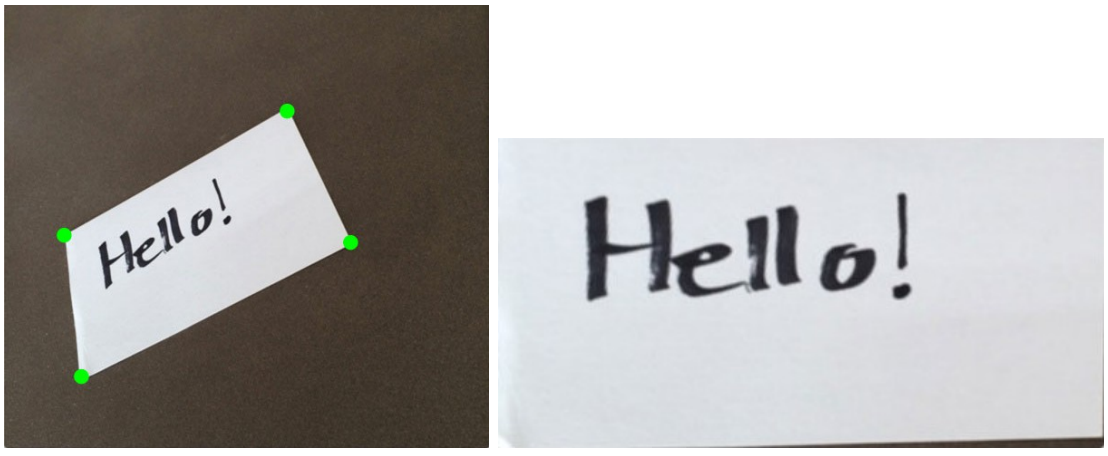
2.3 PROCESSAMENTO DIGITAL DE IMAGENS

Processamento Digital de Imagens (PDI) pode ser definido como técnicas voltadas para a análise de dados multidimensionais, em que o objeto de entrada e de saída são imagens, assim, não existindo fronteira clara entre processamento de imagens e visão computacional (GONZALEZ; WOODS, 2010).

A seguir, serão introduzido alguns tópicos da área de PDI como a transformação de perspectiva, segmentação, limiarização e Hough Transform.

2.3.1 Transformação Espacial de Perspectiva

Uma transformação espacial da perspectiva de uma imagem é uma transformação geométrica do sistema de coordenadas da imagem, em que cada ponto (x,y) é mapeado para um novo ponto (m, n) (RHODY, 2005). Essa técnica é amplamente utilizada no registro de imagens, na correção dos posicionamentos e na remoção de distorções, conforme a Figura 4:



(a) Imagem original. **(b)** Após transformação 4 pontos de base.

Figura 4 – Exemplo de transformação de perspectiva.

Observa-se que há mais especificamente transformações afins, que são definidas como qualquer transformação que preserve a colinearidade (todos os pontos antes presentes em uma linha permanecem em uma linha após a transformação) e as proporções de distâncias (ponto

médio de um segmento mantém-se como o ponto médio após a transformação) (RHODY, 2005).

Uma aplicação desse método é apresentada na Figura 4, visando à sintetização do conteúdo relevante da imagem (no caso, o papel com a mensagem no centro da Figura), determinam-se os quatro pontos referentes aos vértices do papel, e recortando o interior do quadrilátero formado por esses pontos, aplica-se uma função linear, a fim de alterar a perspectiva da imagem, e assim, facilitar a análise da mesma.

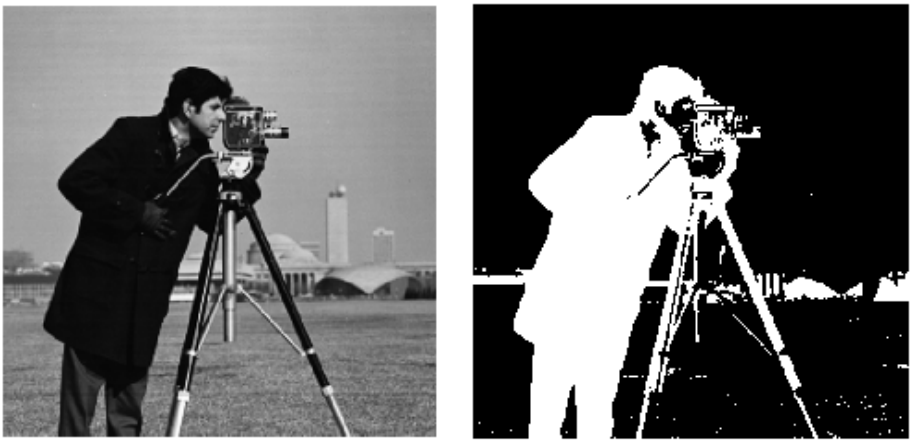
2.3.2 Segmentação

O processo de segmentação de imagens é o definido como o particionamento, uma imagem, segmentando-a em regiões de modo que a interseção entre duas partes distintas seja vazia e a união de todas as regiões resulte na imagem original (HARALICK; SHAPIRO, 1992), isto, consiste em dividir uma imagem em conjuntos, a fim de simplificar e/ou alterar a representação da mesma, visando a auxiliar e a facilitar a análise dessa imagem.

O processo de segmentação pode ser aplicado com diferentes intuitos, sendo comum a finalidade de separar objetos ou segmentos em uma imagem de seu fundo (GONZALEZ; WOODS, 2010).

2.3.3 Limiarização

Limiarização é um processo de PDI, mais especificamente de segmentação de imagens. Esse processo é calcado na diferenciação dos níveis de cinza nos objetos representados em uma imagem. A operação consiste em determinar um ou mais limiares de divisão entre os níveis, de tal modo que se define um grupo de valores acima do limiar e um grupo de valores abaixo, atribuindo a cada grupo um valor fixo a todos os *pixels*. Na Figura 5 é possível ver o resultado da aplicação desse processo.



(a) Imagem original. **(b)** Após limiarização.

Figura 5 – Exemplo de limiarização com método Otsu.

Processa-se a operação determinando um limiar único, o que contribui para transformar a imagem em uma imagem binária, ou seja, terá somente ou a cor preta ou a cor branca para cada pixel (MARQUES; VIEIRA, 1999).

Destaca-se o método Otsu de limiarização, como exemplificado na Figura 5, em que é usado para determinar automaticamente o limiar da imagem. O algoritmo retorna um limite de intensidade único, ou seja, determina um limiar único, que separa os *pixels* em duas classes, primeiro e segundo plano. Esse limite é determinado minimizando a variação de intensidade intraclasse ou, equivalentemente, maximizando a variação entre classes (SEZGIN; SANKUR, 2004).

2.3.4 Circle Hough Transform

Circle Hough Transform (CHT) é uma técnica de extração de recursos para detecção de círculos em uma imagem. É uma especialização do método Hough Transform (algoritmo que gera a extração de instâncias imperfeitas, com uma forma específica, em uma imagem, através de um procedimento de votação)(SHAPIRO; STOCKMAN, 2001). O objetivo da técnica é encontrar círculos com formatos imperfeitos em imagens.

Esse método determina um contorno como sendo um círculo partindo do princípio de que para cada ponto (x, y) no contorno, definindo-se um círculo como o mesmo como centro e com raio igual à metade do comprimento do contorno, se houver um ponto de interseção de todos esses círculos no espaço de parâmetros, esse corresponderia ao ponto central e o contorno em questão seria um círculo.

3 MATERIAIS E MÉTODOS

O desenvolvimento do sistema proposto pode ser divido em cinco etapas: aquisição das imagens, segmentação, leitura/aquisição dos dados, exportação e comparação dos resultados. Cada uma delas engloba conceitos e técnicas de processamento digital de imagens, de reconhecimento óptico de marcas ou de aquisição de imagens digitais.

A Figura 6 apresenta um esquema geral do funcionamento do sistema.

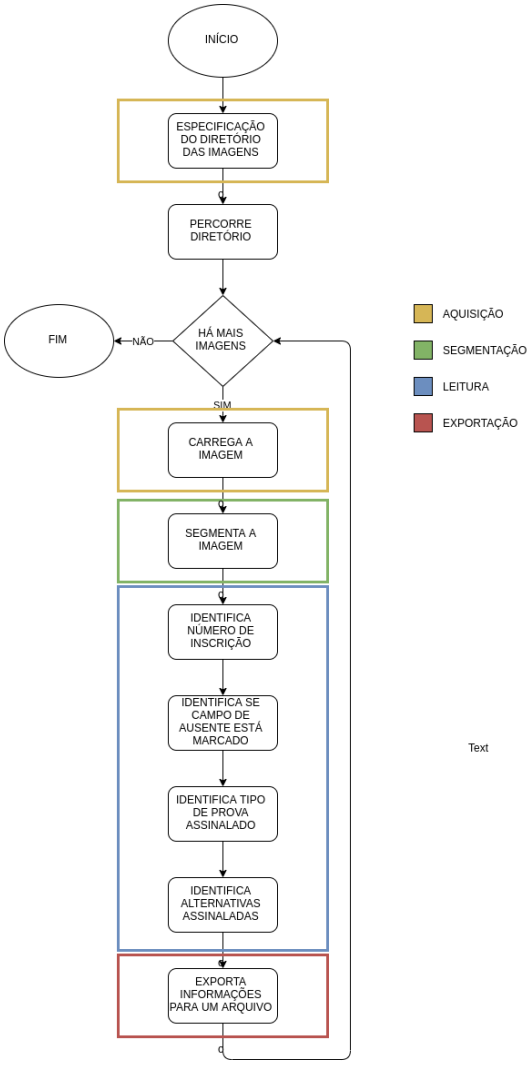


Figura 6 – Diagrama geral simplificado do sistema.

Como apresentado acima, o sistema inicia recebendo do usuário um argumento que especifica o diretório das imagens a serem processadas, a aplicação então percorre o diretório processando cada imagem (demarcado com marrom no diagrama), nesse processo, para cada imagem, a aplicação segmenta-a, dividindo-a segundo as informações que se deseja colher (demarcado com verde no diagrama). Para cada segmento, o sistema processa e extrai as

informações (demarcado com azul no diagrama), e após isso, exporta os dados adquiridos em um arquivo de extensão csv (demarcado com vermelho no diagrama).

Todo o sistema foi desenvolvido utilizando a linguagem Python, com o auxílio da biblioteca OpenCV, uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto (OPENCV, 2019), e da biblioteca ZBar, um conjunto de software de código aberto para leitura de códigos de barras (ZBAR, 2010). Para os experimentos foi utilizado um computador com processador de 3,2 GHz (Intel Core Í5-4460) e 8 GB de memória RAM.

A seguir será apresentada cada etapa do sistema.

3.1 AQUISIÇÃO DAS IMAGENS

A aquisição das imagens inicia-se recebendo do usuário um argumento, por meio da linha de comando, especificando com a *flag* "-i" um diretório contendo as imagens dos gabaritos escaneados a serem processados.

O Código 1 apresenta o comando para inicialização da aplicação a partir da pasta raiz do projeto, em que se substitui "<PATH>" pelo diretório correspondente à base de imagens selecionada.

Código 1 – Comando para execução da aplicação.

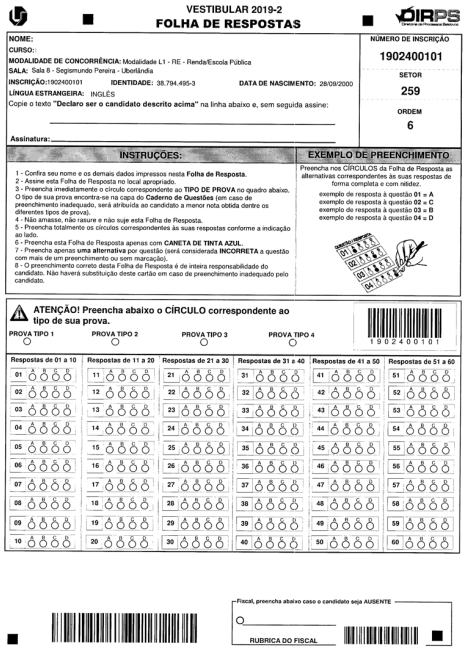
```
1 python3 src/main.py -i <PATH>
```

Após identificado o diretório, o sistema percorre-o, carregando as imagens por meio da função "imread"da biblioteca OpenCV, a qual realiza o processo de aquisição por meio da amortização e da quantização da imagem, representando-a como uma matriz.

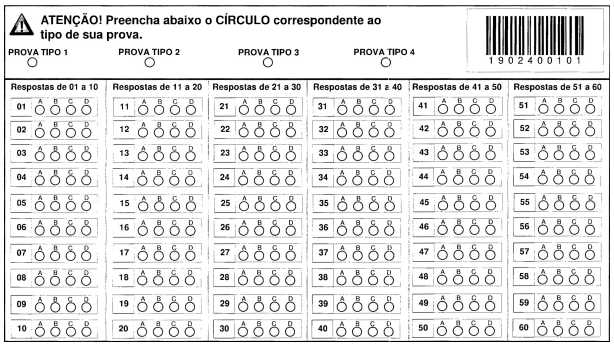
3.2 SEGMENTAÇÃO

Uma vez carregadas as imagens, o sistema deve buscar os elementos os quais possuem as informações relevantes para o processo, esses são: o número de inscrição, o tipo de prova assinalado e as alternativas assinaladas para cada questão no gabarito.

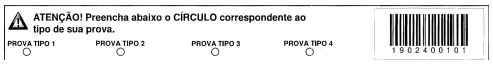
No caso do vestibular UFU 2019-2, todos esses elementos estão localizados na parte central do gabarito, que pode ser visto na Figura 7 (a), mais especificamente em uma tabela composta por cabeçalho, contendo os campos para demarcação do tipo de prova e o número de inscrição (representado como código de barras de simbologia Code-128), e corpo, contendo as questões e seus respectivos campos para demarcação das respostas. Dá-se a essa tabela o nome de Tabela de Conteúdo (TDC), para auxiliar na compreensão.



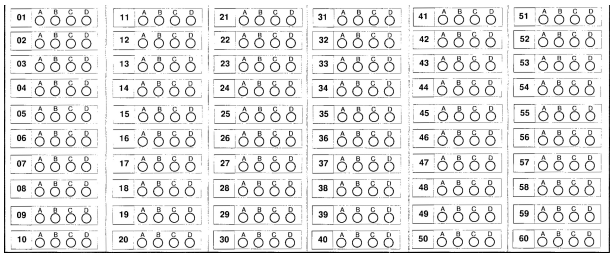
(a) Imagem original.



(b) Após transformação com 4 pontos de base.



(c) Segmento do cabeçalho.



(d) Segmento da tabela de questões.

(e) Segmento dos tipos de provas.

(f) Segmento do número de inscrição.

Figura 7 – Segmentação da prova.

Posto isso, compreende-se que, para a aquisição dos dados das provas, o primeiro passo é segmentar a imagem em subgrupos, segundo a informação a qual se quer adquirir.

Para o reconhecimento da TDC, deve-se primeiramente diminuir a área de busca a fim de facilitar e otimizar o processo, logo, o sistema define e recorta a porção equivalente a um retângulo com vértice superior esquerdo posicionado no ponto (0, h * 0.4), de largura w, e altura h * 0.4, tal que h é a altura e w é a largura da imagem original. Opta-se por uma área maior de corte para compensar o posicionamento por vezes irregular das imagens, esse por conta do processo de escaneamento.

Para recortar a imagem, sendo o segmento almejado um retângulo, pode-se utilizar a função 'splitImage' expressa no Código 2, tal que x e y são as coordenadas do ponto inicial (aresta superior esquerda do retângulo), h é a altura e w é a largura do retângulo. Após recortada a imagem, teremos uma área aproximadamente 60% menor para buscar a TDC.

Código 2 – Código em python para seleção de segmento de uma matriz / recortar área retangular de uma imagem.

```
1 def splitImage(imagem, y, x, h, w):
2     return imagem[y: y + h, x: x + w]
```

Após reduzida a área, pode-se buscar pelas extremidades referentes ao contorno da TDC, porém, devido a possíveis desalinhamentos e ruídos provenientes do processo de escaneamento, como apresentado na Figura 8 (leve desalinhamento e descontinuidade na borda), pode haver problemas na definição do contorno da tabela.

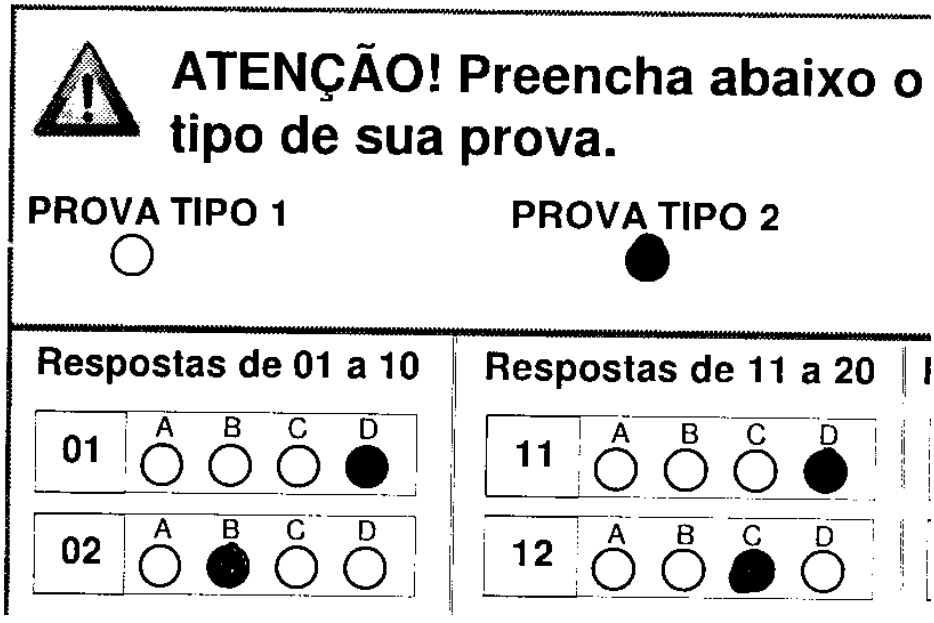


Figura 8 – Exemplo real de ruído e desalinhamento nas imagens escaneadas.

A fim de resolver essa problemática e otimizar a segmentação dos elementos da TDC

(cabeçalho e questões), aplica-se uma transformação morfológica de fechamento, para corrigir os 'buracos' e descontinuidades no contorno da tabela. Aplica-se também uma função de transformação espacial de perspectiva de quatro pontos no contorno da TDC para corrigir o desalinhamento.

No Código 3 faz-se uso dessa solução, a saber, da linha 5 à linha 9, aplica-se um processo de transformação morfológica de fechamento, utilizando a biblioteca OpenCV (importada como cv2). Esse processo consiste respectivamente em converter a imagem em uma outra imagem em escala de cinza, suavizar a mesma com um filtro bilateral, buscar os contornos (função Canny), identificar os elementos da imagem com base nos contornos e aplicar uma transformação morfológica de fechamento nos elementos encontrados.

Código 3 – Código em python para identificação do contorno da Tabela de Conteúdo e aplicação de transformação espacial de perspectiva.

```
1  import cv2
2  from imutils.perspective import four_point_transform
3
4  def getTable(content):
5      gray = cv2.cvtColor(content, cv2.COLOR_BGR2GRAY)
6      gray = cv2.bilateralFilter(gray, 1, 10, 120)
7      edges = cv2.Canny(gray, 10, 250)
8      kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
9      closed = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)
10     contours, _ = cv2.findContours(closed, cv2.RETR_EXTERNAL,
           cv2.CHAIN_APPROX_SIMPLE)
11     contours = sorted(contours, key=cv2.contourArea,
           reverse=True)
12
13     paper = None
14
15     for cnt in contours:
16         arc_len = cv2.arcLength(cnt, True)
17         approx = cv2.approxPolyDP(cnt, 0.1 * arc_len, True)
18
19         if len(approx) == 4:
20             paper = approx
21             break
22
23     table = four_point_transform(content, paper.reshape(4, 2))
24
25     return table
```

Após preparada a imagem, como apresentado nas linhas 10 à 21, busca-se os contornos na imagem e determina-se o contorno relativo à TDC como o contorno com 4 arestas de maior área, ou seja, o maior retângulo da imagem. Com o contorno da TDC identificado, pode-se

agora aplicar a transformação de perspectiva utilizando essas arestas como os 4 pontos de base (etapa apresentada na linha 23).

Após esse processo, tem-se como saída a Figura 7 (b), uma imagem sintetizada da TDC de proporção fixa e com os desalinhamentos compensados. Logo, para os próximos passos, pode-se utilizar proporções fixas que as mesmas servirão para todas imagens da base sem diferenças relevantes.

Um vez com a TDC definida, segmenta-a em duas partes por meio da função exposta no Código 2, sendo a área superior, relativa à 20% da imagem, referente ao cabeçalho (Figura 7 (c)), e os 80% restantes, referentes à tabela de questões (Figura 7 (d)). O cabeçalho, por sua vez, é segmentado em duas partes: 75% iniciais da largura apresentam o tipo de prova (Figura 7 (e)), 25% finais apresentam o código de barras com o número de inscrição (Figura 7 (f)).

A segmentação das questões, por sua vez, se dá com o mesmo processo, porém, em 2 passos. Primeiramente reparte-se horizontalmente a tabela de questões, Figura 7 (d), em 6 partes de igual tamanho, de forma que cada coluna de questões seja contemplada em um segmento. Assim, no segundo passo, para cada um dos segmentos gerados no primeiro passo, reparte-o em 10 partes iguais, englobando em cada parte uma questão.

Esses procedimentos de segmentação resultam em:

- 1 segmento contendo as alternativas para o tipo de prova;
- 1 segmento contendo o código de barras relativo ao número de inscrição;
- 60 segmentos ordenados de igual tamanho, sendo cada um relativo a uma questão.

3.3 LEITURA

Uma vez segmentado, necessita-se abstrair as informações dos segmentos gerados. Para isso, nos campos de múltipla escolha, deve-se definir a alternativa(s) marcada(s) pelo vestibulando e ler o número de inscrição contido no código de barras.

No sistema proposto, o número de inscrição é obtido com o auxílio da biblioteca ZBar e do múdulo zbar-py (ambos de código aberto), um modulo compatível com a linguagem python que provê uma interface para a biblioteca. A leitura se dá passando o segmento relativo ao código de barras, como parâmetro 'img', para o Código 4.

A função de leitura dos campos de múltipla escolha, aplicada para cada uma das 60 questões, é apresentada na Figura 9. Ela consiste em dois métodos que na inviabilidade do primeiro, o segundo assegura a leitura das informações.

Essa função, primeiramente, busca por círculos na imagem por meio do método 'HoughCircles' da biblioteca OpenCV, a qual busca circunferências com raio entre 15 a 25

Código 4 – Código em python para decodificação de código de barras de simbologia Code 128, utilizando o biblioteca ZBar.

```
1 import pyzbar.pyzbar as pyzbar
2
3 def getSubscriptionCode(img):
4     decoded = pyzbar.decode(img, {128})[0]
5
6     return decoded.data
```

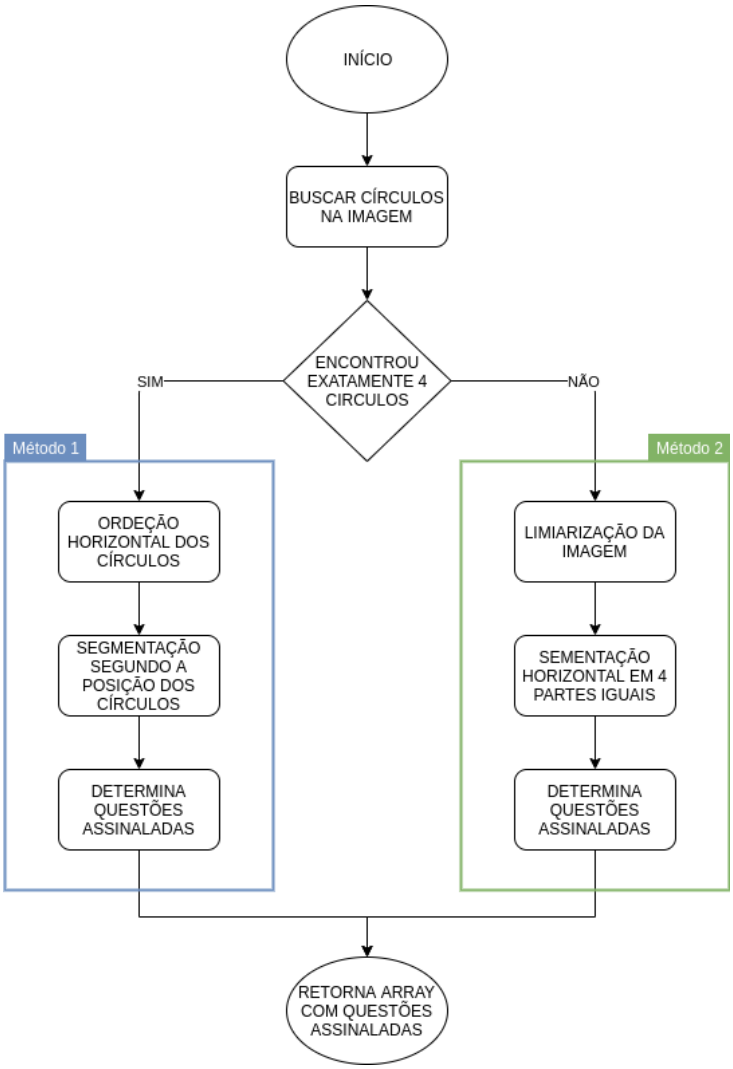
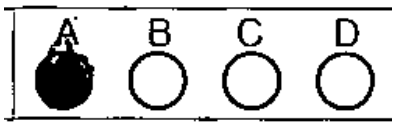


Figura 9 – Diagrama da função de leitura de questões de múltipla escolha.

pixels em uma imagem de *input* similar à Figura 10 (a). Caso encontre os 4 círculos, que correspondem às 4 alternativas da questão, aplica-se o que nomeia-se de método 1, e caso não, aplica-se o que nomeia-se de método 2.

No método 1, as circunferências encontradas são ordenadas horizontalmente pelo eixo X, a fim de usar suas posições para determinar a qual letra/alternativa as mesmas se referem. Posteriormente, recortam-se os segmentos referentes a cada círculo, Figura 10 (b), e aplica-se uma função para determinar se a mesma está marcada. Essa função, por sua vez, conta a quantidade de *pixels* no segmento que não tem valor igual a 0, e caso seja igual ou superior a 75% da imagem, é considerado assinalado.

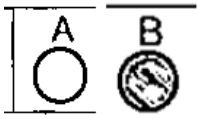
No método 2, descartam-se as circunferências encontradas e direciona-se a busca para o posicionamento das alternativas, assim, aplica uma limiarização na imagem a fim de remover marcas claras e destacar possíveis marcações, reparte horizontalmente a imagem em 4 segmentos de igual tamanho, Figura 10 (c), e aplica uma função para determinar se a mesma está marcada. Essa função de decisão, por sua vez, realiza o mesmo processo de contagem dos *pixels* não 0 do método 1, porém, determina como assinalada se o valor for menor que 10% da imagem (menor pois, como a imagem foi limiarizada, as marcações em preto foram invertidas).



(a) Exemplo de um dos 60 segmentos referentes às questões.

(b) Segmento oriundo do método 1.

(c) Segmento oriundo do método 2.



(d) Exemplo de segmentos com quantidade similar de *pixels* brancos.

Figura 10 – Exemplo de segmentos relacionados à função de leitura das questões.

Assim, também define que o método 1 é prioritário ao método 2, uma vez que, sintetizando a área usada para determinar a marcação, contornam-se problemas como o apresentado na Figura 10 (c). Esses problemas ocorrem por que alguns campos podem ter sido mal assinalados ou apresentar perda de informações devido ao processo de escaneamento, enquanto outros podem incluir bordas alheias no processo de contagem, fazendo com que, como o apresentado na Figura 10 (c), ambas as imagens tenham o mesmo número de *pixels* brancos.

Na leitura do tipo de prova assinalado, uma vez que, devido ao seu processo de segmentação gerar um segmento somente com as alternativas, como exibido na Figura 7 (e), é

possível determinar com melhor exatidão a posição das alternativas, não necessitando incluir bordas ou letras na seleção, assim, torna-se plausível o uso fixo do método 2 da Figura 9.

3.4 EXPORTAÇÃO

Após realizada a leitura dos dados, é necessário exportá-los de forma clara, agrupada e direta. Para isso, durante o processo de leitura, as informações recolhidas são armazenadas em um *array* e ao final são persistidas em um arquivo de formato Comma Separated Value (CSV), um tipo de arquivo de texto leve e legível por basicamente qualquer software de edição de texto (REPICI, 2002).

Como exibido na Tabela 1, são exportados em cada linha, respectivamente:

1. Nome do arquivo lido (coluna ARQUIVO);
2. Número da inscrição (coluna INSCRIÇÃO);
3. Tipo de prova assinalado (coluna TIPO);
4. Alternativa assinalada para cada questão (colunas 1 a 60), a saber:
   - Se somente uma alternativa foi assinalada, exibe a letra correspondente à alternativa;
   - Se mais de uma alternativa foi assinalada, exibe o caractere '+';
   - Se nenhuma alternativa foi assinalada, exibe o caractere '-'.

Tabela 1 – Representação de arquivo exportado pelo sistema.

ARQUIVO	INSCRIÇÃO	TIPO	1	2	3	4	5	...	57	58	59	60
1902400107.tif	1902400107	4	D	C	D	B	A	...	A	C	D	B
1902401002.tif	1902401002	1	B	B	C	A	B	...	C	+	A	D
1902400092.tif	1902400092	3	-	B	-	-	D	...	D	A	D	A
1902400463.tif	1902400463	2	D	B	A	+	B	...	C	D	C	A

Em casos de erro durante o processamento, o sistema exporta somente duas colunas, sendo a primeira apresentando o nome do arquivo e a segunda apresentando a palavra 'ERRO'.

3.5 TESTES E COMPARAÇÃO DOS RESULTADOS

Visando checar e garantir o funcionamento do sistema proposto, foi gerado pela Diretoria de Processos Seletivos da UFU uma planilha listando as alternativas assinaladas por cada

participante na 1ª etapa de seleção vestibular UFU 2019-2. A planilha com as respostas contempla o número de inscrição seguido pelas alternativas assinaladas para cada questão, bem como, se for o caso, o registro de que houve marcações duplas ou nenhuma marcação. Essa planilha é tida como totalmente correta para fins de comparação.

Para testar o sistema, foi utilizado uma base composta por 26.363 gabaritos escaneados da 1ª etapa do vestibular, também cedidos pela Diretoria de Processos Seletivos da UFU, como *input* para o sistema. Para a comparação das planilhas, foi utilizado o número de inscrição como índice em ambas as planilhas, validando linha por linha, e registrando a quantidade de acertos e erros.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos pela metodologia e testes propostos na seção anterior. Uma vez gerado, pelo sistema proposto, um arquivo contendo a relação de candidato e alternativas assinaladas na folha de resposta para cada questão da 1ª etapa de seleção vestibular UFU 2019-2 e tendo-se um arquivo semelhante gerado pela Diretoria de Processos da UFU com os resultados tidos como corretos e esperados pelo sistema.

Espera-se que em comparação ao sistema atual utilizado pela UFU, um *software* OMR comercial, que gasta cerca de 12 horas (testes realizados e cedidos pela Diretoria de Processos Seletivos da UFU), o sistema proposto seja igualmente preciso e demande menor ou similar tempo de processamento.

Enquanto isso, o sistema proposto, com base nos teste exibidos na seção anterior, apresentou, com os 26.363 gabaritos processados, o seguinte desempenho:

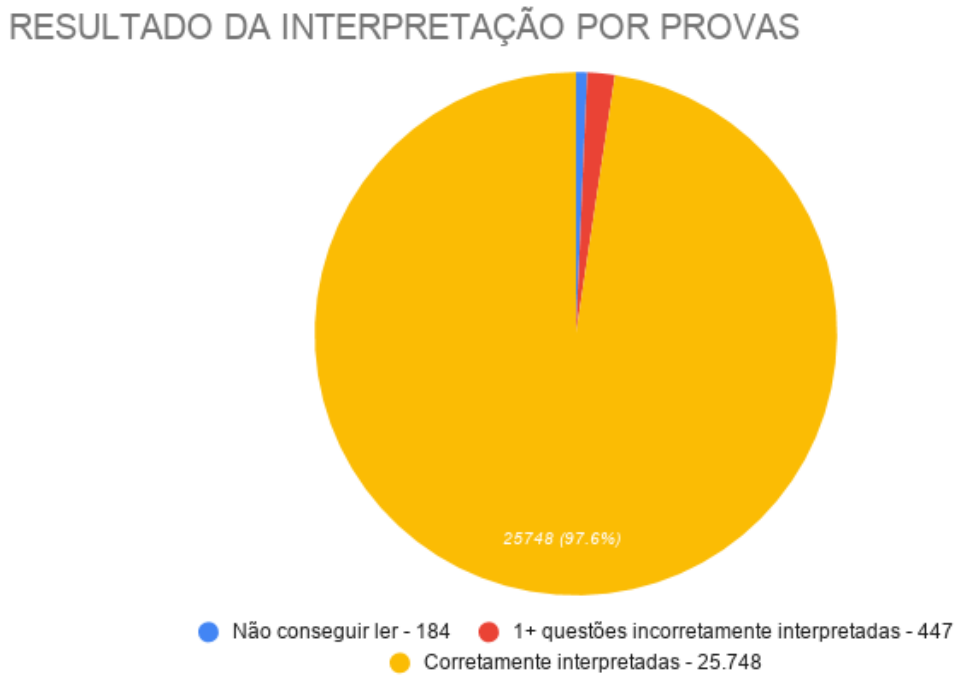


Figura 11 – Gráfico representando o resultado com base no número de provas.

Como exibido no gráfico da Figura 11, com base na número de provas, o sistema apresentou:

- 184 provas o sistema não conseguiu ler;
- 447 provas tiveram pelo menos 1 questão interpretada incorretamente;

- 25.748 provas tiveram todas as questões interpretadas corretamente;
- 97,6% das provas foram interpretadas corretamente;
- 99,3% das provas lidas foram interpretadas corretamente.

Levando em consideração o número de questões interpretadas pelo sistema, tem-se:

Figura 12 – Gráfico representando o resultado da interpretação com base no número de questões.

Como exposto no gráfico da Figura 12, o sistema apresentou:

- 1.558 questões interpretadas incorretamente;
- 1.570.142 questões interpretadas corretamente;
- 99,9% das questões corretamente interpretadas.

Quanto ao tempo gasto para o processamento, foi utilizado a função 'timeit' do módulo, de mesmo nome, embutido na linguagem Python para calcular o tempo gasto na execução do código. Obteve-se o tempo médio de 0,20261870499962242 segundos para o processamento de uma base com 1 gabarito e de 4074.5773104990003 segundos para a base de 26.363 gabaritos, ou seja, aproximadamente 1 hora e 10 minutos para o processamento da base da 1ª etapa de seleção do vestibular.

Logo, o sistema proposto apresentou, aproximadamente:

- 97,6% de sucesso na leitura das provas;
- 99,9% de sucesso na detecção das alternativas assinaladas;
- 1h e 10 min de tempo demandado para a execução do processo.

Analisando os casos de erro na leitura das provas e na interpretação das questões, observou-se alguns pontos responsáveis pela maioria dessas ocorrências. Por exemplo, existem marcações nas mediações do contorno da Tabela de Conteúdo, como exibido na Figura 13, um risco próximo ao código de barras conecta as duas tabelas, assim, durante a detecção dos contornos da TDC para aplicação da transformação de perspectiva, causa-se a detecção de um contorno único para as duas tabelas, e esse possui mais de 4 pontos de aresta, o que inviabiliza a leitura do gabarito pelo sistema.

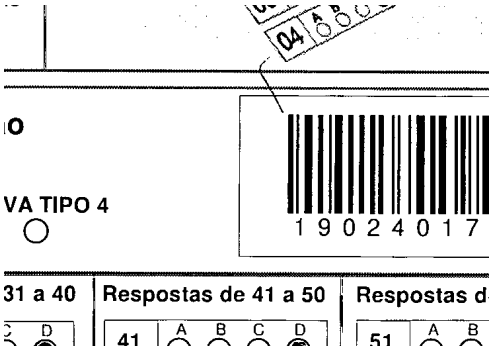


Figura 13 – Exemplo de marcação que pode causar erro na leitura do gabarito.

Além disso, também observou-se que o sistema não detectou marcações de área menor que a especificada (15% para o método 1 e 10% para método 2 da área delimitada para a alternativa), como pode ser visto na Figura 14.

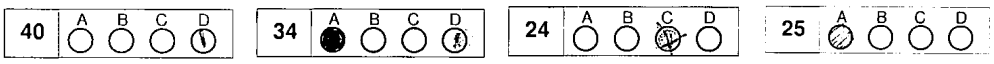


Figura 14 – Exemplos de falha na detecção por área da marcação.

5 CONSIDERAÇÕES FINAIS

O principal objetivo desse trabalho foi a criação de um *software* de OMR utilizando-se somente bibliotecas e tecnologias de código aberto e buscando bons resultados em tópicos como desempenho, custo e assertividade.

Dessa maneira, com base nos resultados alcançados, o sistema proposto apresentou notório desempenho, visto a diferença de tempo de processamento entre o sistema proposto e um sistema comercial, como o utilizado nos processos seletivos da Universidade Federal de Uberlândia.

Quanto a assertividade na interpretação das alternativas assinaladas, mostrou-se satisfatória, dado as relativas alta taxa porcentual de acerto e baixa taxa porcentual de falha de leitura. O custo esperado, por sua vez, supõe-se válido, uma vez observado o uso exclusivo de tecnologias de código aberto, a quantidade de falha de leitura, a qual necessita de interação humana e o tempo de processamento, o qual influência na sustentação do hardware durante o processo.

Mostrou-se necessário trativas para as falhas exibidas na seção anterior, o que pode ser alcançado com alterações nos métodos de detecção e segmentação, provendo alternativas e métodos auxiliares para casos de falha da primeira trativa.

A principal dificuldade encontrada no desenvolvimento desse projeto foi a otimização da assertividade, essa, diretamente afetada pelo ruído oriundo do processo de digitalização dos gabaritos. Assim, requereu-se o levantamento e o teste de diversos métodos de segmentação e de redução de ruídos, comparando os resultados alcançados por cada um e buscando-se o mais assertivo e eficiente para o sistema proposto.

Para trabalhos futuros, espera-se a otimização dos módulos de leitura e detecção do sistema, provendo mais tratativas à sinistros e marcações inesperadas, bem como alterações para tornar o sistema mais genérico, em que a partir de prévia e facilitada configuração feita pelo usuário, como a demarcação das áreas e questões em um *template*, possa-se ler e abstrair as informações de qualquer gabarito de múltipla escolha escaneado.

REFERÊNCIAS

BERGERON, B. P. **Optical mark recognition. Tallying information from filled-in bubbles**. [S.l.: s.n.], 1998. 23–25 p. páginas 13, 16

BLOOMFIELD, Louis A. Why do scantron-type tests only read 2 pencils? can other pencils work? 2006. páginas 16

CALVERT, J.B. Electronics 30 - phototubes. University of Denver, 2002. páginas 16

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento digital de imagens**. [S.l.: s.n.], 2010. v. 3. páginas 17, 18, 19, 20

HARALICK, R. M.; SHAPIRO, L. G. **Computer and Robot Vision**. [S.l.: s.n.], 1992. v. 1. 672 p. páginas 20

IBM - 805 Test Scoring Machine. 2006. <https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_9.html>. Accessed: 2019-10-20. páginas 13

LOKE, Seng Cheong; KASMIRAN, Khairul A.; HARON, Sharifah A. A new method of mark detection for software-based optical mark recognition. **Journal Plos One**, p. 1–15, 2018. páginas 16

MARQUES, Ogê; VIEIRA, Hugo. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. v. 1. páginas 21

MENESES, Paulo Roberto. **Introdução ao processamento de imagens de sensoriamento remoto**. [S.l.: s.n.], 2012. 01–33 p. páginas 19

OPENCV. 2019. <https://opencv.org>. Accessed: 2019-11-16. páginas 23

REPICI, Dominic John. **CSV - The Comma Separated Value File Format**. 2002. <http://creativyst.com/Doc/Articles/CSV/CSV01.htm>. Accessed: 2019-11-20. páginas 30

RHODY, Harvey. Geometric image transformations. **Chester F. Carlson Center for Imaging Science**, p. 1–38, 2005. páginas 19, 20

SEZGIN, Mehmet; SANKUR, Bülent. Survey over image thresholding techniques and quantitative performance evaluation. **Journal of Electronic Imaging**, v. 13, p. 146–165, 2004. páginas 21

SHAPIRO, Linda; STOCKMAN, George. **Computer Vision**. [S.l.]: Prentice-Hall, 2001. páginas 21

ZBAR. 2010. <http://zbar.sourceforge.net/>. Accessed: 2019-11-16. páginas 23