Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Eletrônica – DAELN
Engenharia Eletrônica
Disciplina: IF69D – Processamento Digital de Imagens
Semestre: 2020/1 ADNP
Prof.: Gustavo B. Borba

RELATÓRIO
Sistema de OMR - Projeto Final

Alunos:
Giovanni de Rosso Unruh / 1188232
Franklin Ronaldo Martins Tavares Junior / 1889001

Novembro 2020

1. Objetivo

Implementar um sistema de OMR para ser aplicado na correção de provas. O sistema desenvolvido deve ser capaz de:
1. Adquirir as imagens das folhas de resposta (FR);
2. Identificar as MO e a resposta associada a cada questão;
3. Comparar as respostas com as de um gabarito;
4. Atribuir uma nota.

2. Fundamentação Teórica

No desenvolvimento deste trabalho foram utilizadas técnicas de processamento de imagem como detecção de bordas, formas e re-escala. Nos parágrafos a seguir é feita uma breve descrição da teoria por trás dessas técnicas. Lembrando que para o projeto a grande maioria desses algoritmos já é fornecida pelo ambiente de programação, então não foi preciso implementá-las.

2.1 Detecção de bordas:

A detecção de bordas basicamente é detectar diferenças abruptas nos níveis de cinza entre pixels próximos, ou de outra maneira, o gradiente da imagem, visto que esse operador resulta na diferenciação entre as cores dos pixels. Para obter o gradiente utiliza-se de máscaras em geral de ordem NxN (2x2, 3x3…) aplicando com o píxel a ser calculado no centro. Alguns exemplos são as máscaras Roberts, Prewitt e Sobel (adaptado do livro de referência).

2.2 Detecção de forma de objeto:

Para detectar formas em uma imagem utiliza-se a transformada de Hough. A ideia é encontrar um conjunto de pontos P(xi,yi) que seja um subconjunto da forma a se encontrar. O algoritmo 5.2 de (livro referência) descreve como detectar retas em uma imagem (adaptado do livro).

Para casos onde a forma possa ser dividida em um conjunto razoável de retas o método citado pode ser usado. Como exemplo para detecção de quadrados em uma imagem, que é um conjunto de 4 retas.

2.3 Conversão de cores:

Uma imagem RGB 888**,** 256 valores possíveis para cada cor (vermelho, verde e azul), pode ser convertida para níveis de cinza com a seguinte equação:

$$Pi = 0.333 * Ri + 0.5 * Gi + 0.1666 * Bi \quad (1)$$

onde Pi é o pixel índice i em nível de cinza e Ri, Gi e Bi são no niveis RGB do pixel i (adaptado de A Theory Based on Conversion of RGB image to Gray image) [1].

2.4 Reescala:

A transformação de uma imagem de WxH pixels para 2*Wx2*H deve ser feita com um algoritmo de interpolação para tentar preservar as características da imagem original. Para isso os principais métodos de interpolação são vizinhos mais próximo, bilinear, bicúbica e polinômios de Lagrange (adaptado do livro de referência).

2.5 Limiarização:

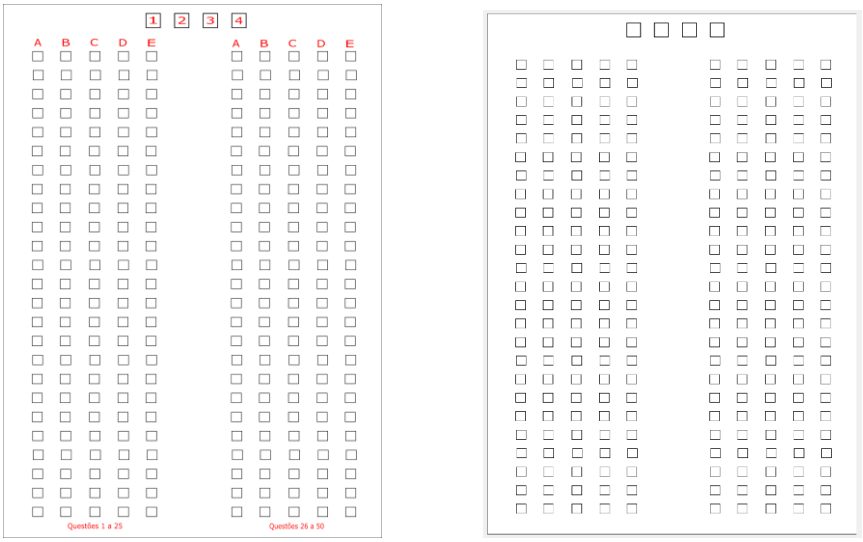
"A limiarização é uma das técnicas mais simples de segmentação e consiste na classificação dos pixels de uma imagem de acordo com a especificação de um ou mais limiares" (citação direta do livro de referência). Podemos então definir um nível entre os 256 valores para ser o limiar para decidir se um pixel é preto ou branco na imagem e aplicá-lo de forma global ou de forma local, onde para o local podemos definir o tamanho da janela que será usada.

2.6 Crescimento de regiões:

"A segmentação baseada em crescimento de regiões é um procedimento que agrega pixels com propriedades similares em regiões" (citação direta do livro de referência). Essa técnica pode ser usada para realçar estas regiões, definir áreas de interesse para outros processos ou até mesmo filtragem para retirar ruídos dessas regiões.

3. Implementação

As imagens são adquiridas através do aplicativo CamScanner, que irá corrigir a perspectiva e planificar a imagem. Assim que carregada no nosso algoritmo a imagem é tratada, para termos um padrão na análise passamos ela em um *resize* para deixar em 2970 x 2100, o tamanho foi opcional mesmo, após o *resize* passamos a imagem em um filtro RGB atenuando os tons vermelhos, visto que na nossa folha de respostas criamos detalhes em vermelhos que não serão necessários na análise, apenas para facilitar o preenchimento pela pessoa. Como abaixo:



(a) *(b)*

Figura 1- (a) Imagem antes do filtro de vermelho. (b) Imagem depois do filtro de vermelho

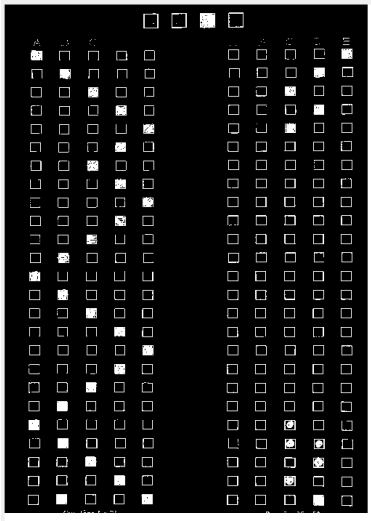
Com a imagem BW, partimos para o processamento buscando facilitar a obtenção dos quadrados de resposta.

Para tal, utilizamos a seguinte sequência de funções:

3.1 *Adaptative threshold*:

É uma implementação de limiarização local, criada durante o decorrer do semestre. A ideia da função é, dada uma imagem "*img*", um tamanho de janela "*windowSize*" e o passo "*step*" percorrer toda a imagem aplicando a técnica *otsu* para limiarização nesta janela apenas, percorrer o "*step*" fornecido e aplicar novamente, até percorrer toda a imagem. Percebemos que para fotografias a aplicação desse método acabava sendo mais eficiente do que aplicar a limiarização em toda imagem de uma vez só.

Para o uso correto das funções seguintes, utilizamos o negativo da imagem resultante do *Adaptative threshold*.

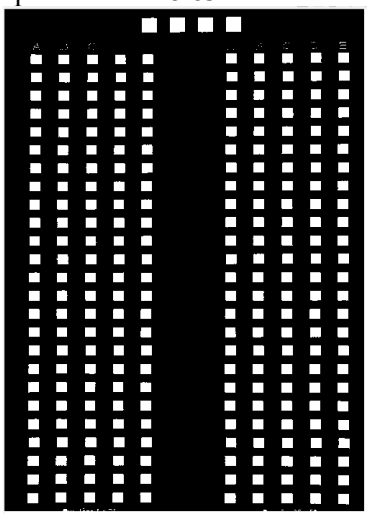


(a)

Figura 2 - (a)Imagem negada resultante do Apadtative threshold.

3.2 Imfill:

Para facilitar a captura das posições dos quadrados de resposta, utilizamos a função *Imfill* que faz um preenchimento de pixels do plano de fundo de uma imagem preto e branca. Os pixels que serão preenchidos são aqueles que são alcançados pelas bordas ou pelos "buracos" escolhidos aleatoriamente na imagem com o parâmetro "*holes*".



(a)

Figura 3 - (a)Imagem com quadrados de resposta preenchidos.

3.3 Bwareaopen:

Visto que o resultado da função citada acima ainda possui ruídos, como os resquícios das letras e foram parcialmente atenuados, utilizamos a função *Bwareaopen* que remove componentes conexos de uma imagem que tenham área menor que o parâmetro P em pixels, ou seja, deixa a imagem mais limpa.

3.3 Regionprops:

Para este trabalho estamos interessados em encontrar quadrados que cerquem as áreas de interesse, no caso os quadrados de resposta e de tipo de prova, então utilizamos a propriedade "*boundingBox*" da função *regionprops* que retorna uma lista com as posições x e y e as larguras em x e y dos quadrados que fazem o contorno dos objetos na imagem. Assim como resultado temos uma estrutura com as informações essenciais de todos os quadrados de resposta.

Com as informações adquiridas acima, desenvolvemos uma sequência interações que percorrem a imagem Original, aplicando um BW localmente nos cortes obtidos a partir dos parâmetros da função *regionprops*, pois precisaremos diferenciar os quadrados vazios dos preenchidos, passando por cada uma das posições de quadrado adquiridas e calculando a média de pixels "pretos" desse quadrado, se ela for maior que um valor que determinamos para um quadrado "Corretamente" preenchido, então inserimos em uma matriz resposta o valor 1 na posição referente aquele quadrado. Vale ressaltar que respostas duplicadas são desconsideradas na folha de resposta, assim não é possível burlar o gabarito.

Repetimos o processamento e aquisição de respostas com a folha de Gabarito, que tem suas respostas armazenadas em uma matriz Gabarito, e ao final do código realizamos a comparação das linhas da matriz, somando a quantidade de acertos.

4. Resultados e conclusões

Obtivemos bons resultados com o código desenvolvido, para fotos dentro dos parâmetros criados, ou seja, fotos com boa iluminação, adquiridas com o *camScanner*, planificada e com a angulação correta, o algoritmo mostrou 100% de eficiência.

Para fins de testes, realizamos dois testes fora dos padrões, um onde a imagem foi adquirida pelo App porém com a angulação incorreta, ou seja "torta", e o código errou apenas em 2 verificações, marcando 94,1% de precisão, e outro o processamento de uma prova sem planificação, sem utilizar o aplicativo *camScanner*, apenas cortando a folha da maneira correta e ainda assim o resultado foi aceitável, de uma prova com 34 questões marcadas corretamente, o algoritmo reconheceu 28 como corretas, ou seja, 82,4% de precisão para um caso extremamente fora dos padrões.

Referências

[1] **A Theory Based on Conversion of RGB image to Gray image.** Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.4410&rep=rep1&type=pdf.

[2] **Schwartz, H.P.W. R.** ***Análise de imagens digitais: princípios, algoritmos e aplicações*****.** Editora: Cengage Learning Brasil, 2007. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522128365/. Acesso em: 22 Nov 2020.