Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Eletrônica – DAELN
Engenharia de Computação
Disciplina: IF69D – Processamento Digital de Imagens
Semestre: 2020/2 APNP
Prof.: Gustavo B. Borba

RELATÓRIO
Optical Mark Recognition (Correção automática de provas)

Alunos:
Felipe Augusto Stark / 1683950
Ricardo Wellington Baldon / código

05.2021

1. Objetivo

O objetivo deste trabalho é reproduzir e implementar um algoritmo denominado OMR (Optical Mark Recognition), que realiza a correção automática de provas de múltipla escolha através do processamento de sua folha de resposta.

2. Fundamentação Teórica

Optical Mark Recognition é a denominação do processo de captura de dados "marcados" por humanos, como documentos, formulários e provas. Processo muito utilizado em correção de provas [1].

O OMR consiste em capturar uma imagem de um papel, e através do seu processamento obter as marcas realizadas neste, entendendo seu significado. A análise e o processamento da imagem se dão por meio de método de processamento digital de imagens, que auxilia na interpretação dos dados coletados digitalmente.

Para aplicar um método de OMR aplicado a correção de uma prova, algumas etapas devem ser seguidas, estão estas descritas nas subseções a seguir.

2.1. Elaboração da folha de respostas

A folha de resposta possui um conjunto de caixas retangulares e circulares que correspondem a um conjunto de perguntas e as alternativas. Estas caixas seguem um padrão de dimensões para que possa ser mais facilmente processado pelo algoritmo.

2.2. Leitura e processamento

A leitura engloba o processo de reconhecimento da imagem, para isso o OMR considera a quantidade de pixels nulos e não nulos da imagem.

Para fazer uma leitura precisa, é necessário realizar um pré-processamento da imagem, que consiste em realizar a conversão em escala de cinza da imagem, aplicar uma transformação Gaussiana. Este processo prepara a imagem para sua segmentação.

A segmentação consiste em separar a imagem em múltiplas regiões, analisando os objetos em os conjuntos de pixels de cada região separadamente. No caso deste projeto, foi realizada uma segmentação com o método de Threshold, que determina uma separação para a intensidade dos pixels da imagem.

Após a segmentação, pode-se de fato aplicar o processamento que reconhecerá as marcas na imagem, para este trabalho utilizou-se um método de comparação de pixels não nulos, descrito na seção subsequente.

3. Implementação

O funcionamento do OMR consiste no seguinte processo:

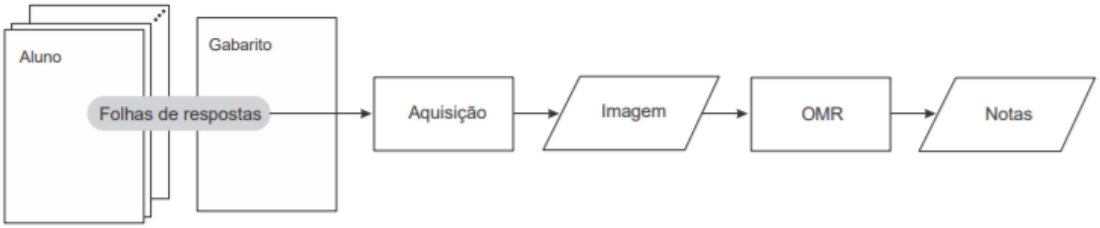


Figura 1 – Fluxo do projeto
(Fonte: Especificações do projeto).

A aquisição da imagem consiste em com um smartphone realizar a digitalização da folha de respostas, com a imagem obtida, pode-se fazer o processamento da imagem, aplicando os conceitos abordados no referencial teórico afim de calcular uma nota com base na folha de respostas do gabarito e do aluno.

Abaixo temos o modelo de folha de respostas utilizado:

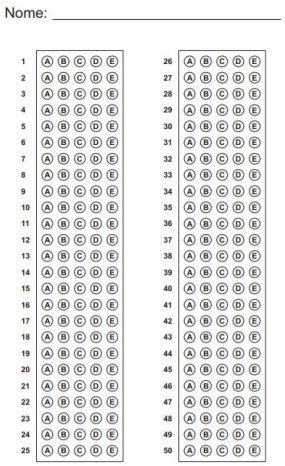


Figura 2 – Modelo de folha de respostas utilizado

3.1. Descrição inicial do código

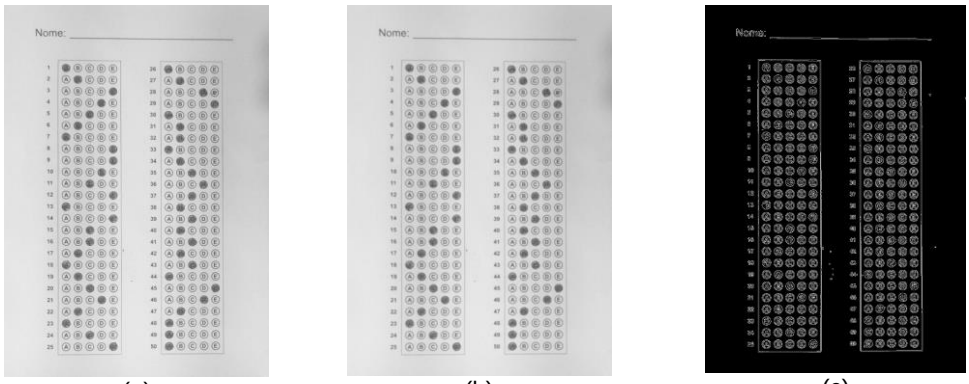
O código foi implementado em linguagem Python, utilizando as bibliotecas NumPy de processamento de dados e a biblioteca OpenCV de processamento de imagem. Além das bibliotecas da linguagem Python, utilizou-se métodos inspirados no algoritmo de OMR de Murtaza's Workshop, disponível na referência [3].

A arquitetura do projeto está dividida em um arquivo *main.py*, que consiste nas chamadas das funções e exibição das notas. Além disso, existe o arquivo *preprocessamento.py* contendo as operações de pré-processamento e preparação da imagem, no arquivo processamento temos as funções que compões o algoritmo do OMR, e no arquivo *calculos_notas.py* existem os recursos para cálculo e validação das notas.

O código, bem como as especificações de projeto estarão disponíveis no GitHub no repositório público de link: https://github.com/akastark/OMR.

3.2. Pré processamento da imagem

O pré-processamento da imagem consiste na transformação de uma imagem colorida em uma imagem composta por uma escala de cinza. Feito isso, a imagem passa por uma função Gaussiana, com o intuito de reduzir o ruído, uma vez sem ruído, aplica-se o algoritmo de Canny, para ressaltar os contornos da imagem.



(a) (b) (c)

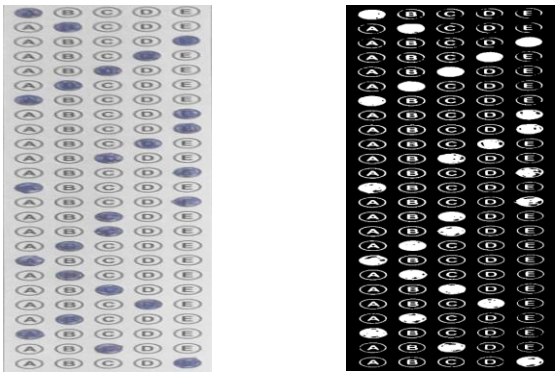
Figura 3 – (a) Imagem em escala de cinza. (b) Imagem com filtro Gaussiano. (c) Imagem após algoritmo de Canny.

3.2. Processamento e coleta de respostas

Uma vez que a imagem se encontra pré-processada, podemos aplicar o processamento que realizará o reconhecimento das questões assinaladas. Iniciando estas ações com a localização dos contornos da imagem através da OpenCV, localizados os contornos, é necessário identificar quais deste são retângulos, para isso foi implementada uma função que localiza os retângulos através da sua área de contorno e sua quantidade de retas no perímetro.

No caso da folha de respostas utilizadas, teremos dois retângulos, cada um deles delimita as alternativas de 25 questões. Localizados os retângulos, precisamos encontrar seus vértices para indicar as coordenadas de recorte da imagem, isso é feito por uma função implementada que localiza os vértices através da aproximação do perímetro.

No momento em que temos as coordenadas, precisamos fazer o recorte dos retângulos na imagem, para isso, criamos uma matriz de transformação, a fim de aplicar uma máscara de planificação na imagem, deixando o recorte mais preciso. Em cada um dos recortes, aplica-se a segmentação pro *Threshold,* definindo pixels entre branco e preto, ou seja, nulos e não nulos.



(a) (b)

Figura 4 – (a) Imagem recortada (b) Imagem com Threshold.

Neste ponto, as respostas estão prontas para serem segmentadas e computadas, então, para cada um dos retângulos (25 questões), a imagem é recortada em 25 linhas e 5 colunas, onde para cada coluna em cada linha é analisado o número de pixels não nulos da imagem, onde o valor máximo de pixels não nulos em uma coluna de uma linha, indica uma alternativa assinalada, então é computado o índice desta coluna e inserido em um vetor de 50 índices, correspondente a cada questão da folha de respostas.

(a) (b)

Figura 5 – (a) Alternativa assinalada (b) Alternativa não assinalada.

3.2. Tratamento de erros

Alguns erros podem ocorrer no preenchimento do gabarito ou da folha de respostas do aluno, como questões em branco ou questões duplamente preenchidas.

No caso de uma questão onde a alternativa com maior número de pixels não nulos tiver este número ao menos 50% maior que a alternativa com menor número de pixels não nulos, esta é uma questão válida, caso contrário considera-se que não foi preenchida. No caso de um gabarito, a questão é anulada, no caso de folha de resposta do aluno, a questão é considerada errada.

Para tratar o preenchimento duplo, verifica-se a alternativa com o maior número de pixels não nulos divindo-se pela segunda maior alternativa em número de pixels não nulos gera um quociente maior que 0.8, em caso positivo, considera-se apenas uma alternativa com preenchimento, a maior. Caso contrário, duas alternativas estão preenchidas e a questão é anulada no gabarito e considera errada em folha de respostas do aluno.

3.3. Cálculo de nota

Para calcular a nota final da prova, comparamos índice a índice dois vetores, o gabarito da prova e as respostas do aluno, caso sejam iguais, atribui-se uma nota equivalente a questão. Para os casos onde uma questão não se encontra assinalada no gabarito, desconsideramos da correção e a questão é anulada, caso não esteja assinala na prova do aluno, a questão é considerada como errada.

Caso o número máximo de pixels não nulos seja menor que a 150% da coluna com o mínimo de pixels não nulos em uma questão, significa que esta não teve alternativa assinalada, tornando-se uma questão nula.

4. Resultados e conclusões

O algoritmo cumpriu seu objetivo de processar uma imagem da folha de respostas e comparar com a de um aluno, gerando uma nota com base nas comparações.

Além disso, o algoritmo também consegue responder de forma positiva a pequenos erros como: ruído, tratar questões em branco e planificar imagens levemente tortas.

É interessante considerar melhorias futuras como a otimização do reconhecimento da folha de respostas, além um reconhecimento de um padrão de folha genérico.

Referências

[1] Optical mark recognition. In: Wikipedia, the free encyclopedia. Disponível em:. <https://en.wikipedia.org/wiki/Optical_mark_recognition>. Acesso em: 16/05/2021.

[2] Segmentação,processamento de imagem. In: Wikipedia, a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Segmentação_(processamento_de_imagem)>. Acesso em: 16/05/2021.

[3] OPTICAL MARK RECOGNITION (OMR) MCQ Automated Grading. In: Murtaza`s Workshop – Robotics and AI. Disponível em:. <https://www.murtazahassan.com/opencv-projects/>. Acesso em: 02/05/2021.

ANEXO I – Fluxograma simplificado do algoritmo