Processamento de Cartão Resposta Utilizando Linguagem Python

Trabalho realizado para a matéria de Computação Gráfica do Curso de Ciência da Computação

1st Joana Shizu
Universidade Estadual do Norte do Paraná
Bandeirantes, Brasil
camargojoana09@gmail.com

2nd Melissa Francielle
Universidade Estadual do Norte do Paraná
Bandeirantes, Brasil
melfrancy08@gmail.com

***Abstract*—This paper presents a Computer Vision algorithm designed to automate the correction of multiple-choice answer sheets. Developed in Python using OpenCV, the system processes high-resolution images captured via mobile devices to isolate and interpret candidates' marked responses across 60 questions, each with five possible answers. The methodology involves preprocessing steps such as grayscale conversion, binarization, and perspective rectification, follower by segmentation into question regions. The marked choice in each question is identified through pixel density analysis in thresholded regions. A dataset of four hundred and ninety nine images with lighting and distances was used to validate the system. The results demonstrate an accuracy, with an average processing time of seconds and memory consumption under per image. The approach proves effective and resource-efficient, making it suitable for large-scale assessments. Future work includes enhancing OCR performance and integrating deep learning for improved detection of ambiguous marks.**

***Index Terms*—CG, Visão Computacional, Processamento de Imagens, OCR, Python**

I. Introdução

A Visão Computacional tem se dado uma área propícia na automação de tarefas repetitivas e que podem facilmente apresentar erros humanos. Entre as diferentes formas de aplicação da visão computacional, destaca-se o uso da automatização de correção e identificação de cartões-respostas em vestibulares, concursos públicos, exames institucionais entre outras formas de processos avaliativos. Essas correções, quando realizadas de forma manual podem gerar cansaço físico, mental, sendo sujeita a falhas e demandando tempo significativo do avaliador. Utilizando a biblioteca OpenCV, tornou-se mais fácil desenvolver soluções para esses tipos de problemas utilizando técnicas de processamento de imagens para detectar e interpretação dos cartões.

Este trabalho propõe o desenvolvimento de um sistema, escrito em linguagem Python, que identifica e isola regiões relevantes de cartões-respostas digitalizados, segmentando as áreas necessárias e interpretar as alternativas marcadas pelos candidatos. A base de dados utilizada é composta por quatrocentos e noventa e nove imagens, digitalizadas de diferentes ilumuniações, posicionamento, simulando cenários reais de uso.

Os resultados indicam que o sistema podem apresentar um desempenho razoável tanto em termos de acurácia quanto tempo de processamento, podendo ser uma alternativa para automatização de correções com eficiência e baixo custo. Este artigo descreve o protocolo adotado, os métodos e análise utilizada, além dos resultados obtidos.

II. Problema

Como a correção manual de cartões-respostas podem ser cansativas e podem demandar tempo, é necessário então usar um sistema que seja capaz de lidar com um conjunto questões computacionais e visuais. Nesse contexto, há um conjunto de dados composto por quatrocentos e noventa e nove imagens de cartões digitalizados, cada um contendo sessenta questões com cinco alternativas.

O principal problema a ser resolvido é a identificação das marcações feitas pelos candidatos, contendo variabilidade de iluminação, constrate da imagem, rotação e posicionamento do cartão, preenchimento incompleto, ou em excesso, ou a falta deles, ruídos, manchas ou outras quaisquer interferências possíveis das imagens. Assim sendo necessário garantir que o sistema consiga, localizar as áreas de marcação, segmentar a imagem e as regiões necessárias, interpretar e comparar as respostas do gabarito para as métricas de análise. Trantando-se então de um clássico problema de processamento, segmentação e interpretação marcação visual, que exige uma combinação de técnicas de visão computacional e tratamento de imagens.

III. Hipótese

A hipótese deste trabalho é que os cartões-resposta podem ter uma análise automatizada utilizando algoritmos de visão computacional em linguagem Python. Por meio do uso de bibliotecas especificas para área de processamento de imagens, como OpenCV e Pytesseract, sendo possível identificar e interpretar as marcações.

A validação da hipótese será feita a partir de comparação dos resultados gerados pelo algoritmo após a análise e um arquivo já adquirido em formato CSV contendo as respostas corretas.

É esperado que o sistema consiga atingir um nível aceitavel de acurácia ao detectar as alternativas, além de ser esperado um desempenho razoável voltados a tempo de processamento e uso de memória.

A. Hipótese de Processamento

Acredita-se que para as técnicas de pré-processamento, como redimensionamento, aplicação de filtros possa ser fatores importantes para a eficácia do sistema. Ao padronizar o tratamento das imagens utilizando um protocolo, espera-se que o algoritmo consiga garantir maior precisão e tratamento das imagens. Esse processo é essencial para garantir a confiabilidade das análises.

IV. Método

A metodologia escolhida neste trabalho baseia-se no seguimento de um protocolo que padronize a forma de tratamento das imagens dos cartões. Esse protocolo tem como objetivo garantir que as imagens passem por etapas de pré-processamento e processamento, permitindo a identificação das alternativas Essa padronização é essencial para assegurar a consistência e a eficácia do algoritmo em diferentes condições.

A. Protocolo

O protocolo desenvolvido para este trabalho tem como principal objetivo padronizar o processo de pré-processamento das imagens, principalmente na etapa de interpretação dos dados e análise das alternativas detectadas. Para isso, foram aplicadas técnicas como redimensionamento da imagem, aplicação dos filtros e recorte de determinados pontos, que serão detalhados posteriormente. O pré-processamento é fundamental para facilitar a identificação do ROIs, das colunas e blocos de questões, para que a leitura seja mais precisa por meio do OCR. O protocolo está estruturado com a seguintes fases: aquisição, pré-processamento, segmentação e interpretação das informações extraídas das imagens. Todos os códigos citados durante o trabalho podem ser encontrados no link em questão https://github.com/Melissa-Francielle/Processamento_Cartao_Resposta/tree/main

B. Aquisição da Imagem

Esta fase consiste na obtenção das imagens dos cartões de resposta a serem analisados pelos sistema. Os quatrocentos e noventa e nove cartões de resposta foram digitalizados utilizando scanner com resolução de 300 DPI, snedo aplicado a binarização automatica das imagens, resultando em arquivos JPEG preto e branco prontos para o processamneto. Cada imagem está associada a um registro no arquivo CSV contendo o gabarito de respostas, utilizado para validar a extração das próximas etapas.

C. Pré-processamento

Esta etapa tem como objetivo preparar as imagens adquiridas para as fases subsenquentes melhorando sua qualidade e estrutura para facilitar a análise.

- **Conversão de cor limiarização:** as imagens são convertidas para escalas de cinza e, passa pela binarização de Otsu, utilizando o parâmetro `cv2.THRESH_BINARY_INV + Otsu`
- **Fechamento morfológico:** após a binarização, aplica-se um kernel 3×3 para eliminar ruídos, falhas e conectar contornos incompletos, melhorando assim os elementos gráficos das imagens.
- **Detecção de Marcadores:** triângulos escuros posicionados nos cantos dos cartões são utilizados como marcadores. A função `_localizar_triangulos()` identifica seus vértices e, caso os quatros sejam localizados, calcula-se a matriz de transformação e corrige a perspectiva das imagens.

D. Segmentação

Na fase de segmentação, as imagens pré-processadas são analisadas com o objetivo de isolar as regiões relevantes, neste caso, as linhas contendo apenas as questões indiivduais.

- **Divisão em Colunas:** O cartão de resposta é dividido em três colunas verticais durante o processamento, respeitando uma margem lateral definida pelo parâmetro `MARGEM_COL_FRAC`, o que garante a segmentação das colunas de forma eficaz para as áreas de marcação.
- **Detecção de linhas de questões:** EM cada coluna, os contornos com formato aproximado de quadrados são identificados por meio da função `_square_contours`. Esses contornos são agrupados verticalmente, formando linhas correspondentes às questões. Regiões como cabeçalho e rodapé - que geralmente apresentam rótulos como **A B C D E** - são descartadas por não fazerem parte das respostas.
- **Recorte individual:** Após a identificação das linhas, cada linha de questão é recortada e salva como um arquivo de imagem nomeado como `Qxx.png`, totalizando até sessenta imagens por cartão de resposta.

E. Interpretação dos dados

A fase de intepretação é responsável pela analse dos recortes obtidos na etapa da segmentação e extrair a partir disso as alternativas assinaladas em cada questão. As operações realizadas nessa fase correspondem ao módulo de reconhecimento óptico de marcações.

- **Extração de marcações:** Primeiramente, para cada questão individual `Qxx.png`, calcula-se o preenchimento no interior (miolo) dos cincos quadrados correspondetes às alternativas. Caso o preenchimento em uma das regiões ultrapasse de 40% da área do quadrado, a alternativa equivalente `(A B C D E)` é considerada como marcada.
- **Identificação do Candidato:** A identificação dos candidatos são feitas por meio do nome do diretório em que suas imagens estão armazenadas. Cada pasta segue o padrão **010NNN01**, onde **NNN** representa o número identificador do candidato. A função `extrair_id_pasta`

realiza a extração desse número, possibilitando que as respostas sejam indexadas corretamente.

- **Comparação com Gabarito:** Por fim, as respostas extraidas são comparadas com o gabarito oficial, que está armazenado em um arquivo CSV. Essa comparação é executada utilizando um script `comparar_respostas.py`, que também registar o tempo de processamento e o pico de uso de memória por meio da biblioteca `tracemalloc`.

V. Experimentos

a) Conjunto de dados: Foram processadas 499 imagens com resoluções médias de 22000×34000 px. O gabarito oficial o soma 23.868 comparações (499 candidatos $\times 60$ questões menos registros ausentes), armazenados em um arquivo CSV.

b) Aplicação: O sistema proposto foi aplicado ao conjunto de imagens de cartões de resposta, com objetivo de avaliar a capacidade de reconhecer e extrair as alternativas marcadas nos cartões. Durante a execução, cada imagem foi submetida a fases definidas no protocolo, os resultados foram comparados com o arquivo CSV de referência.

Além de realizar a extração das alternativas assinaladas, o sistema estabelece os dados para avaliação. Essa aplicação permitiu medir desempenho do método aplicado em termos de acurácia, uso de memória, validando a eficácia da abordagem apresentada.

c) Métricas de desempenho: A acurácia global foi definida a partir da razão entre o número de respostas extraídas corretamente pelo sistema e o total de comparações realizadas, conforme expressa a Equação (1)

$$\text{Acurácia} = \frac{\text{Respostas idênticas}}{\text{Comparações realizadas}} \times 100; \qquad (1)$$

Além do cálculo realizado também foi incluído a análise de outras métricas de avaliação do sistema: A execução completa

TABLE I
Resumo de desempenho do sistema

Métrica	**Valor**
Avaliação da acurácia	
Comparações realizadas	23 868
Respostas idênticas	18 851
Acurácia global	78.98 %
Medições de desempenho	
Tempo de processamento	0.490 s
Pico de memória	8.81 MB

de todos os cartões consumiu **0,490 segundos** e atingiu um pico de **8,81MB** de uso de memória, demonstrando a leveza do sistema para uso em ambientes desktops convencionais.

VI. Discussão e Conclusão

A acurácia obtida de **78,98%** revela que, embora o sistema tenha alcançado um desempenho satisfatório, ainda há margem para melhorias, especialmente na etapa de OMR, onde falhas de detecção ou ruídos podem resultar em falsos negativos. Apesar das limitações, o sistema demonstrou um funcionamento eficiente: o protocolo aplicado aos cartões de resposta resultou um tempo médio de processamento e consumo de memória reduzido. O uso das bibliotecas em Python, foram essenciais para garantir um desempenho e rápida implementação.

Há potênciais melhorias futuras que podem aumentar a acurácia do sistema sem comprometer a eficiência, tornando a abordagem mais viável.

References

[1] OpenCV Documentation. [Online]. Available: https://docs.opencv.org/

[2] Pytesseract: Python-tesseract OCR tool. [Online]. Available: https://pypi.org/project/pytesseract/

[3] Automated Grading using Optical Mark Recognition (OMR) – MCQ Detection. YouTube. [Online]. Available: https://youtu.be/0IqCOPlGBTs?si=gvbvm6HeY9uLEd3u