Universidade Tecnológica Federal do Paraná – UTFPR – Campus Curitiba
Departamento Acadêmico de Eletrônica – DAELN
Curso de Engenharia Eletrônica
Disciplina: IF69D e OPEET014 – Processamento Digital de Imagens
Semestre: 2025.2
Prof: Gustavo B. Borba

RELATÓRIO
Reconhecimento Ótico de Marcas

Alunos:
Ariane Chiminazzo / 2462281
Matheus Salvi / 1815121

11.2025

1. Objetivo

O objetivo deste trabalho é implementar em MATLAB um sistema de Reconhecimento Ótico de Marcas (Optical Mark Recognition – OMR) para correção automática de folhas de respostas de provas de múltipla escolha. O sistema recebe imagens de folhas-resposta capturadas por smartphone, realiza correção geométrica, detecta as bolhas preenchidas do tipo de prova (A, B, C ou D) e das 50 questões, compara essas respostas com um gabarito previamente definido em arquivo (gabaritos.mat) e calcula o número de acertos de cada prova.

2. Fundamentação Teórica

O Reconhecimento Óptico de Marcas (OMR) é uma técnica de processamento digital de imagens destinada à detecção de marcas em posições pré-definidas de um formulário [1]. Essas marcas são produzidas pelo usuário (caneta ou lápis) e aparecem como regiões escuras sobre um fundo claro. Aplicações típicas incluem correção de provas, enquetes e votações [1].

De modo geral, um sistema OMR envolve: aquisição da imagem, pré-processamento (correção geométrica e de iluminação), segmentação das regiões de interesse (bolhas), detecção das marcas preenchidas e interpretação dos resultados, associando cada marca a uma questão e alternativa [1],[2].

Neste trabalho, as folhas-resposta são geradas artificialmente em MATLAB, com layout conhecido: página A4 de 2480 x 3508 px contendo quatro quadrados pretos nos cantos, bolhas para o tipo de prova e 50 questões com cinco alternativas em duas colunas.

2.1 Aquisição e conversão para níveis de cinza

As imagens das folhas são capturadas por smartphone em RGB e lidas com imread. A conversão para níveis de cinza é realizada pela função rgb2gray, que usa a combinação ponderada dos canais [1]:

$$0,299 * R + 0,587 * G + 0,114 * B$$

Em seguida, a imagem é normalizada para o intervalo [0,1] com mat2gray.

2.2 Estimativa de escala e detecção dos marcadores de canto

A foto possui altura H_f e largura W_f. O layout original da folha tem altura $H_o = 3508$ e largura $W_o = 2480$. No código, a escala é estimada por

$$escalaX = \frac{Wf}{Wo},\ escalaY = \frac{Hf}{Ho}\ e\ escala = \frac{escalaX+escalaY}{2}$$

Sabendo que o lado nominal dos quadrados de canto é $L_0 = 80$ px, estima-se o lado esperado na foto como

$$L_{est} = L_0 * escala$$

As áreas mínima e máxima aceitáveis para os quadrados são

$$A_{min} = (0,5 * L_{est})^2,\ A_{max} = (2 * L_{est})^2$$

A imagem é binarizada com o limiar global de Otsu [3], obtido por graythresh. Se T é o limiar, o código utiliza um limiar mais rigoroso 0,7 * T:

$$BW(x,y) = \begin{cases} 1, & In(x,y) < 0,7 * T \\ 0, & caso\ contrário \end{cases}$$

Componentes conectados são analisados por regionprops, que fornece área e retângulo delimitador. A razão de aspecto a = w / h é usada para selecionar elementos aproximadamente quadrados, com $0,7 \leq a \leq 1,30$. Os quatro maiores candidatos (em área) são considerados marcadores de canto.

As coordenadas desses marcadores na foto (x, y) são associadas às coordenadas conhecidas na folha original (x′, y′). A relação entre elas é modelada como transformação projetiva [1],[2]:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad x_c = \frac{x'}{w}, \quad y_c = \frac{y'}{w}$$

Onde H é uma matriz 3 x 3 estimada por fitgeotrans, e (x_c, y_c) são as coordenadas corrigidas na folha retificada. A aplicação da transformação a todos os pixels é feita por imwarp.

2.3 Detecção de bolhas e limiar adaptativo

Depois da retificação, a folha está na resolução original ($H \approx H_o$, $W \approx W_o$). O raio nominal das bolhas é $r_o = 25$ px. Para uma imagem com altura H, o raio estimado é

$$r_{est} = r_o * \frac{H}{H_o}$$

Procura-se por círculos com raio no intervalo

$$r_{min} = 0,85 * r_{est},\ r_{max} = 1,15 * r_{est}$$

A detecção é feita com a transformada de Hough para círculos, implementada pela função imfindcircles [1],[4].

Para cada círculo detectado, define-se uma máscara circular interna de raio $r_{int} = \alpha r$, com $\alpha = 0,45$ no código:

$$\text{mask}(x,y) = \begin{cases} 1, & \text{se } x^2 + y^2 \leq r_{\text{int}}^2, \\ 0, & \text{caso contrário.} \end{cases}$$

A intensidade média interna da bolha é

$$I_{\text{bolha}} = \frac{1}{N_{\text{mask}}} \sum_{(x,y)\in \text{mask}} G(x,y)$$

Onde G é a imagem filtrada em tons de cinza e N_{mask} é o número de pixels com máscara 1. Para "combater" sombras, calcula-se também a média de uma janela quadrada maior ao redor da bolha:

$$I_{\text{local}} = \frac{1}{N_{\text{janela}}} \sum_{(x,y)\in \text{janela}} G(x,y)$$

O limiar adaptativo é definido como

$$t_{adpt} = max(Ilocal - 0,18,\ 0,15)$$

A bolha é classificada como preenchida se

$$I_{bolha} < t_{adpt}$$

Alternativamente, o preenchimento pode ser visto como uma razão entre pixels escuros (N_{esc} e o total da região (N_{tot}) [1]:

$$r = \frac{N_{esc}}{N_{tot}}$$

Nesse caso, uma bolha seria considerada preenchida se $r > r_{th}$. Embora o código utilize a abordagem baseada em intensidade média e limiar adaptativo, o conceito de razão de preenchimento permanece útil para interpretar o método.

2.4 Mapeamento das bolhas em questões e gabarito

Conhecendo-se as posições base das colunas X_1 e X_2, a coordenada horizontal de cada bolha preenchida permite decidir em qual coluna ela está:

$$\text{col} = \arg\min_{j \in \{1,2\}} |c_x - X_j|$$

Onde (c_x, c_y) é o centro da bolha. A linha da questão é obtida pela diferença vertical em relação à primeira linha de questões Y_0:

$$\text{linha} = \text{round}\left(\frac{c_y - Y_0}{\Delta Y}\right) + 1$$

Em que ΔY é o espaçamento vertical entre linhas. A alternativa (1 a 5) é dada por

$$\text{alt} = \text{round}\left(\frac{c_x - X_{\text{col}}}{\Delta X_{\text{alt}}}\right) + 1$$

Com ΔX_{alt} representando o espaçamento horizontal entre bolhas. O número global da questão é

$$q = (col - 1) * 25 + linha,\ 1 \leq q \leq 50$$

Dessa forma, constrói-se um vetor de respostas R(q), q = 1,…,50, contendo a alternativa marcada em cada questão.

Os gabaritos para cada tipo de prova (A, B, C, D) são armazenados numa estrutura gabaritos, em que cada campo (gabaritos.A, gabaritos.B etc.) é um vetor de 50 inteiros entre 1 e 5. A correção é feita comparando, elemento a elemento, o vetor de respostas R(q) com o gabarito G(q).

---

3. Implementação

A implementação foi construída combinando scripts e funções modulares:

- criar_gabaritos.m: gera aleatoriamente gabaritos para as provas A, B, C e D e salva em gabaritos.mat.

- gera_folha_resposta_e_gabarito.m: produz uma imagem em branco no tamanho A4, desenha marcadores de canto, cabeçalho, bolhas do tipo de prova e a grade das 50 questões; opcionalmente preenche as bolhas corretas conforme o vetor de gabarito.

- gera_multiplos_gabaritos.m: chama a função anterior para todos os tipos de prova e salva as folhas geradas em uma pasta de saída.

- ler_folha_resposta_por_foto.m: lê a foto, converte para tons de cinza, normaliza, estima escalas, projeta o tamanho esperado dos marcadores, binariza com Otsu, seleciona quadrados dentro da faixa de área e razão de aspecto, estima a transformação projetiva e gera a folha retificada. Em seguida chama ler_folha_resposta_retificada.

- ler_folha_resposta_retificada.m: executa o pré-processamento adicional, estimativa de raio das bolhas, detecção de círculos por imfindcircles, classificação das bolhas com limiar adaptativo, separação entre bolhas de tipo de prova e de questões, e mapeamento das bolhas de questões em índices de coluna, linha e alternativa, retornando o tipo de prova e o vetor de respostas.

- conta_circulos_preenchidos.m: função auxiliar usada para testes, que contabiliza quantas bolhas foram detectadas e quantas foram classificadas como preenchidas, exibindo os resultados graficamente.

- corrigir_provas.m: script principal que adiciona o caminho da pasta Metodos_Auxiliares, carrega gabaritos.mat, lê as imagens de FR1.png a FR5.png, chama ler_folha_resposta_por_foto para cada uma e calcula o número de acertos por meio da soma da expressão lógica.

O usuário pode, portanto, gerar os gabaritos, gerar e imprimir as folhas, aplicar a prova, fotografar as folhas preenchidas, salvar as imagens na pasta Folhas_Resposta e executar corrigir_provas.m no MATLAB.

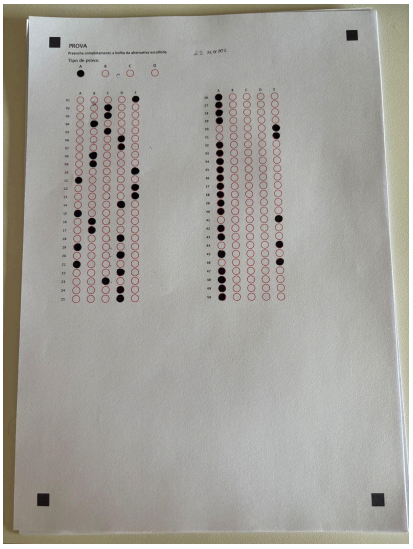
---

4. Resultados e conclusões

Foram realizados testes com cinco folhas-resposta preenchidas, armazenadas como FR1.png a FR5.png. Em situações em que os quadrados de canto estavam visíveis e as bolhas apresentavam contraste adequado, o sistema conseguiu detectar corretamente os marcadores, registrar a folha, identificar o tipo de prova e contabilizar o número de acertos conforme o gabarito correspondente.

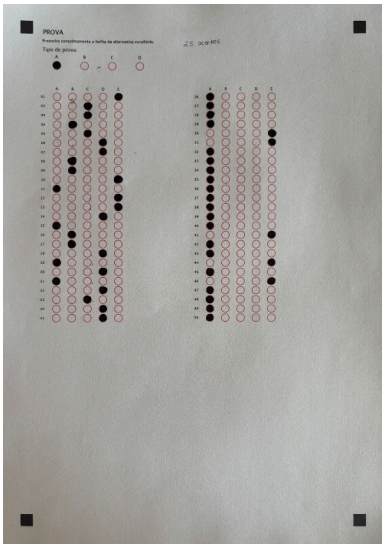
Na Figura 1 pretende-se ilustrar uma folha capturada por smartphone e, na Figura 2, a folha após o registro projetivo. Em outra figura (Figura 3) podem ser apresentados os círculos detectados (em verde) e as bolhas classificadas como preenchidas (em vermelho), conforme os resultados da função ler_folha_resposta_retificada.

Figura 1 – Imagem obtida por foto com smartphone.



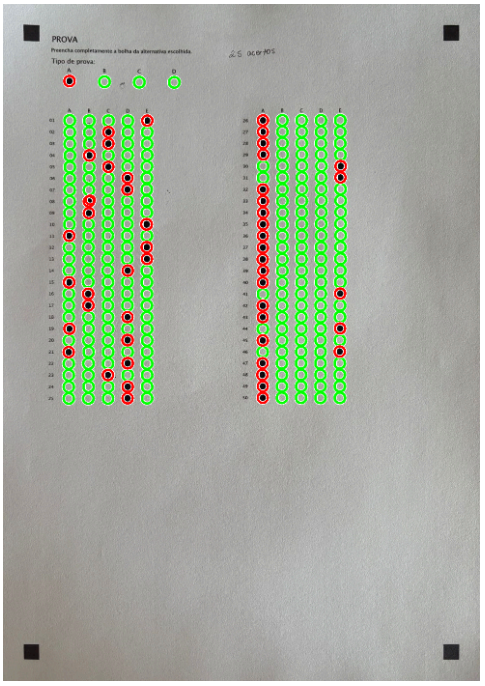
Fonte: Autores, 2025.

Figura 2 – Imagem retificada



Fonte: Autores, 2025.

Figura 3 – Bolhas detectadas e diferenciadas entre preenchidas e não preenchidas.



Fonte: Autores, 2025.

Os testes indicam que o método é robusto a rotações e variações de escala graças à etapa de correção geométrica baseada em transformação projetiva. A classificação adaptativa de bolhas mostrou-se eficiente para compensar variações moderadas de iluminação.

As principais limitações observadas estão relacionadas a fotos com perspectivas muito distintas, pois podem prejudicar os limiares adaptativos, e a casos em que um ou mais marcadores de canto não aparecem totalmente na imagem, tornando a estimação da matriz H instável.

Como trabalhos futuros, sugerem-se: (i) aprimorar a detecção de marcadores de canto com métodos baseados em correlação de templates ou detecção de cantos de Harris [1]; (ii) refinar o cálculo do limiar adaptativo usando estatísticas robustas (por exemplo, mediana em vez de média); (iii) tratar explicitamente questões com múltiplas bolhas preenchidas (anulação da questão ou marcação como erro); e (iv) desenvolver uma interface gráfica simples para facilitar o uso do sistema por usuários não familiarizados com o ambiente MATLAB.

Referências

[1] GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3rd ed. Upper Saddle River: Pearson, 2008.

[2] MATHWORKS. **Image Processing Toolbox** – User's Guide. Natick, MA: The MathWorks Inc., 2025.

[3] OTSU, N. "**A Threshold Selection Method from Gray-Level Histograms.**" IEEE Transactions on Systems, Man, and Cybernetics, v. 9, n. 1, p. 62–66, 1979.

[4] DUDA, R. O.; HART, P. E. "**Use of the Hough Transformation to Detect Lines and Curves in Pictures.**" Communications of the ACM, v. 15, n. 1, p. 11–15, 1972.