Visão por Computador

Projeto "Exams evaluation using computer vision"

André Salgueiro, 50645 & Filipe Costa, 65092

***Resumo* - Este projeto trata da avaliação de uma folha de exame, procurando a tabela que contém as respostas do aluno, analisando as mesmas e, tendo em conta a solução do exame, corrigindo-as.**

***Abstract* - This project deals with the evaluation of an exam sheet, looking for the table containing the student's answers, analyzing them and, taking into account the solution of the exam, correcting them.**

I. INTRODUÇÃO

O temas deste projeto é "Exams evaluation using computer vision", ou seja, a avaliação de exames usando visão por computador. Para tal utilizamos métodos e máscaras de modo a encontrar a tabela que contém as respostas dos alunos e através do algoritmo conseguir saber qual a resposta a cada questão comparando-a com a solução do exame e verificando se está correta ou errada.

II. O ALGORITMO

A. Introdução de Dados

Logo no início da execução do programa é pedido ao utilizador que insira informação sobre a tabela que será analisada, são necessários os seguintes dados:

- o nome do ficheiro da imagem a analisar;
- o número mecanográfico do aluno;
- a quantidade de colunas da tabela;
- a quantidade de linhas da tabela;
- a quantidade de perguntas;
- a quantidade de respostas possíveis, ou seja, o número de opções das quais o aluno tem que escolher uma;
- as cotações respectivas de resposta certa ou errada;
- o nome do ficheiro que contém a solução do exame.

O ficheiro que contém a solução do exame aceita dois formatos, um que contém apenas a letra correspondente à resposta correta por linha e o outro formato aceita por linha o número da resposta, seguido de espaço, seguido da resposta. Em ambos os casos as respostas devem estar ordenadas numericamente.

No final do programa é dado ao utilizador a opção de analisar um novo teste utilizando as mesmas configurações.

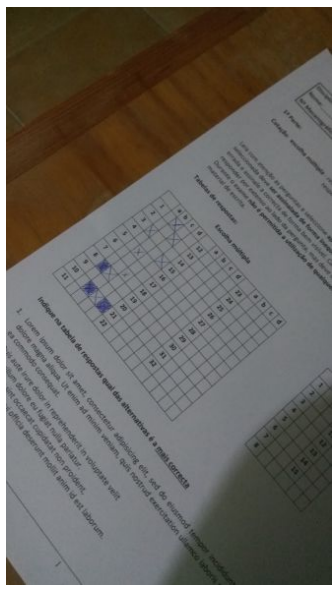

Fig. 1 - Imagem original.

B. Pré-processamento

A imagem é lida em tons de cinzentos pois não é necessária a componente de cor para a detecção. É feito um pré-processamento composto por um Gaussian Blur, seguido de um Adaptive Threshold invertido (linhas a branco e fundo a preto) e um dilate, onde o tamanho da dilatação varia consoante a resolução da imagem original.

C. Deteção da Tabela

Para a detecção da grelha usamos um algoritmo para encontrar o maior *BLOB* (Binary Large OBject). Depois, faz-se *fill* com cor branca a esse *BLOB* e todos os outros com cor preta para os esconder (Fig. 2).

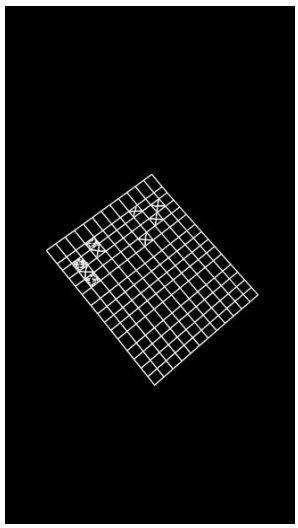

Fig. 2 - O maior BLOB.

D. Tratamento da Tabela

Depois de extrair uma imagem binária da grelha, procedemos à correcção da perspectiva, para isso aplicamos a função de *Find Contours* com o parâmetro *CV_RETR_EXTERNAL* para obter o contorno de fora da grelha. Com este contorno vamos encontrar um polígono que encaixe nele e um *bounding rectangle*.

Para corrigir a perspectiva é necessário mapear os 4 cantos do retângulo original nos 4 cantos do destino, no entanto a função *Find Contours* não consegue distinguir a orientação da imagem (o 1º canto é sempre o canto com menor valor de y) sendo necessário corrigir manualmente o mapeamento dos cantos. A nossa abordagem a este problema foi a seguinte, se o nº linhas for inferior ao nº de colunas, verifica-se se a distância entre o 1º e o 2º canto é superior à distância entre o 1º e o 4º canto, caso se verifique os cantos são rodados para a esquerda 2 posições. O mesmo acontece quando o nº de linhas é superior ao nº de colunas e a distância entre o 1º e o 2º canto é inferior à distância entre o 1º e o 4º canto. Com os cantos devidamente mapeados procede-se à transformação.

Com a perspectiva corrigida segue-se à detecção das interseções das linhas da grelha, isto é feito através do cálculo da máscara das linhas verticais e horizontais (visível na Fig. 3) obtidas através de operações morfológicas de erosão e dilatação, aplicando um *AND Bitwise* para apenas as interseções serem visíveis.

Para descobrir as coordenadas dos pontos das interseções procuramos o seu contorno e para filtrar eventuais contornos detectados por ruído estes são ordenados por área e apenas considerados os $n^o\ linhas+1 * n^o\ colunas+1$ contornos. Em seguida as interseções passam por dois processos de ordenação, o primeiro *top-to-bottom* seguido de *left-to-right,* para que as coordenadas sejam ordenadas a partir do canto superior esquerdo para o canto inferior direito.

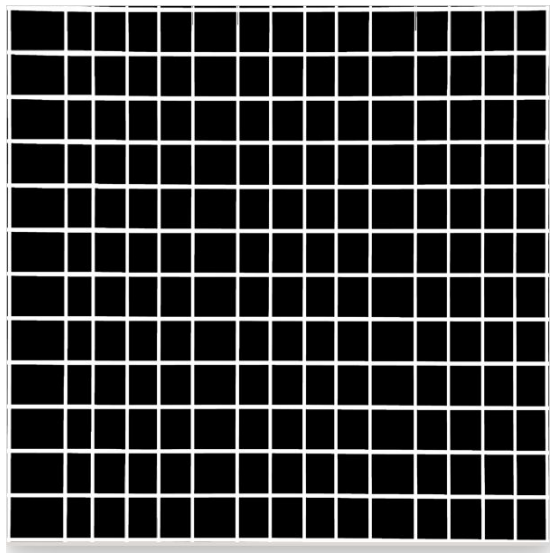

Fig. 3 - Detecção das linhas.

E. Tratamento das Células

Os resultados do pré-processamento feito inicialmente não foram suficientemente positivos para o tratamento das células, então é realizado um novo pré-processamento, neste usando um blockSize maior no *Adaptive Threshold* (*blocksize* = 7), eliminando um maior número de ruído no interior das células.

Usando as coordenadas obtidas pelos contornos das interseções é gerado um rectângulo identificando cada célula. Este retângulo sofre uma ligeira redução de maneira a não detectar as linhas da grelha.

Usando os retângulos é feito um *crop* à *Region of Interest* de modo a analisar cada célula individualmente (como se pode verificar na Fig. 5). Para detectar se a célula tem um X, é rasurada ou está vazia é feito uma soma da matriz da célula dividindo pela sua área, obtendo assim um rácio de preenchimento que é semelhante mesmo sobre diferentes resoluções de imagem.

Usando oito imagens diferentes de grelhas de resposta foram anotados os valores mínimos e máximos de cada caso e calculado um valor médio para o *threshold* para

indicar o respectivo caso. É criada uma tabela que representa a análise feita por parte do algoritmo de deteção com a seguinte nomenclatura:

- 0 - célula vazia;
- 1 - célula com resposta (cruz);
- 77 - célula quase cheia (rasurada) representa uma resposta que não deve ser tida em conta;
- 99 - célula que não pertence ao espaço de respostas.

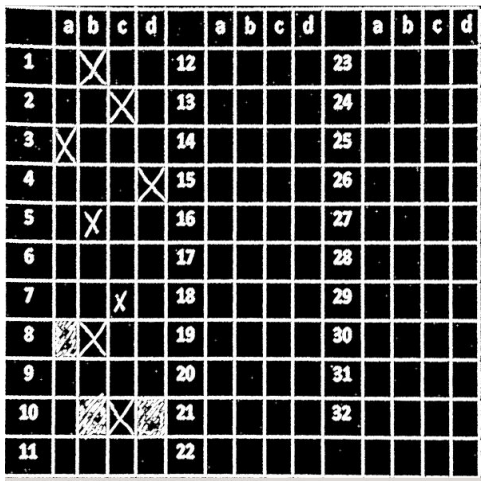

Fig. 4 - Região de interesse (Region of Interest, ROI).

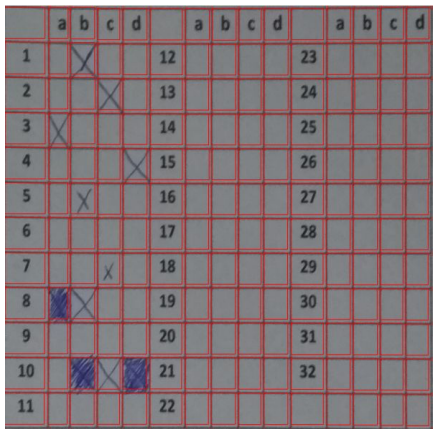

Fig. 5 - Perspectiva Corrigida e Separação de células.

F. Correção das respostas

A partir do ficheiro com a solução são calculados os índices que correspondem à posição da resposta correta a cada questão na tabela, considerando para isso a ordem alfabética (sendo que o 'a' corresponde ao índice 1) ou, para o caso do verdadeiro e falso, verdadeiro corresponde ao índice 1 e falso ao índice 2.

Estes índices são depois comparados com a posição das respostas do aluno às várias questões e a resposta é considerada correta caso a posição seja igual ao índice.

Ainda durante a correção se mais do que uma resposta for detectada por questão, a resposta considera-se errada.

O índice 0 não é utilizado pois corresponde à coluna que contém o número da questão.

III. RESULTADOS

É apresentada uma janela apenas com a tabela como ROI onde são apresentados retângulos à volta da resposta detectada indicando se esta é correta (retângulo verde) ou errada (retângulo vermelho), se a resposta for errada um retângulo amarelo é apresentado representando a resposta correta. Em rodapé é apresentado o número mecanográfico do aluno, seguido do número de respostas acertadas e erradas, bem como a nota obtida no exame (calculada através das cotações fornecidas). Esta imagem é guardada no disco na pasta Results com o seguinte formato NMEC.jpg.

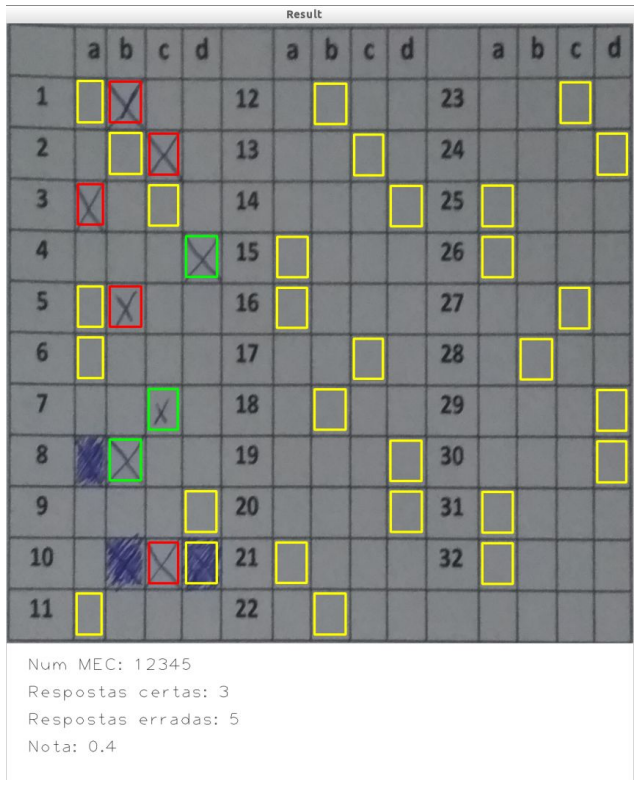

Fig. 6 - Correção escrita para ficheiro.

IV. PROBLEMAS

O algoritmo está desenhado para funcionar com apenas uma tabela e não com múltiplas tabelas na mesma imagem/folha.

Dado que são realizadas transformações sobre a tabela de modo a corrigir a perspectiva, o algoritmo apenas funciona para tabelas completas (nas quais não falte nenhum canto).

V. REFERÊNCIAS

[1]http://answers.opencv.org/question/63847/how-to-extract-tables-from-an-image/

[2]http://docs.opencv.org/master/d1/dee/tutorial_moprh_lines_detection.html

[3]http://stackoverflow.com/questions/22519545/automatic-perspective-correction-opencv

[4]http://aishack.in/tutorials/sudoku-grabber-opencv-detection/