AIML-Powered OMR Sheet Bubble Detection and Alignment Tool

Preliminary Approach Submission

This proposed preliminary approach integrates traditional image processing with deep learning to develop a scalable, robust solution to OMR Sheet Bubble Detection. The major challenges are in : skewed alignments, smudges, and low-quality scans.

Sheet Alignment :

The four corners of the OMR sheet are typically marked with distinct shapes, such as dots, squares, or other high-contrast markers. The detection process would involve converting the scanned OMR sheet to grayscale to simplify computations and reduce noise.

Thresholding: Applying binary thresholding or adaptive thresholding to isolate the high-contrast markers. Adaptive thresholding will be particularly effective for handling variations in lighting across the scanned image.

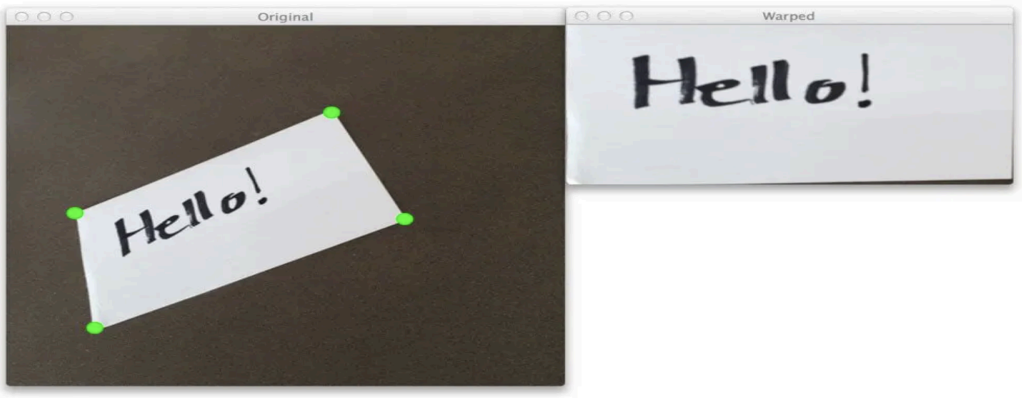
Contour Detection: Using OpenCV's cv2.findContours function to identify all potential contours in the image and then filtering these contours based on size, shape, and position to isolate the four corner markers.

Perspective Transformation:

Once the four corner markers are identified, the image is corrected using a perspective transformation to align the image with a standard coordinate system:

Calculating Transformation Matrix: Using OpenCV's cv2.getPerspectiveTransform, we will compute the matrix that maps the detected corner markers to their expected positions in a perfectly aligned sheet.

Transformation: Using cv2.warpPerspective to transform the scanned image and correct any skew or perspective distortion.



Source : https://pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/

Rotation Correction:
If the scanned image is rotated (e.g., upside down or sideways), the orientation can be corrected by analyzing the relative positions of the corner markers and rotating it to correct orientation using OpenCV's cv2.rotate function or by adjusting the perspective transformation matrix.

Source : https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html

Bubble Detection :

Bubbles on OMR sheets are typically circular or elliptical in shape and are evenly distributed across the sheet. These characteristics make bubble detection relatively straightforward using contour-based or morphological methods:

Contour Detection: Using OpenCV's cv2.findContours function, contours can be identified that likely correspond to bubbles. This function identifies boundaries in the binary image, which are then filtered based on their area and shape to identify circular or elliptical objects.
Hough Circle Transform: An alternative or complementary approach is the **Hough Circle Transform**, available in OpenCV. This method identifies circular shapes in the image, making it particularly effective for bubble detection, where circles are marked.

Bubble Size and Shape Analysis:

After detecting the contours or circles, the next step would be to filter out irrelevant or non-bubble shapes:
Shape Filtering: Based on predefined criteria (e.g., area, aspect ratio), any detected objects that do not correspond to the expected shape of a bubble can be filtered out.
Size Thresholding: Since the bubbles will have a certain size range, the detected contours are further filtered by their area, ensuring only bubbles of a specific size are considered.

Mark Detection:

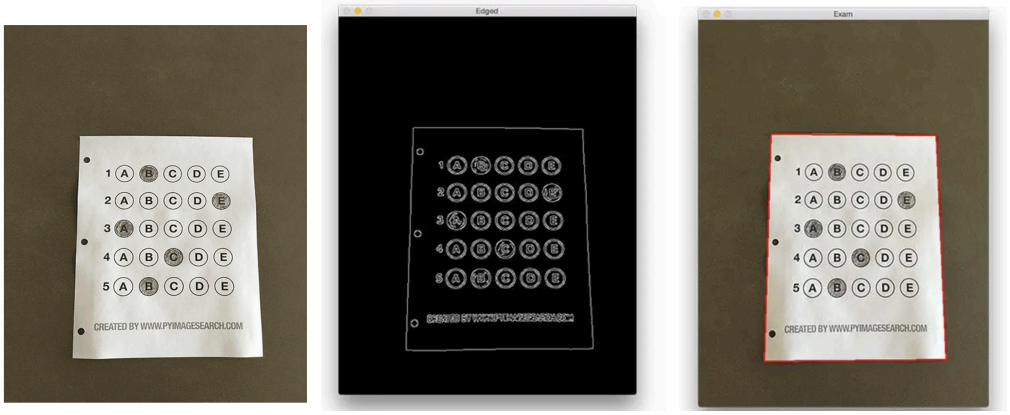
After detecting potential bubbles, the next step will be to interpret whether each bubble is marked (filled) or unmarked (empty). This can be done by evaluating the intensity of pixels within the detected bubble contour:

Pixel Intensity: For each detected bubble, analyzing the pixel intensity inside the detected circle or contour. If the pixels inside the bubble are darker (indicating ink or shading), the bubble is considered marked. If the pixels are lighter (indicating no marking), the bubble is considered unmarked.

Thresholding for Marked vs. Unmarked: A pixel intensity threshold is set to distinguish between marked and unmarked bubbles. This threshold can be dynamic and adjusted based on the scanning quality (e.g., resolution, scan brightness).

**For more complex scenarios, such as distinguishing between smudges and actual marks, a lightweight convolutional neural network (CNN) can be trained to classify bubbles as marked or unmarked. The CNN will be trained on a labeled dataset of cropped bubble regions and optimized for high precision and recall.

Edge detection algorithms, such as Canny Edge Detection, to identify areas where a mark might be partially present can be used.



Source :
https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/

References :

1. https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/
2. https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html
3. https://pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/