# Documentation for Bubble Sheet MCQ Checker Code

## Overview

This project processes a bubble sheet image, identifies marked answers for 10 multiple-choice questions (MCQs), and evaluates the answers against a predefined answer key. The steps include image preprocessing, contour detection, bubble detection, and grading.

## 1. Importing Required Libraries

import cv2  
import numpy as np  
import matplotlib.pyplot as plt

- OpenCV (cv2): Used for image processing tasks such as reading, resizing, and edge detection.  
- NumPy: Facilitates mathematical operations and array manipulations.  
- Matplotlib: Visualizes images and results.

## 2. Image Preprocessing

### Loading and Grayscale Conversion

image = cv2.imread('images/bubble\_sheet.jpg')  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

- Reads the input image and converts it to grayscale, which simplifies further processing by reducing color information.

### Blurring and Edge Detection

blurred = cv2.GaussianBlur(gray, (5, 5), 0)  
edges = cv2.Canny(blurred, 75, 200)

- GaussianBlur reduces noise and smoothens the image.  
- Canny edge detection highlights edges for contour detection.

## 3. Finding and Sorting Contours

contours, \_ = cv2.findContours(edges.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
contours = sorted(contours, key=cv2.contourArea, reverse=True)

- Detects contours in the image and sorts them by size to identify the bubble sheet's outer boundary first.

### Approximating the Bubble Sheet Boundary

for contour in contours:  
 perimeter = cv2.arcLength(contour, True)  
 approx = cv2.approxPolyDP(contour, 0.02 \* perimeter, True)  
 if len(approx) == 4:  
 bubble\_sheet\_contour = approx  
 break

- Approximates the largest rectangular contour, which represents the bubble sheet.

## 4. Perspective Transformation

warped = four\_point\_transform(gray, bubble\_sheet\_contour.reshape(4, 2))

- Applies a perspective transformation to align the bubble sheet for accurate processing.

## 5. Bubble Detection

### Dividing into Sections

questions = np.vsplit(warped, 10)

- Splits the transformed image into sections for each question.

### Detecting Filled Bubbles

for question in questions:  
 contours, \_ = cv2.findContours(question, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 # Process each contour to determine the marked answer.

- Identifies and processes contours for each question section to determine filled bubbles.

## 6. Grading the Responses

### Defining the Answer Key

ANSWER\_KEY = {0: 'A', 1: 'C', 2: 'B', ...}

- Predefines the correct answers for each question.

### Comparing Detected Answers

if detected\_answer == correct\_answer:  
 score += 1

- Compares the detected answer with the predefined answer key and calculates the total score.

## 7. Visualizing Results

cv2.putText(image, f"Score: {score}/10", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)  
plt.imshow(image)

- Displays the final score on the image and visualizes the results using Matplotlib.

## 8. Helper Functions

### Four Point Transformation

def four\_point\_transform(image, points):  
 # Maps the points of the bubble sheet to a top-down perspective.

- Transforms the bubble sheet to ensure correct alignment and processing.

## Conclusion

This program leverages OpenCV and Python for automated bubble sheet checking. By combining image processing techniques and simple logic, it provides accurate results with visual feedback. Each code block addresses a specific step in the workflow, ensuring modular and readable implementation.