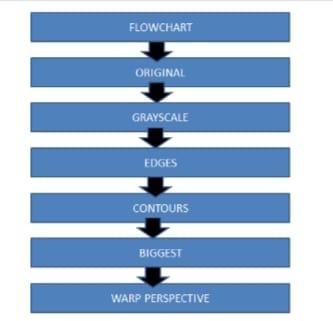
**PROJECT REPORT {BY-DEVIKA GAUTAM}**

**Project**: MCQ SHEEET CHECKING SYSTEM/OMR BUBBLE SHEET CHECKER

**INTRODUCTION:** Optical Mark Reader(OMR) is an automatic machine for evaluating multiple choice question scripts. The OMR checks between an input answer sheet and the template answer sheet. OMR machines are being used to correctly evaluate MCQ scripts for centralized board examinations . However these machines such as the Scantron OMR readers are not feasible for usage in smaller scale, such as examina- tions in schools due to the expense of the machines. In such cases the examiners manually check the MCQ scripts which is very time consuming. This is where our application can come in very handy, by removing the manual labor of checking the scripts. OMR machines typically require compatible OMR paper which are costlier than regular paper.

**REVIEW:**

****

**Methodology:**

* **Step #1:** Image Detection
* **Step #2:** Apply a perspective transform to extract the top-down, birds-eye-view of the exam.
* **Step #3:** Extract the set of bubbles (i.e., the possible answer choices) from the perspective transformed exam.
* **Step #4:** Sort the questions/bubbles into rows.
* **Step #5:** Determine the marked (i.e., “bubbled in”) answer for each row.
* **Step #6:** Lookup the correct answer in our answer key to determine if the user was correct in their choice.
* **Step #7:** Repeat for all questions in the exam.

**Experiments : Implementation of algorithm**

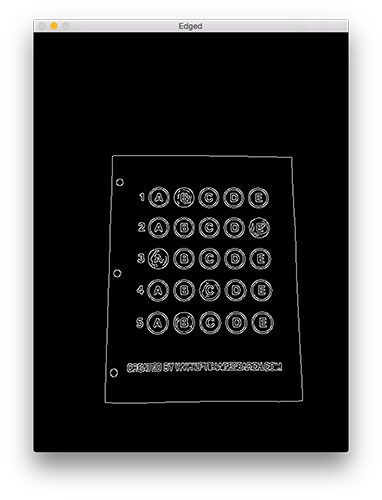
**1.**Import the necessary packages

2.Construct the argument parse and parse the arguments;define the answer key which maps the question number; to the correct answer

ANSWER\_KEY = {0: 1, 1: 4, 2: 0, 3: 3, 4: 1}

**3.** Next, let’s preprocess our input image;we load our image from disk, followed by converting it to grayscale, and blurring it to reduce high frequency noise.

We then apply the Canny edge detector  to find the *edges/outlines* of the exam.(Below I have included a screenshot of exam after applying edge detection)



**Figure 1 :**Applying edge detection to our exam neatly reveals the outlines of the paper.

**4**.Now we will use it as a marker to apply a perspective transform to the exam, obtaining a top-down, birds-eye-view of the document.

**5.** Find contours in the edge map, larger contours will be placed at the front of the list, while smaller contours will appear farther back in the list;then initialize; the contour that corresponds to the document; ensure that at least one contour was found; the sorted contours; approximate the contour; if our approximated contour has four points,; then we can assume we have found the paper.

**Figure 2:** An example of drawing the contour associated with the exam on our original image, indicating that we have successfully found the exam.



**6**. As this area corresponds to the outline of the exam.Now that we have used contours to find the outline of the exam, we can apply a perspective transform to obtain a top-down, birds-eye-view of the document:

- apply a four point perspective transform to both the

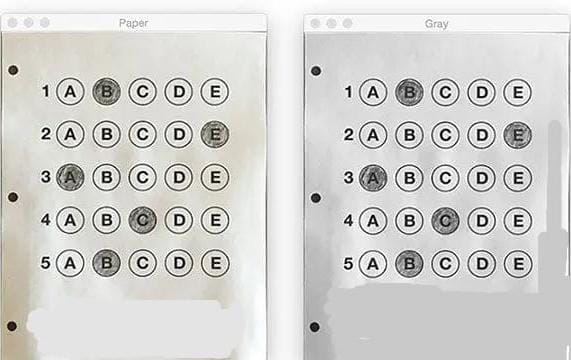
- original image and grayscale image to obtain a top-down

- birds eye view of the paper

In this case, we’ll be using my implementation of the **four point transform** function which:

1.Orders the *(x, y)*-coordinates of our contours in a *specific,* reproduciblemanner.

2.Applies a perspective transform to the region,this function handles taking the “skewed” exam and transforms it, returning a top-down top-down view of the document:



**Figure 3:**Obtaining a top-down, birds-eye view of both the original image *(*left) along with the grayscale version (right*)*.

**7.**We found our exam in the original image.Now we have to apply a perspective transform to obtain a 90 degree viewing angle of the document.For grading the document we will starts with **binarization**, or the process of thresholding/segmenting the *f*oreground from the background of the image:

**8.** Apply Otsu's thresholding method to binarize the warped piece of paper, our exam is now a binary image:



**Figure 4:** Using Otsu’s thresholding allows us to segment the foreground from the background of the image.

**9.**The *background*of the image is black,while the foreground is white *.*This binarization will allow us to once again apply contour extraction techniques to find each of the bubbles in the exam:

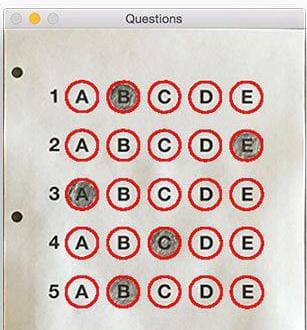
#find contours in the thresholded image, then initialize the list of contours that correspond to questions;loop over the individual contours to determine which regions of the image are bubbles.

# compute the bounding box of the contour, then use the bounding box to derive the aspect ratio in order to label the contour as a question, region should be sufficiently wide, sufficiently tall, and have an aspect ratio approximately equal to 1.

In order for a contour area to be considered a bubble, the region should:

1. Be sufficiently wide and tall (in this case, at least 20 pixels in both dimensions).
2. Have an aspect ratio that is *approximately* equal to 1.

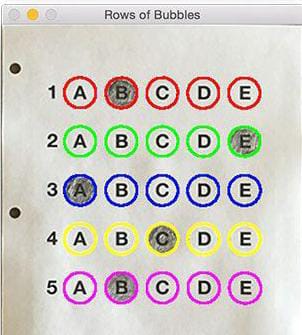
As long as these checks hold, we can update our questionCnts list and mark the region as a bubble.

**Figure 5:** Using contour filtering allows us to find all the question bubbles in our bubble sheet exam recognition software.only the question regions of the exam are highlighted and nothing else.

**10.**We can now move on to the “grading” portion of our OMR system: #First we must sort the question contours top-to-bottom, then initialize which will ensure that rows of questions that are closer to the top of the exam will appear irst in the sorted list nd also initialize a bookkeeper variable to keep track of the the total number of correct answers

# Start looping over our queations as each question has 5 possible answers, to loop over the question in batches of 5 apply NumPy array slicing and contour sorting to sort the current set of contours from left to right.

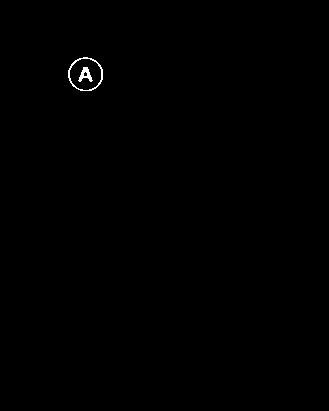
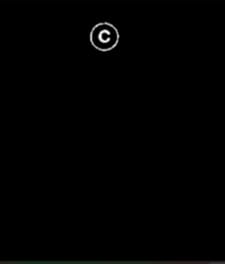
# sort the contours for the current question from left to right, then initialize the index of the bubbled answer.

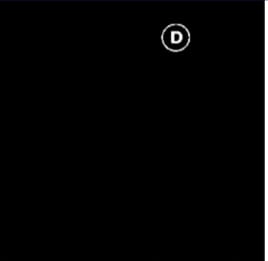
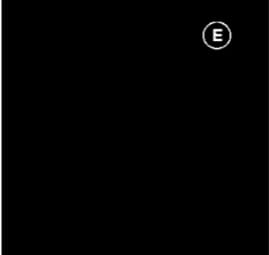


**Figure 6:** By sorting our contours from top-to-bottom, followed by left-to-right, we can extract each row of bubbles. Therefore, each row is equal to the bubbles for one question.

**11.** The next step is to determine which bubble is filled in the given row of bubbles and can be done by using thresh image and counting the number of non-zero pixels (i.e., *foreground pixels*) in each bubble region:

# loop over the sorted bubbles in the row,then construct a mask that reveals only the current "bubble" for the question. Apply the mask to the thresholded image, thenthe number of non-zero pixels in the bubble area if the current total has a larger number of total non-zero pixels, then we are examining the currently bubbled-in answer

  ** **

Clearly, the bubble associated with “B” has the most thresholded pixels, and is therefore the bubble that the user has marked on their exam.

**12.** This next code block handles looking up the correct answer in the ANSWER\_KEY updating any relevant bookkeeper variables, and finally drawing the marked bubble on our image:

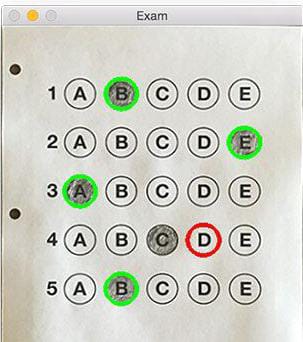
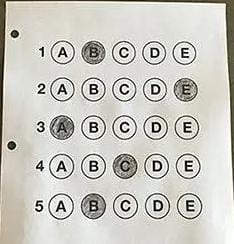
# \*correct\* answer

# check to see if the bubbled answer is correct

# draw the outline of the correct answer on the test.

# initialize the contour color and the index

**Results:** If the test taker is *correct*, we’ll highlight their answer in *green*. However, if the test taker made a mistake and marked an incorrect answer, we’ll let them know by highlighting the *correct* answer in red:

**Figure 8:** Drawing a “green” circle to mark “correct” or a “red” circle to mark “incorrect”.

Finally, our last code block handles scoring the exam and displaying the results to our screen. This is the output of our fully graded example image:

In this case, the reader obtained an 80% on the exam. The only question they missed was #4 where they incorrectly marked *“C”* as the correct answer (*“D”* was the correct choice).

**Conclusion:** Hence Our application provides a cheap solution for evaluating MCQ scripts. Although it may not be able to supplant the automated scanners worth thousands of dollars used in large scale standardized examinations; our application will be very useful in the day to day small scale examinations held in schools.

**References: 1.** https://pyimagesearch.com

**2.**“Education board bangladesh - computer center - at a glance.” http://www.educationboard.gov.bd/computer/atag lance.php

**3.**Tutorials regarding Computer vision,Robotics and AI Projects **(**Murtaza’s Workshop**)**

**4.**https://www.bogotobogo.com

**5.**www.addmengroup.com