Practical-CV / OMR-Scanner-and-Test-Grader-using-OpenCV

Code Issues Pull requests Actions Projects Security Insights

26 stars 19 forks 2 watching 1 Branch 0 Tags Activity Custom properties

Public repository

1 Branch 0 Tags Go to file t Go to file Add file Code

hackbansu readme edit 871ed8b · 5 years ago

images	initial commit	5 years ago
outputs	initial commit	5 years ago
README.md	readme edit	5 years ago
multiple_choice_template.psd	initial commit	5 years ago
test_grader.py	initial commit	5 years ago
test_grader_mine.py	initial commit	5 years ago

README

OMR Scanner and Test Grader using OpenCV

Scans an OMR and Grades the responses

Awesome OMR Scanner and Grader using just OpenCV and Python

All thanks to Adrian Rosebrock (from pyimagesearch) for making great tutorials. This project is inspired from his blog: Bubble sheet multiple choice scanner and test grader using OMR, Python and OpenCV. I have included the author's code and the one i wrote my self as well.

Key Points

1. Combines the following techniques:
    i. Building document scanner
    ii. Sorting the contours
    iii. Perspective transform to get top-down view
2. Steps involved:

   i. Detect the OMR sheet
   ii. Apply perspective transform to get the top-down view of the sheet
   iii. Extract out all the bubbles in the sheet
   iv. Sort them in rows
   v. Determine the answer bubble for each row
   vi. Match with the correct answer
   vii. Do this for all questions (all rows)
3. Assumptions:
   i. The app assumes that the OMR document we are scanning is the main focus of the image.
   ii. All 4 edges of the OMR document are visible in the image.
   iii. The largest rectangle available in the image will be the OMR document.
4. Stored the correct answer keys in a dict in python.
5. Used Canny edge detection for detecting the edges in the document and Gussian blur for reducing high frequency noise.
6. OpenCV has the way to get the top-down view of the image. Used that methodology to get the top-down view.
7. Used Otsu's thresholding method for thresholding.
8. Determined the bubbles using the aspect ratio of approx one (1) for it's bounding rectangle.
9. Used bitwise operations and masking to find the filled in bubble using the amount of shaded pixels in the bubble.

Requirements: (with versions i tested on)

1. python (3.7.3)
2. opencv (4.1.0)
3. numpy (1.61.4)
4. imutils (0.5.2)

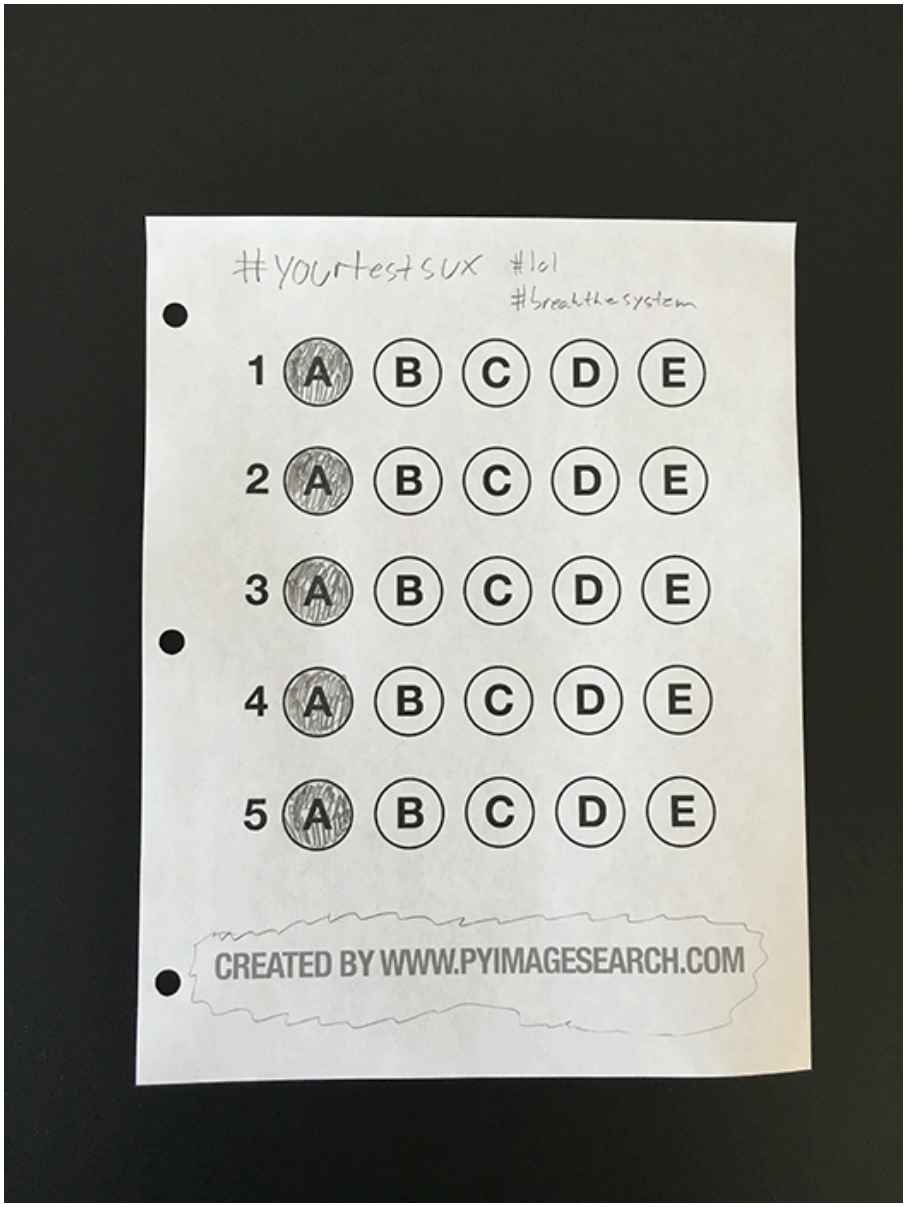
Commands to run the detection:

```
python test_grader.py --image images/test_02.png
```

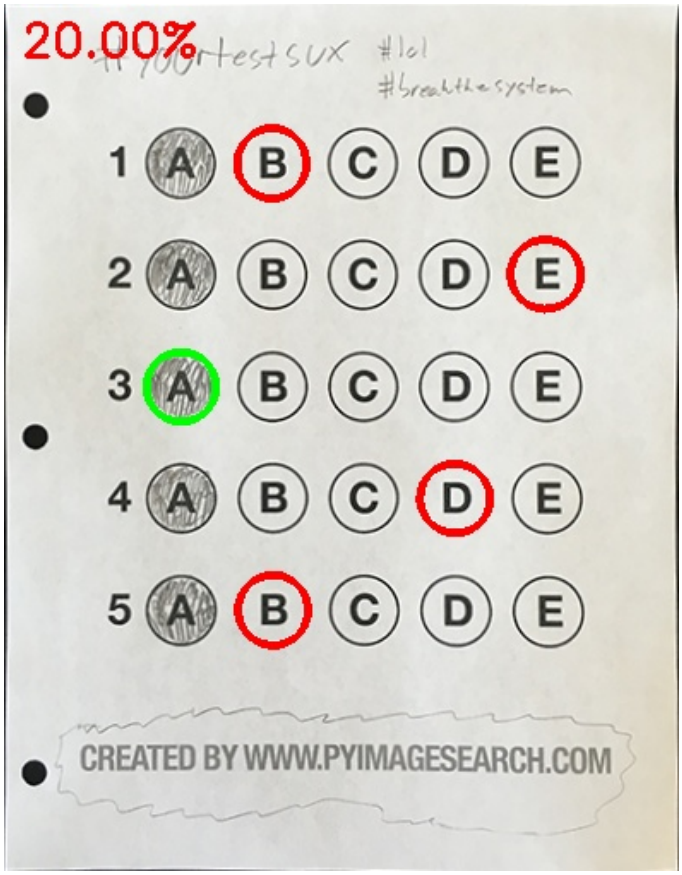
Results:

The results are pretty amazing. The grading system is working perfectly fine.

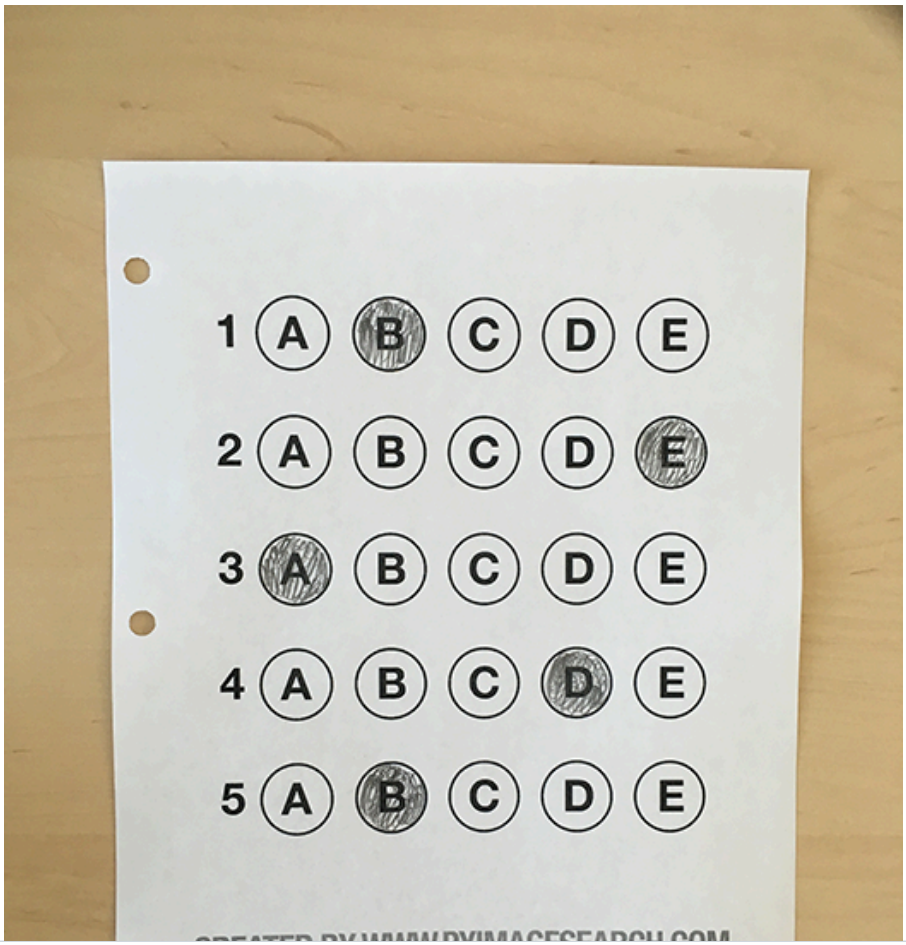
input


#yourtestsux #lol
#breakthesystem
1 A B C D E
2 A B C D E
3 A B C D E
4 A B C D E
5 A B C D E
CREATED BY WWW.PYIMAGESEARCH.COM

Output


20.00%
#lol
#breakthesystem
1 A B C D E
2 A B C D E
3 A B C D E
4 A B C D E
5 A B C D E
CREATED BY WWW.PYIMAGESEARCH.COM

input



Releases

No releases published

Packages

No packages published

Languages

- **Python** 100.0%