Data Science Blogathon 2024_From_Data_to_Intelligence

Problem Statement:

- In education, test results administration and proper grading system enhances fairness while providing timely results to the learners.
- There is a significant problem in using the traditional method of grading through mark entry
- The main disadvantage of manual grading is that it is slow, prone to mistakes as well as not very consistent.
- COTS based Commercial Optical Mark Recognition (OMR) systems are costly and come with fixed specification.

Proposed Solution:

- Write an OpenCV based program that provides high accuracy on scanned test papers and automates grading of tests as an OMR scanner and test grader.

Tasks Required:

- Design the possible technique that can be implemented into a system that will give the ability to read OMR test sheets and grading it with high efficiency.
- Admirably assess and recognize observable bubbles on the test sheets/foils.
- Map Data and Calculate Scores.
- Handle Errors and Inconsistencies.

Note:

- According to the desired image preprocessing results, methods of changing lighting, skew and paper quality will be applied to scanned images.
- Create an easy to navigate interface for users, particularly educators, that allows for a efficient test scoring system.

Through your participation to this challenge, you will contribute to the development of test grading, thus improving the overall tests solutions within the learning sector.

If you wish there was a faster, more efficient way to assess student performance? Look no further! Here is how to leverage the power of OpenCV, a free and open-source computer vision library, to build your very own OMR scanner and test grader. For simplicity I have divided the article in following sections and subsections:

Section 1: Understanding OMR Technology

- Definition
- Elements of an OMR System
- Working of OMR
- Benefits of Benefits of Automating Test Grading with OMR
- Applications of OMR
- Limitations of OMR

Section 2: Introduction to OpenCV

- What is OpenCV?
- Why is OpenCV particularly suitable here?
- Applications of OpenCV

Section 3 : Antecedents to Starting

Section 4: Workflow to Building the OMR Scanner and Test Grader

Section 5: Step-by-Step Code Implementation

- Detailed Code
- Explanation
- GitHub link

Section 6: The Challenges and Their Solutions

Section 7: Potential Improvements

Section 8: Conclusion

Section 9: Others

- Additional Resources
- FAQs

Section 1: Understanding OMR Technology

Definition

- OMR means Optical Mark Recognition

- OMR is an electronic system that scans and reads human-marked information on specially designed paper /forms.

Elements of an OMR System

1. Scanner: An apparatus used in scanning the digital image of the document.

2. Processing Software: An application to perform image processing on the scanned image and identify regions where marks have been made.

3. Output System: Summarizes and presents output in working format.

Working of OMR

1. Form
    - The sheets are provided with particular areas which are intended for marking mainly little circles or bubbles for instance those in the conventional multiple choice questions (MCQs).
    - These portions are provided in a way that would facilitate this system to identify these portions without any problem.

2. Scanning the Form
    - An OMR captures the image of an optical mark recognition sheet on a computer by scanning it in minimum time and maximum precision.

3. Processing the Image
    - Software looks at specific points and scans an image.
    - Techniques like thresholding are then applied to determine the marked bubbles (darker portion) from the empty bubbles (lighter portion).

4. Interpreting the Marks
    - According to the design of the OMR sheet, the computer software interprets the locations of the darkened spaces.
    - Reasoning these spaces with the correct answers.

5. Output
    - Last but not least, software provides results often scores or data reports.

Benefits of Automating Test Grading with OMR:

- Speed and Efficiency
    - Pass numerous tests in a short amount of time
    - Consume less time and effort of educators.
    - Efficient as it reduces human interference which leads to unbiased results.

- Consistency
    - Reduces bias by ensuring grading criteria developed for one test are applied to another test.

- Data Collection
    - OMR systems can store data so that it can be used for further analysis.

Applications of OMR:

- Commonly used when administering a test for multiple choice questions.
- Employed in gathering information from questionnaires and questionnaires where respondents tick their preferences.
- Other companies employ OMR in inventory management as workers use codes on the forms to indicate the level of stock.

Limitations of OMR:

- Limited Answer Formats:
- Form Design
- Blots or some other shapes that make accidental marks can cause mistakes.

In conclusion, OMR is an effective software solution for automatic scoring and analysis of filled forms in different fields, especially where the amount of data is very large

Section 2: Introduction to OpenCV

What is OpenCV?

- It is a Python library.
- Open Source Computer Vision Library
- It is also an open-source library.
- It was originally created by Intel in the year 1999.
- It offers a broad variety of tools and functionalities for computer vision and image processing.
- It was originally created by Intel corporation back in 1999.

Why is OpenCV particularly suitable here?

OpenCV is a building block and perfect fit to develop efficient and accurate OMR scanner and test grader because -

- It gives us tools required for building stable and precise OMR system.

- It provides numerous utilities such as image filtering and thresholding, required for OMR processing.
- It should be noted that being an open source it is free therefore no charges for acquiring a license.
- If we got stuck there is a friendly and vibrant community that can provide plenty of information and assistance.
- It is very efficient in processing and is capable of processing the images and videos in real time. And that is why OpenCV develop fast and responsive OMR system to work with big amount of test sheets.
- OpenCV is a computer vision library that supports multiple platforms, so it can be used on any operating system.
- It can be easily used with other libraries and frameworks including, but not limited to NumPy and scikit-image .
- Allows developers to create robust and flexible OMR system.

Applications of OpenCV

There are many other applications of OpenCV; however, listing some examples:

- Object Detection and Recognition
- Image Filtering and Enhancement
- Video Analysis
- Feature Detection and Matching
- Camera Calibration
- Gesture Recognition
- Medical Imaging
- Document Analysis

Section 3 : Antecedents to Starting

Before we dive into the code, let's ensure you have the necessary tools:

A) Python

- Python is a widely used high-level programming language.

- To write and execute code in python, we first need to install Python on our system.
- Installing Python is very easy. Here are its steps.

Step 1 − Select Version of Python to Install (Preferably Python 3)

Step 2 − Download Python Executable Installer

- Go to official site of python (www.python.org), move to the Download for Windows section.
- All the available versions of Python will be listed. Select the required version.
- Click on Download.

Step 3 − Run Executable Installer

- For instance if we downloaded the Python 3.9.1 Windows 64 bit installer.
- Run the installer.
- Make sure to select both the checkboxes at the bottom and then click Install New.
- On clicking the Install Now, The installation process starts.
- Installation process takes few minutes to complete and wait till installation is successful.

Step 4 − Verify Python is successfully installed on Windows

- To ensure this follow the given steps −
- Open the command prompt.
- Type ‘python’ and press enter.
- The version of the python which you have installed will be displayed if the python is successfully installed on your windows.

Step 5 − Verify Pip was installed

- Pip is a powerful package management system for Python software packages.
- Thus, make sure that you have it installed. To verify if pip was installed
- Open the command prompt.
- Enter pip –V to check if pip was installed.

B) OpenCV

- Install OpenCV using pip.
- To do so use command: pip install opencv -python

Installation Command: pip install opencv-python

C) NumPy

- Another helpful library for numerical operations (usually installed with OpenCV).
- Install NumPy using pip; the installation command is : pip install numpy

Installation Command: pip install numpy

- Usage of NumPy in context of OMR scanner and test grader is as follows :
    - Base package providing support for arrays and matrices.
    - Effective management and processing of image data in arrays.
    - Calculating points and conducting mathematical operations on points (e. g. , calculating distances).
    - Generation of array of points for perspective transformation.

D). imutils

- Install imutils using pip, the installation command is : pip install imutils

Installation Command: pip install imutils

- Usage of imutils library in context of OMR scanner and test grader is as follows:
    - Simplifies common image processing tasks and operations.
    - Reducing complexities associated with contours and their management.
    - Provide utility functions for image transformations and resizing.
    - Provides utility functions to sort contours and handle image paths.

Section 4: Workflow to Building the OMR Scanner and Test Grader

1. Setting Up the Environment:
    1. Install OpenCV
    2. Install Additional Libraries

2. Designing the OMR Sheet:
    1. Design the Sheet
    2. Ensure Easy Detection

3. Scanning the OMR Sheet:
    1. Capture Image
    2. Preprocess Image

4. Processing the Image:

    1. Convert to Grayscale
    2. Thresholding
    3. Contour Detection

5. Analyzing the Marked Bubbles:
    1. Shape Analysis
    2. Bubble Identification

6. Mapping Detected Marks to Answers:
    1. Compare with Answer Key
    2. Calculate Scores

7. Handling Errors and Inconsistencies:
    1. Error Correction
    2. Skewed Image Correction

8. Outputting Results:
    1. Generate Report
    2. Display Scores

Section 5: Step-by-Step Code Implementation

Detailed Code

Python

```
# code
print("GFG")
import subprocess
import sys

def install(package):
    subprocess.check_call([sys.executable, "-m", "pip", "install",
package])

install("opencv-python")
install("numpy")
install("imutils")

import numpy as np
import cv2
import imutils
from imutils import contours

def order_points(pts):
    """
    Order points in the order: top-left, top-right, bottom-right, bottom-
left.
    Args:
    pts: Array of four points (x, y) from the image.

    Returns:
    rect: Array of points ordered as top-left, top-right, bottom-right,
bottom-left.
    """
    rect = np.zeros((4, 2), dtype="float32")
```

```
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]  # top-left
    rect[2] = pts[np.argmax(s)]  # bottom-right
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]  # top-right
    rect[3] = pts[np.argmax(diff)]  # bottom-left
    return rect

def four_point_transform(image, pts):
    """
    Apply a perspective transform to obtain a top-down view of the image.
    Args:
    image: Input image.
    pts: Array of four points (x, y) representing the corners of the
object.

    Returns:
    warped: The warped (transformed) image.
    """
    rect = order_points(pts)
    (tl, tr, br, bl) = rect
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")
    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
    return warped

# Load and preprocess the image
image_path = r"C:\Users\Shubh\Downloads\omr.png"
image = cv2.imread(image_path)

if image is None:
    raise FileNotFoundError(f"The image file {image_path} could not be
loaded. Please check the file path.")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
edged = cv2.Canny(blurred, 75, 200)

# Find contours in the edge map
cnts = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
docCnt = None

# Ensure at least one contour was found
if len(cnts) > 0:
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)

    # Loop over the sorted contours
    for c in cnts:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
```

```
        # If the contour has four points, assume it is the document
        if len(approx) == 4:
            docCnt = approx
            break

if docCnt is None:
    raise ValueError("No document contour found in the image.")

# Apply the four point transform to obtain a top-down view of the paper
paper = four_point_transform(image, docCnt.reshape(4, 2))
warped = four_point_transform(gray, docCnt.reshape(4, 2))
thresh = cv2.threshold(warped, 0, 255, cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]

# Find contours in the thresholded image
cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
questionCnts = []

# Loop over the contours
for c in cnts:
    (x, y, w, h) = cv2.boundingRect(c)
    ar = w / float(h)

    # Filter out contours that are not circular
    if w >= 20 and h >= 20 and ar >= 0.9 and ar <= 1.1:
        questionCnts.append(c)

# Sort the question contours top-to-bottom
questionCnts = contours.sort_contours(questionCnts, method="top-to-
bottom")[0]
correct = 0
Answer_key = {0: 1, 1: 4, 2: 0, 3: 2, 4: 0}

# Loop over each question in batches of 5
for (q, i) in enumerate(np.arange(0, len(questionCnts), 5)):
    cnts = contours.sort_contours(questionCnts[i:i + 5])[0]
    bubbled = None

    # Loop over the sorted contours
    for (j, c) in enumerate(cnts):
        mask = np.zeros(thresh.shape, dtype="uint8")
        cv2.drawContours(mask, [c], -1, 255, -1)
        mask = cv2.bitwise_and(thresh, thresh, mask=mask)
        total = cv2.countNonZero(mask)

        if bubbled is None or total > bubbled[0]:
            bubbled = (total, j)

    color = (0, 0, 255)
    k = Answer_key[q]

    # Check if the bubbled answer is correct
    if k == bubbled[1]:
        correct += 1

print(f"Correct Answers: {correct}")
```

Explanation to Code :

- Imports:
    - numpy as np: For numerical operations.
    - cv2: OpenCV library for image processing.
    - imutils: Utility functions for image processing.
    - contours: For contour sorting

- Functions:
    - order_points(pts): Orders points in a consistent manner (top-left, top-right, bottom-right, bottom-left).
    - four_point_transform(image, pts): Applies a perspective transform to get a top-down view of the image.

- Main Code:
    - Loads and preprocesses the image (grayscale, blur, edge detection).
    - Finds and sorts contours to identify the document.
    - Applies perspective transformation to get a top-down view of the document.
    - Thresholds the image to make bubbles easier to detect.
    - Finds and filters contours to identify question bubbles.
    - Sorts question contours and checks each question's answer against the answer key.
    - Prints the number of correct answers.

Please find my Github link to understand the code deeply:

https://github.com/Shubham-Saxena-16/OpenCV-Hacks-for-Educators-Build-a-Time-Saving-OMR-Scanner-in-Python

Section 6: The Challenges and Their Solutions

Common Challenges

While building your OMR scanner, the challenges I encountered are :

- Varying Lighting Conditions: Specify ‘Thresholding → Adaptive’ from the menus at the top of the IDE.

- Skewed Images: Perform a transformation to correct the skew that was followed as a previous step.

- Imperfect Marks: They should also use the methods of contour approximation and filtering to enhance detection.

Solutions

One day it all begins with this or that, and then it continues…Here are some ways to enhance your OMR scanner:

- Multiple Answer Support: First, each code implements the functionality of finding the answer/s to the posed question/s.

- Integration with Other Tools: Review the options of the software for your scanner to connect to the educational system and improve the assessment tools.

Section 7: Potential Improvements

The project effectively provides an OMR system using OpenCV where the marked answers can be detected and graded on an OMR sheet. It enhances the input image, finds out the contours of the document, rectifies the image and successfully computes the marked answers.

While this is a solid foundation, there's always room for improvement. Here are some areas to consider for further development:

- Use Adaptive Thresholding
- Use sophisticated noise reduction methods
- Develop a Simple GUI
- Make It Possible to Process Several OMR Sheets at Once
- In addition to scalability, it is also vital to optimize the processing pipeline for speed.
- Employ the use of Multi-Threading or Parallel Processing
- Enable Loading the Answer Key from a File
- Provide Visual Feedback on the Processed OMR Sheet
- Incorporate Machine Learning Algorithms

By making these enhancements, it becomes possible to increase OMR scanner flexibility, make the interface more intuitive, and enlarge the field of application to cover a vast variety of test configurations to provide an all-encompassing assessment.

Section 8: Conclusion

Based on this project (challenge), it is evident that OpenCV can be used to develop an OMR scanner and test grader. It makes the grading process more efficient and less time-consuming since it relies on image processing and computer vision toolkits. The implemented code successfully addresses the following key steps:

- Filtering the scanned image for noise removal and for edge enhancement.
- Localizing the region of the OMR sheet in the acquired image.
- Identifying individual bubbles on the sheet.
- Translating marked bubbles to answer keys to get scores.

- The code optimally pre-processes the input image, outlines the document boundaries, corrects the perspective distortions, and recognizes and grades the marked answers.

- Nevertheless, there are several aspects that can be enhanced.
    - Enhancements could include
        - better error handling
        - adaptive thresholding
        - improved preprocessing steps
        - advanced contour filtering techniques.
    - Based on the above findings the following can be done to improve the system's functionality and security:
        - integrating configurable parameters
        - implementing user-friendly GUI
        - providing detailed documentation

Final Thoughts

Creating OMR scanner with OpenCV is one of those rewarding projects that demonstrate how far computer vision can go in handling real-life problems. It can be a starting point for further work and improvements in the process of automated assessment. For your assignment you can play, you can tweak and innovate and come up with an asset that saves time and energy in grading and allows the educators to do what is most important, teaching students.

Section 9: Others

Additional Resources

1. Document to OpemCV
2. OpenCV-Python Tutorials
3. Tutorials for contrib modules:
4. Numpy Basic
5. Python Documentation

FAQs : (Frequently Asked Questions)

Question 1: What is an OMR scanner?

Answer: An Optical Mark Recognition (OMR) scanner is an equipment that captures marks on paper, which may include filled bubbles in multiple-choice examination answer scripts. It makes the grading process easier because it determines which options have been selected.

Question 2: In what way does OpenCV assist in the development of an OMR scanner?

Answer: OpenCV is a set of image processing and computer vision primitives that can be used to preprocess images, detect edges contiguous regions and contours, perform

transformations, and recognize patterns—all of which are critical for constructing an OMR scanner.

Question 3: Can I process multiple OMR sheets in one run?

Answer: Yes, you can enable batch processing by writing code that processes multiple images in a loop, possibly using multi-threading or parallel processing to speed up the process.

Question 4: How can I provide visual feedback on the processed OMR sheet?

Answer: You can draw contours around detected bubbles and use different colors to highlight correct and incorrect answers directly on the image. OpenCV functions like cv2.drawContours and cv2.putText can be used for this purpose.

Question 5: How can I load the answer key from an external file?

Answer: You can load the answer key from external files like JSON or CSV. Use Python's json module to read JSON files and the csv module to read CSV files. This approach adds flexibility and ease of updating the answer key.

Question 6: Is it possible to develop a GUI for the OMR scanner?

Answer: Yes, you can develop a simple GUI using libraries like Tkinter, PyQt, or Kivy. A GUI can allow users to upload OMR sheets, specify answer keys, and view results interactively, making the system more user-friendly.

Question 7:Can the system recognize handwritten digits or text?

Answer: Yes, by integrating Optical Character Recognition (OCR) techniques using libraries like Tesseract, the system can be expanded to recognize handwritten digits or text in designated areas of the OMR sheet.

Question 8: What are some common errors and how can I handle them?

Answer: Common errors include issues with image loading, contour detection, and thresholding. Improving error handling with detailed messages and checks at various stages of the process can help diagnose and resolve these issues effectively.

Question 9: How can machine learning enhance the OMR scanning process?

Answer: Machine learning algorithms can be used to classify contours based on shape and size more accurately, automatically identify the OMR sheet layout, and reduce the need for manual configuration. Incorporating machine learning can improve the robustness and accuracy of the system.

Question 10:Can I scan more than one OMR sheet at one time?

Answer: Yes, you can enable batch processing by writing code that processes multiple images in a loop, possibly using multi-threading or parallel processing to make the task faster.

Question 11::How is it possible to give the visual output of the processed OMR sheet?

Answer: It allows drawing contours around the detected bubbles and using different colors to highlight the correct and the incorrect answers on the image. OpenCV functions like cv2. drawContours and cv2. For this purpose, putText can be used.

Question 12:What is the procedure for loading the answer key from an external file?

Answer: The answer key can be also imported from other files such as JSON or CSV. To read the JSON files, use Python’s json module, and for CSV files, use Python’s csv module. This approach enhances flexibility and ease to update the answer key when necessary.